

Table of contents

Content	Page
Table of tables	2
Table of figures	2
Problem statement	3
Literature review	4
Introduction to Neural Network	4
Introduction to XGBoost	6
Introduction to Decision Trees	7
Introduction to Random Forests	8
Introduction to Naïve Bayes	9
Methodology	12
Data Preparation Phase	12
Data Collection	13
Data Exploration	13
Data Visualization	13
Data Preprocessing Phase	17
Data Cleansing	17
Analysis Phase	17
Neural Network	18
XGBoost	18
Decision Trees	18
Decision Trees and Random Forests	19
Naïve Bayes	19
Results and Discussions	19
Neural Network	20
XGBoost	22
Decision Trees	24
Decision Trees and Random Forests	29
Naïve Bayes	32
Model Selection	34
Model Prediction	35
Conclusions and Discussions	36
References	37

Table of tables

Content	Page
Table 1 Interpretation of correlation strength	16
Table 2 Identify relationship of x and y	16
Table 3 Summary of the model's interpretation	34

Table of figures

Content	Page
Figure 1 Neural Network architecture	4
Figure 2 XGBoost architecture	6
Figure 3 Decision Trees architecture	7
Figure 4 Random Forests architecture	8
Figure 5 Naïve Bayes architecture	9
Figure 6 Flow chart of this research	12
Figure 7 Data Exploration summary from the data frame in ipynb file	13
Figure 8 The boxplot of overall dataset	14
Figure 9 Histogram plot of the dataset	15
Figure 10 The correlation analysis	15
Figure 11 The result of checking the missing data	17
Figure 12 The accuracy of the neural network model	20
Figure 13 Confusion matrix of the neural network model	20
Figure 14 Classification report of the neural network model	20
Figure 15 Learning curve on the NN model	21
Figure 16 The accuracy of the XGBoost model	22
Figure 17 Confusion matrix of the XGBoost model	22
Figure 18 Classification report of the XGBoost model	23
Figure 19 Learning curve on the XGBoost model	23
Figure 20 The accuracy of the Decision Trees model with entropy	24
Figure 21 Confusion matrix of the Decision Trees model with entropy	25
Figure 22 Classification report of the Decision Trees model with entropy	25
Figure 23 The accuracy of the Decision Trees model with Gini	26
Figure 24 Confusion matrix of the Decision Trees model with Gini	26
Figure 25 Classification report of the Decision Trees model with Gini	27
Figure 26 Learning curve on a Decision Tree model	28
Figure 27 Example of a Decision Trees architecture by using Gini index	28
Figure 28 The accuracy of the Decision Trees and Random Forests model	29
Figure 29 Confusion matrix of the Decision Trees and Random Forests model	29

Content	Page
Figure 30 Classification report of the Decision Trees and Random Forests model	30
Figure 31 Learning curve on the Decision Trees and Random Forests model	30
Figure 32 Example of the Decision Trees and Random Forests architecture	31
Figure 33 The accuracy of the Naïve Bayes model	32
Figure 34 Confusion matrix of the Naïve Bayes model	32
Figure 35 Classification report of the Naïve Bayes model	33
Figure 36 Learning curve on the Naïve Bayes model	33
Figure 37 XGBoost prediction on test set	35
Figure 38 Confusion matrix of the XGBoost prediction on test set	35

Problem statement

Most disasters are water-related. Floods, landslides, storms, heat waves, wildfires, extreme cold, droughts and waterborne disease outbreaks are all becoming more frequent and more intense, mainly due to climate change. The impacts of disasters include loss of life and damage to water and sanitation infrastructure, such as waterpoints, wells, toilets and wastewater treatment facilities. Consequently, water quality monitoring, analysis, and prediction have emerged as important challenges in several uses of water in our life. In this research considers the importance of the safety water by making prediction using machine learning with selected algorithms for saving lives and livelihoods. Machine learning is one of the most important and famous decision support tools nowadays. Recent progress in machine learning has been driven both by the development of new learning algorithms and theory and by the ongoing explosion in the availability of online data and low-cost computation. The adoption of data-intensive machine-learning methods can be found throughout science, technology and commerce, leading to more evidence-based decision-making across many walks of life, including health care, manufacturing, education, financial modeling, policing, and marketing. All of these supports in machine learning are crucial for forecasting and motivating in the future applications. This research aims to develop the machine learning models for prediction water safe with given dataset and obtain the most accurate model. The prediction models conclude neural networks (NN), XGBoost, Decision Tree, Decision Trees and Random Forests and Naïve Bayes for binary classification problem.

Literature review

The literature review concludes the effectiveness of the article, paper and journal with commitment to use artificial intelligence, machine learning and deep learning trends for solving water safe issues.

Introduction to Neural Networks

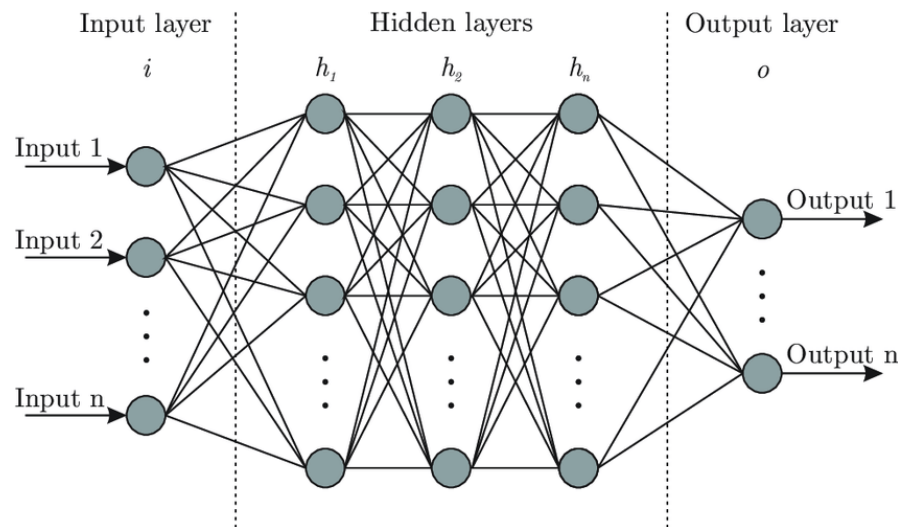


Figure 1 Neural Network architecture

Author. (n.d.). *Designing your neural networks*. Medium. Retrieved April 17, 2025, from <https://medium.com/data-science/designing-your-neural-networks-a5e4617027ed>

Neural Networks (NN) are computational models inspired by the human brain's interconnected neuron structure. They are fundamental to many machine learning algorithms today, allowing computers to recognize patterns and make decisions based on data. A neural network is a series of algorithms designed to recognize patterns and relationships in data through a process that mimics the way the human brain operates. Let's break this down: At its core, a neural network consists of neurons, which are the fundamental units akin to brain cells. These neurons receive inputs, process them, and produce an output. They are organized into distinct layers: an Input Layer that receives the data, several Hidden Layers that process this data, and an Output Layer that provides the final decision or prediction. The adjustable parameters within these neurons are called weights and biases. As the network learns, these weights and biases are adjusted, determining the strength of input signals. This adjustment process is akin to the network's evolving knowledge base. Before training starts, certain settings, known as hyperparameters, are tweaked. These determine factors like the speed of learning and the duration of training. They're akin to setting up a machine for optimal performance. During the training phase, the network is presented with data, makes a prediction based on its current

knowledge (weights and biases), and then evaluates the accuracy of its prediction. This evaluation is done using a loss function, which acts as the network's scorekeeper. After making a prediction, the loss function calculates how far off the prediction was from the actual result, and the primary goal of training becomes minimizing this "loss" or error.

Backpropagation plays a pivotal role in this learning process. Once the error or loss is determined, backpropagation helps adjust the weights and biases to reduce this error. It acts as a feedback mechanism, identifying which neurons contributed most to the error and refining them for better future predictions. To adjust the weights and biases efficiently, techniques like "gradient descent" are employed. Imagine navigating a hilly terrain, and your goal is to find the lowest point. The path you take, always moving towards a lower point, is guided by gradient descent. Lastly, an essential component of neural networks is the activation function. This function decides whether a neuron should be activated based on the weighted sum of its inputs and a bias. To visualize the entire process, think of a neural network trained to recognize handwritten numbers. The input layer receives the image of a handwritten digit, processes the image through its layers, making predictions and refining its knowledge, until it can confidently identify the number.

Introduction to XGBoost

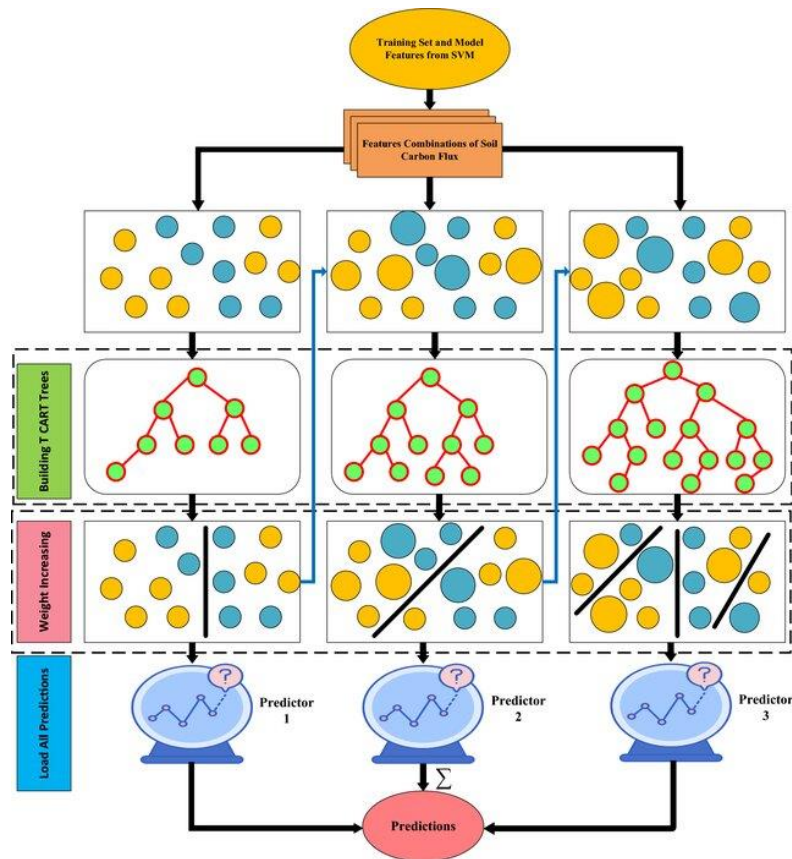


Figure 2 XGBoost architecture

Ding, H. (2024). Establishing a soil carbon flux monitoring system based on support vector machine and XGBoost [Figure]. *Soft Computing*, 28(21), 1–24. <https://doi.org/10.1007/s00500-024-09641-y>

XGBoost is an optimized implementation of Gradient Boosting and is a type of ensemble learning method. Ensemble learning combines multiple weak models to form a stronger model. XGBoost uses decision trees as its base learners combining them sequentially to improve the model's performance. Each new tree is trained to correct the errors made by the previous tree and this process is called boosting. It has built-in parallel processing to train models on large datasets quickly. XGBoost also supports customizations allowing users to adjust model parameters to optimize performance based on the specific problem.

It builds decision trees sequentially with each tree attempting to correct the mistakes made by the previous one. The process can be broken down as follows:

Start with a base learner: The first model decision tree is trained on the data. In regression tasks this base model simply predict the average of the target variable.

Calculate the errors: After training the first tree the errors between the predicted and actual values are calculated.

Train the next tree: The next tree is trained on the errors of the previous tree. This step attempts to correct the errors made by the first tree.

Repeat the process: This process continues with each new tree trying to correct the errors of the previous trees until a stopping criterion is met.

Combine the predictions: The final prediction is the sum of the predictions from all the trees.

Introduction to Decision Trees

Decision tree trained on all the iris features

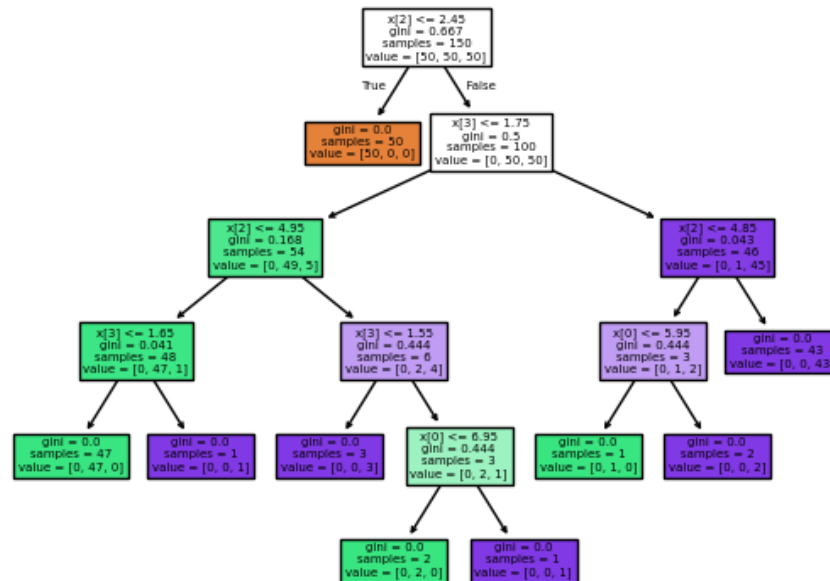


Figure 3 Decision Trees architecture

Scikit-learn developers. (n.d.). Decision Trees. Scikit-learn documentation. Retrieved April 17, 2025, from <https://scikit-learn.org/stable/modules/tree.html>

Decision tree is a simple diagram that shows different choices and their possible results helping you make decisions easily. A decision tree is a graphical representation of different options for solving a problem and show how different factors are related. It has a hierarchical tree structure starts with one main question at the top called a node which further branches out into different possible outcomes where:

Root Node is the starting point that represents the entire dataset.

Branches: These are the lines that connect nodes. It shows the flow from one decision to another.

Internal Nodes are Points where decisions are made based on the input features.

Leaf Nodes: These are the terminal nodes at the end of branches that represent final outcomes or predictions

We have mainly two types of decision tree based on the nature of the target variable: classification trees and regression trees.

Classification trees: They are designed to predict categorical outcomes means they classify data into different classes. They can determine whether an email is “spam” or “not spam” based on various features of the email.

Regression trees : These are used when the target variable is continuous It predict numerical values rather than categories.

Introduction to Random Forests

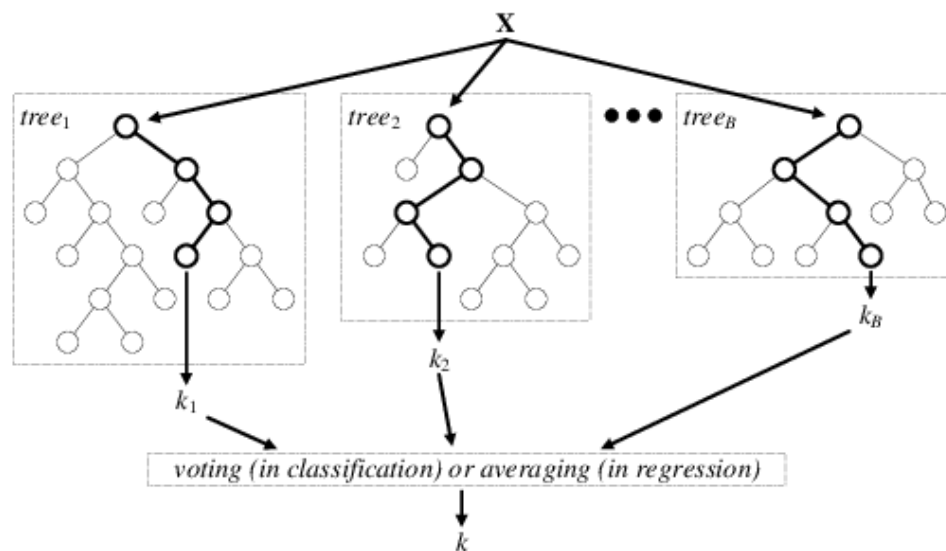


Figure 4 Random Forests architecture

Stern, M. K., Beck, J. E., & Woolf, B. P. (2003). Naive Bayes Classifiers for User Modeling. University of Massachusetts Amherst. Retrieved April 17, 2025, from https://www.researchgate.net/figure/Architecture-of-the-random-forest-model_fig1_301638643

Random Forest algorithm is a powerful tree learning technique in Machine Learning to make predictions and then we do voting of all the trees to make prediction. They are widely used for classification and regression task. It is a type of classifier that uses many decision trees to make predictions. It takes different random parts of the dataset to train each tree and then it combines the results by averaging them. This approach helps improve the accuracy of predictions. Random Forest is based on ensemble learning. Imagine asking a group of friends for advice on where to go for vacation. Each friend gives their recommendation based on their unique perspective and preferences (decision trees trained on different subsets of data). You then make your final decision by considering the majority opinion or averaging their suggestions (ensemble prediction). Random Forest builds multiple decision trees using random samples of the data. Each tree is trained on a different subset of the data which makes each tree unique. When creating each tree the algorithm randomly selects a subset of features or variables to split the data rather than using all available features at a time. This

adds diversity to the trees. Each decision tree in the forest makes a prediction based on the data it was trained on. When making final prediction random forest combines the results from all the trees. For classification tasks the final prediction is decided by a majority vote. This means that the category predicted by most trees is the final prediction. For regression tasks the final prediction is the average of the predictions from all the trees. The randomness in data samples and feature selection helps to prevent the model from overfitting making the predictions more accurate and reliable.

Introduction to Naïve Bayes

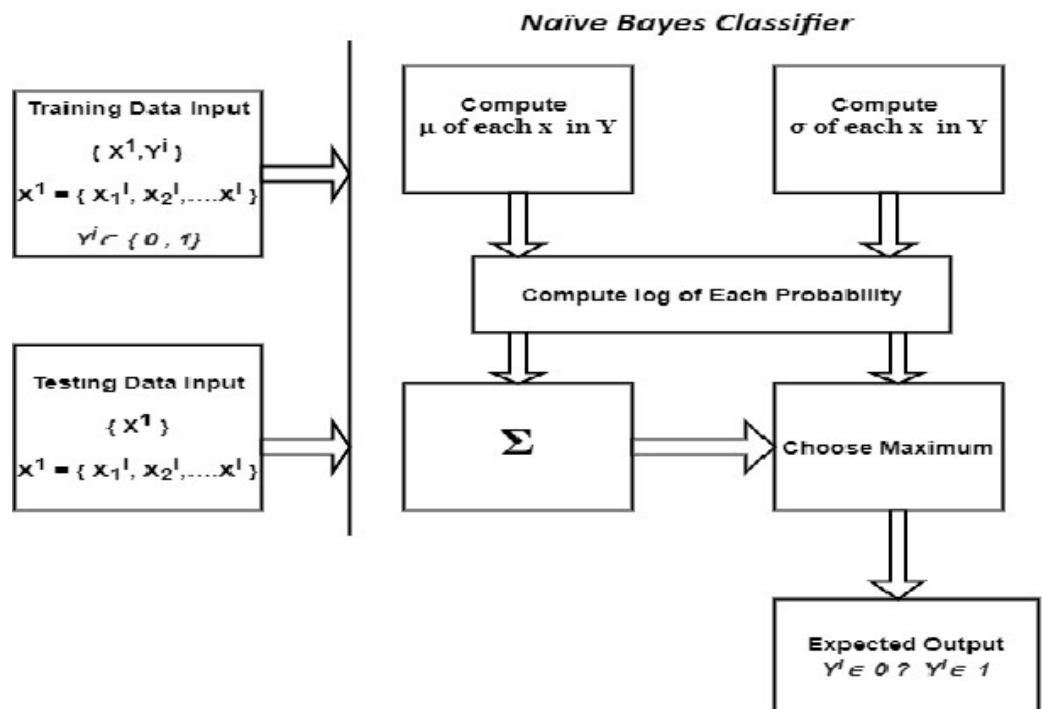


Figure 5 Naïve Bayes architecture

Berrar, D. (2018). Bayes' Theorem and Naive Bayes Classifier. Retrieved April 17, 2025, from https://www.researchgate.net/figure/Architecture-of-Naive-Bayes-Classifier_fig1_368646502

The Naïve Bayes classifier is a supervised machine learning algorithm that is used for classification tasks such as text classification. They use principles of probability to perform classification tasks. Naïve Bayes is part of a family of generative learning algorithms, meaning that it seeks to model the distribution of inputs of a given class or category. Unlike discriminative classifiers, like logistic regression, it does not learn which features are most important to differentiate between classes. Naïve Bayes is also known as a probabilistic classifier since it is based on Bayes' Theorem. It would be difficult to explain this algorithm without explaining the basics of Bayesian statistics. This theorem, also known as Bayes' Rule, allows us to "invert" conditional probabilities. As a reminder, conditional probabilities represent the probability of an event given some other event has occurred. Bayes' Theorem

is distinguished by its use of sequential events, where additional information later acquired impacts the initial probability. These probabilities are denoted as the prior probability and the posterior probability. The prior probability is the initial probability of an event before it is contextualized under a certain condition, or the marginal probability. The posterior probability is the probability of an event after observing a piece of data. A popular example in statistics and machine learning literature (link resides outside ibm.com) to demonstrate this concept is medical testing. For instance, imagine there is an individual, named Jane, who takes a test to determine if she has diabetes. Let's say that the overall probability having diabetes is 5%; this would be our prior probability. However, if she obtains a positive result from her test, the prior probability is updated to account for this additional information, and it then becomes our posterior probability.

In recent research of Torky Mohamed proposes a machine learning framework for classifying drinking water samples as safe or unsafe and predicting the Water Quality Index (WQI). Nine classification and six regression models were evaluated using a benchmark dataset. Among them, Random Forest and LightGBM showed superior performance in classification, achieving an average accuracy of 94.7%, while LightGBM and Extra Trees Regression excelled in WQI prediction with high accuracy (0.99 and 0.95) and low error rates. The results highlight the effectiveness of AI in water quality assessment for broader environmental applications.

The study of Dawood Thikra presents an Artificial Neural Network (ANN) model for predicting potable water quality failures in urban water distribution networks, using historical data from El Pedregal, Peru. The model, optimized through a scaled conjugate gradient algorithm, achieved 92% validity with low prediction errors (MAE: 0.08, RMSE: 0.15). Sensitivity analysis revealed that water quality, pressure, and maintenance practices significantly influence failure risk. The model offers a proactive, data-driven tool for policymakers to assess risk levels and implement preventive strategies for public health protection.

In comparison, the study of Kuthe Annaji developed a classification model for predicting Water Quality Classification (WQC) based on the Water Quality Index (WQI) using seven parameters. Support Vector Machine (SVM) and Extreme Gradient Boosting (XGBoost) algorithms were evaluated. XGBoost outperformed SVM, achieving 94% accuracy and a 6% misclassification rate, compared to SVM's 67% accuracy and 33% error. Cross-validation further confirmed XGBoost's superiority with an average accuracy of 90%. These results highlight XGBoost as a more effective model for water quality classification.

The study of Niyongabo Alian applies machine learning and deep learning techniques to forecast water quality and urban water consumption in Lake Tanganyika. Using a dataset with physicochemical parameters and consumption records, Random Forest, K-Nearest Neighbor, and SVM were used for classification, with Random Forest achieving the highest

accuracy (99.89%). For time-series prediction, GRU outperformed LSTM and BiLSTM, with strong R^2 and NSE scores for both water demand and quality index forecasting. The findings demonstrate the potential of these models for enhancing sustainable water resource management and urban planning.

Varalakshmi P. presents a methodology for assessing drinking water quality using statistical quality control and Bayesian algorithms to improve classification accuracy. The approach evaluates the impact of environmental factors such as industrial effluents and acid rain, aiming to determine water suitability for human consumption. The integration of statistical and probabilistic techniques enhances reliability in water quality classification.

Due to the research article in fact that this research will be applied with 5 satisfaction accurate models with assigned dataset full of Neural Network, XGBoost, Decision Trees, Decision Trees and Random Forests and Naïve Bayes.

Methodology

The methodology of this work is divided into 3 phases and sub-methods that conclude Data Preparation Phase, Data Preprocessing Phase and Analysis Phase followed by figure 6.

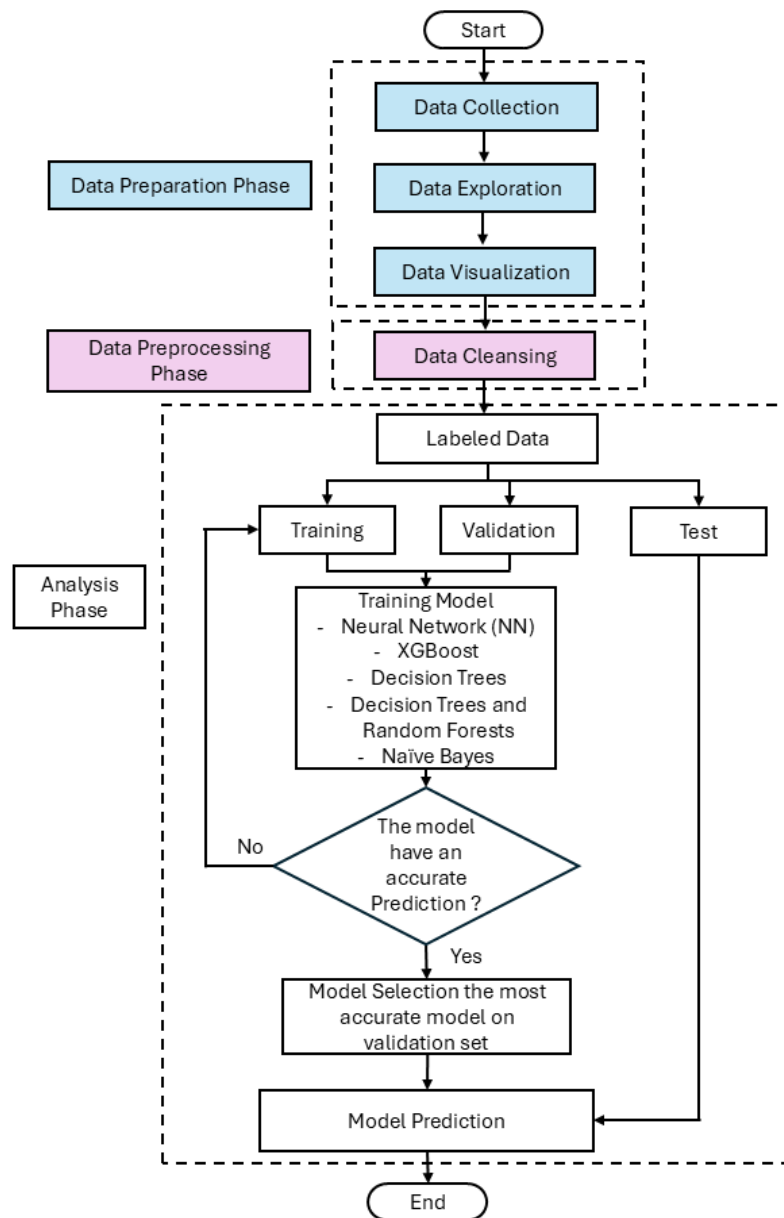


Figure 6 Flow chart of this research

Data Preparation Phase

As mentioned on the flow chart. The Data Preparation Phase consists of 3 sub-methods that are Data Collection, Data Exploration and Data Visualization.

Data Collection

Data collection and acquisition belong to given assignment data.

Data Exploration

This step can help to find initial patterns, characteristics and interesting points, especially to the data analytics roles. By the way picking the appropriate analysis method will be more advanced when using parallel to data visualization steps. The basic information of this data by reading xlsx file has 4,796 numbers with 21 columns. The 20 columns are aluminium, ammonia, arsenic, barium, cadmium, chloramine, chromium, copper, fluoride, bacteria, viruses, lead, nitrates, nitrites, mercury, perchlorate, radium, selenium, silver and uranium represented to be x value to predict the 1 column of is_safe that is represented to be y value in the binary classification problem. The characteristic of x variable is numerical data contrast to y variable that is categorical data. The total dataset contains 7,996 examples and is randomly divided into three parts: training data (4,796 samples – 60%), cross-validation data (1,600 samples – 20%), and test data (1,600 samples – 20%). Lastly, the dataset is well labelled so in the analytics phase I used supervised learning to train the model rather than use unsupervised learning.

```
... <bound method DataFrame.info of          aluminium ammonia arsenic barium cadmium chloramine chromium \
0      2.40      6.82      0.660      1.87      0.100      7.56      0.47
1      0.03     15.84      0.020      1.31      0.060      0.19      0.02
2      0.01      7.10      0.170      4.34      0.020      4.81      0.38
3      0.08     26.78      0.010      0.87      0.000      0.45      0.02
4      0.08     28.19      0.001      0.25      0.001      3.58      0.68
...      ...      ...      ...      ...      ...      ...      ...
4791     0.08      8.75      0.020      0.99      0.000      0.06      0.08
4792     0.31     29.13      0.390      1.91      0.030      3.58      0.46
4793     0.00     24.82      0.050      0.05      0.080      0.51      0.00
4794     0.04      0.51      0.040      0.87      0.010      0.06      0.06
4795     0.08     21.01      0.050      0.03      0.080      0.45      0.09

      copper fluoride bacteria ... lead nitrates nitrites mercury \
0      0.33      0.82      0.00 ... 0.093      2.09      1.48      0.004
1      0.06      1.21      0.00 ... 0.042     10.34      0.45      0.005
2      0.86      1.30      0.89 ... 0.020     10.01      1.92      0.004
3      0.75      0.20      0.00 ... 0.104      5.69      0.48      0.006
4      1.29      1.04      0.49 ... 0.079     15.05      1.09      0.005
...      ...      ...      ...      ...      ...      ...      ...
4791     0.16      0.39      0.41 ... 0.037     17.53      0.57      0.005
4792     0.05      1.28      0.77 ... 0.001      2.99      2.14      0.002
4793     0.06      1.27      0.00 ... 0.117     12.88      0.26      0.005
4794     1.50      0.92      0.44 ... 0.184     10.27      1.21      0.007
4795     1.49      0.65      0.63 ... 0.160     14.00      0.68      0.003
...      ...      ...      ...      ...      ...      ...      ...
4793      3.48      2.82      0.10      0.10      0.03      0
4794      0.82      1.94      0.04      0.06      0.08      0
4795      3.32      1.43      0.02      0.08      0.01      0

[4796 rows x 21 columns]>
```

Figure 7 Data Exploration summary from the data frame in ipynb file

Data Visualization

The visualization of the data is necessary tools for diving into the insights of data. About this work, I used popular libraries such as matplotlib and seaborn for making visualization tools

to detect the variability of the data, analyze, and plot the graph. To measure the data variability by making boxplot and histogram for showing patterns from Figure 8 to Figure 9 consequently. And the correlation analysis will be used to identifying how strong relationships between x and y and so on with x and x show on figure 10.

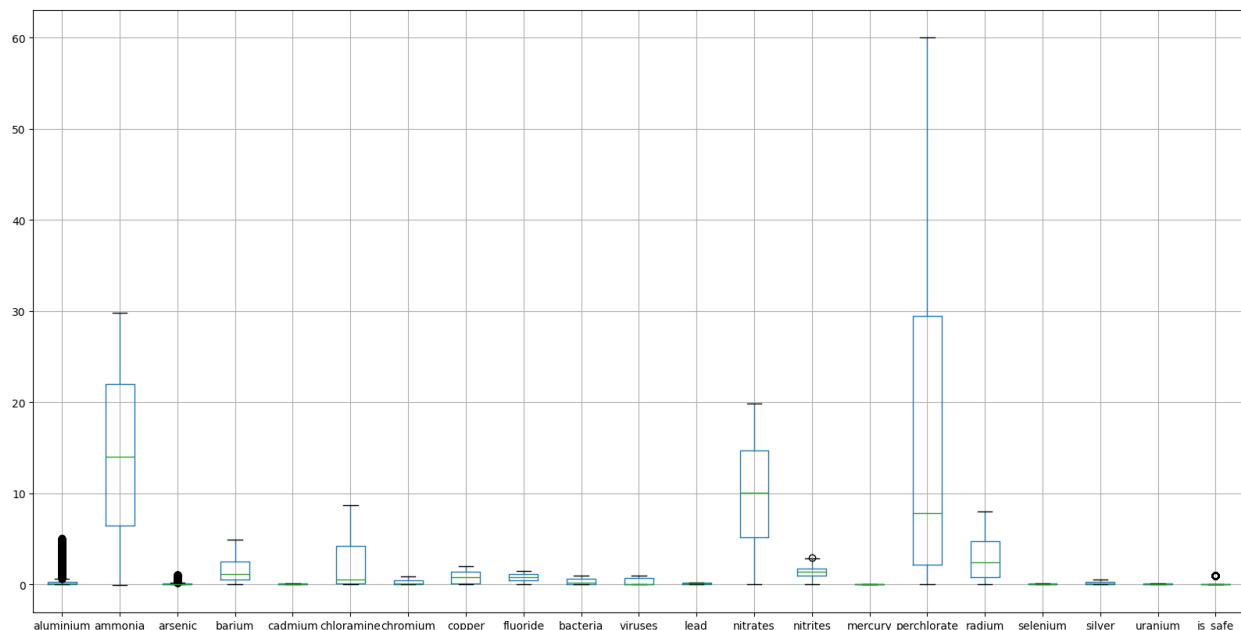


Figure 8 The boxplot of overall dataset

The boxplot visualizes the distribution of various water quality parameters, including chemical, biological, and heavy metal contaminants. Notable observations include significant variability in the concentrations of ammonia, nitrates, and particularly perchlorate, all of which exhibit wide interquartile ranges and high maximum values, indicating substantial fluctuation in their presence across samples. In contrast, elements such as cadmium, lead, selenium, silver, and uranium demonstrate low median values with narrow interquartile ranges, suggesting consistent and generally low concentrations. Several parameters, including aluminium, arsenic, and nitrites, show the presence of outliers, pointing to occasional spikes in their levels. The binary variable "is_safe", likely indicating overall water safety classification (e.g., safe = 1, unsafe = 0), shows a strong skew towards 1, implying that most samples meet safety standards. Overall, the boxplot highlights both the variability and stability among different contaminants and offers insights into potential risk factors in water quality monitoring. The histogram plot belongs to the boxplot.

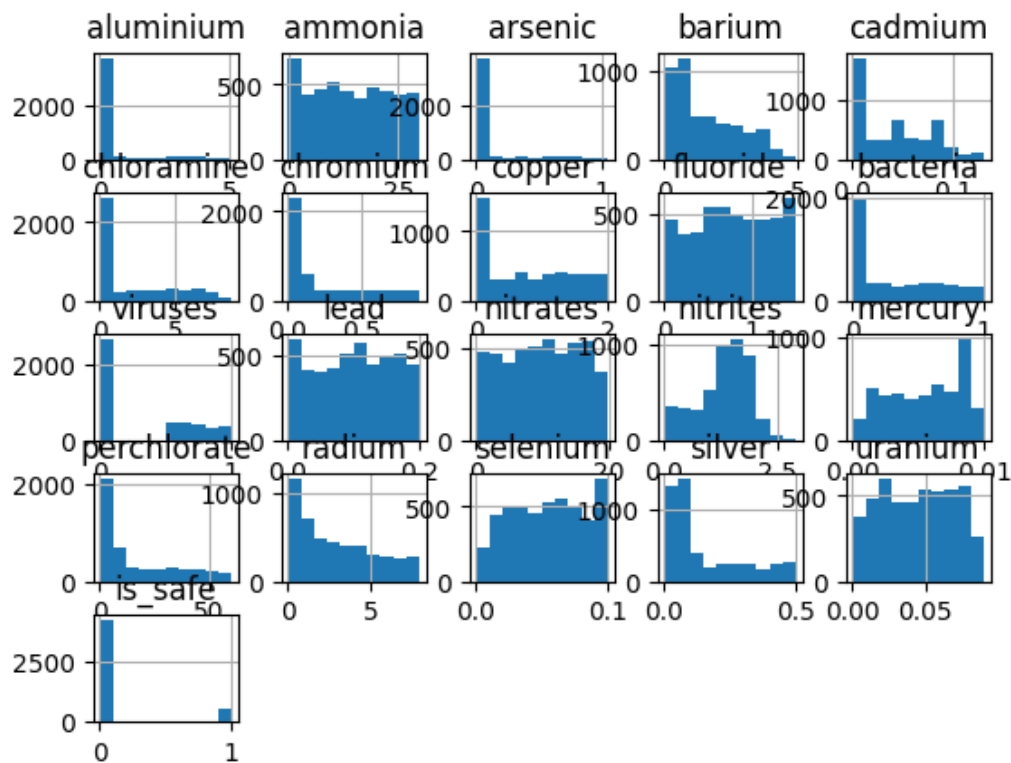


Figure 9 Histogram plot of the dataset

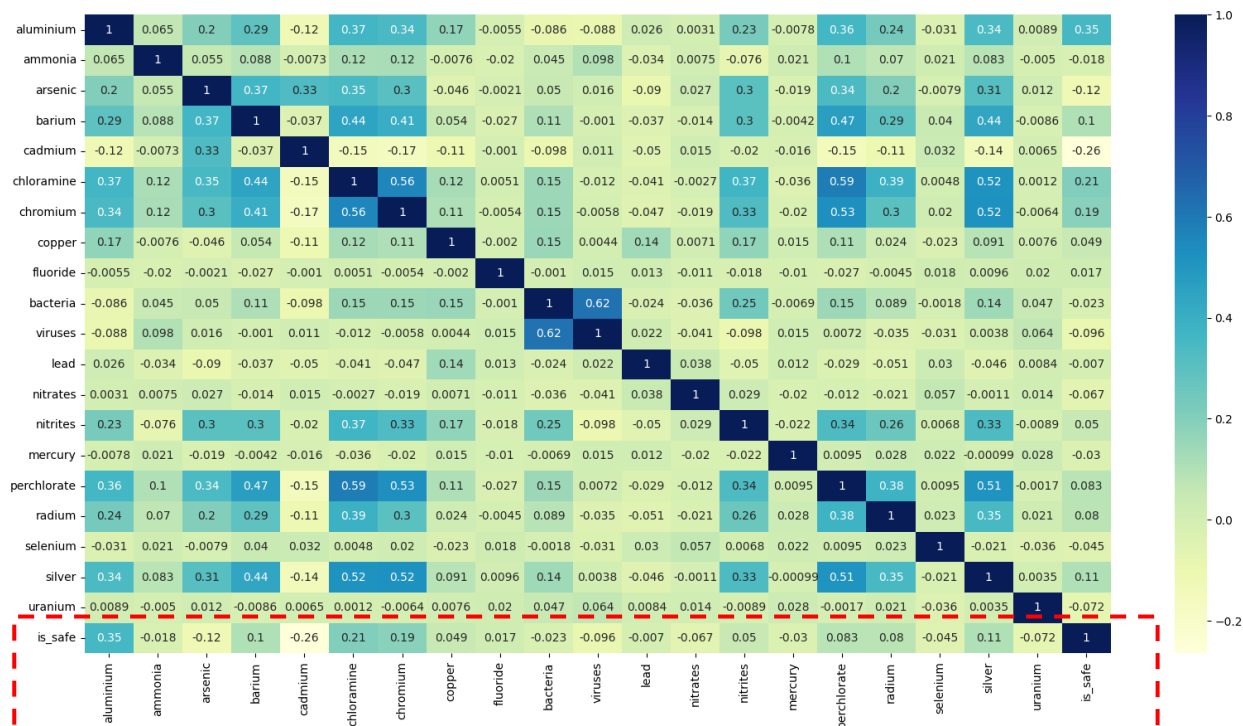


Figure 10 The correlation analysis

From the figure 10, graph plot shows that x variables have a negligible to moderate relationship with y variable from the strongest [aluminum – is_safe] about 0.35 (35%) to the lowest [lead – is_safe] about -0.007 (0.7%) consequently. The scoring criteria defines in table 1.

Table 1 Interpretation of correlation strength

Correlation Coefficient	Strength
0.00 – 0.10	Negligible
0.11 – 0.30	Weak
0.31 – 0.50	Moderate
0.50 – 1.00	Strong

Table 2 Identify relationship of x and y

Pair	Correlation Coefficient	Strong/Moderate /Weak	Positive/Negative
aluminum – is_safe	0.35	Moderate	Positive
ammonia – is_safe	-0.018	Negligible	Negative
arsenic – is_safe	-0.12	Weak	Negative
barium – is_safe	0.1	Negligible	Positive
cadmium – is_safe	-0.26	Weak - Moderate	Negative
chloramine – is_safe	0.21	Weak	Positive
chromium – is_safe	0.19	Weak	Positive
copper – is_safe	0.049	Negligible	Positive
fluoride – is_safe	0.017	Negligible	Positive
bacteria – is_safe	-0.023	Negligible	Negative
viruses – is_safe	-0.096	Negligible	Negative
lead – is_safe	-0.007	Negligible	Negative
nitrites – is_safe	-0.067	Negligible	Negative
nitrites – is_safe	0.05	Negligible	Positive
mercury – is_safe	-0.03	Negligible	Negative
perchlorate – is_safe	0.083	Negligible	Positive
radium – is_safe	0.08	Negligible	Positive
selenium – is_safe	-0.045	Negligible	Negative
silver – is_safe	0.11	Weak	Positive
uranium – is_safe	-0.072	Negligible	Negative

The x variables have many columns for complying with machine learning models so that can be used many computation times too. However, if I use the cut off (pruning) to reduce the dimension by dropping the low-correlation features (Negligible) might help with simple regression but not work for criteria for feature importance. In evaluated models with more complex such as neural networks, XGBoost, Random forests and etc. may capture the necessary or non-linear relationships with target and also with multicollinearity which can cause redundancy in linear models. In those cases, cutting off based on linear correlation could remove valuable. The future applies aim to improve model interpretability is cut off

features with weak correlation to reduce overfitting and use more advanced feature selection method or model-based importance metrics.

Data Preprocessing Phase

As far as I write on the flow chart. The Data Preprocessing Phase consists of 1 sub-method that is Data Cleansing.

Data Cleansing

This method is to check the missing data and operate to cope with it by coding.

```
aluminium      0
ammonia         0
arsenic         0
barium          0
cadmium         0
chloramine      0
chromium        0
copper          0
fluoride        0
bacteria        0
viruses         0
lead            0
nitrates        0
nitrites        0
mercury         0
perchlorate     0
radium          0
selenium        0
silver          0
uranium         0
is_safe         0
dtype: int64
```

Figure 11 The result of checking the missing data

The results show that in this data there are no missing data occurring. So, in this research do not need to be handling.

Analysis Phase

The data that used in this research is water safety that splitting into 3 sets already with ratio are Training set 60%, Validation set 20% and Testing 10% before applying to evaluate the machine learning models. In this part I will contain the training method of each model with potential setting up parameters to get high accuracy of validation set, feature selection to concentrate on fixing the overfitting or underfitting. The results of the models will show in the results and discussions.

Neural Network

The parameters of a learning algorithm to set up for training method.

Input layer = 4796

- Nodes in input layer

Hidden layer (n_h) = 100

- Nodes in Hidden layer

Output layer = 4796

- Nodes in Output layer

Alpha (Learning rate) = 0.01

Number of iterations = 500

- Activation Functions is a sigmoid function, used to apply non-linear transformation on input to map it to output.

Error Computation

- Loss = Actual_Value - Predicted_Value
- Cost = Summation (Loss)

Log loss function

- $-\text{Sum} (\text{Log} (\text{Pred}) \text{ Actual} + \text{Log} (1 - \text{Pred}) \text{ Actual}) / m$

XGBoost

The parameters of a learning algorithm to set up for training method.

Number of iterations = 500

Alpha (Learning rate) = 0.01

- L1 regularization term on weights. Increasing this value will make model more conservative.
- range: $[0, \infty]$

Depth of each tree = 4

- Maximum depth of a tree. Increasing this value will make the model more complex and more likely to overfit. 0 indicates no limit on depth. Beware that XGBoost aggressively consumes memory when training a deep tree. exact tree method requires non-zero value.
- range: $[0, \infty]$

Fraction of samples to use for each tree = 0.8

Fraction of features to use for each tree = 0.8

Decision Trees

The parameters of a learning algorithm to set up for training method.

criterion{"gini", "entropy", "log_loss"}, default="gini"

- The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “log_loss” and “entropy” both for the Shannon information gain.

max_depthint, default=None

- The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

Decision Trees and Random Forests

The parameters of a learning algorithm to set up for training method.

n_estimators : int = 500

- The number of trees in the forest.

criterion{“gini”, “entropy”, “log_loss”}, default=“gini”

- The function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “log_loss” and “entropy” both for the Shannon information gain, see Mathematical formulation. Note: This parameter is tree-specific.

max_depthint, default=None

- The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

Naïve Bayes

The parameters of a learning algorithm to set up for training method.

priorsarray-like of shape (n_classes,), default=None

- Prior probabilities of the classes. If specified, the priors are not adjusted according to the data.

var_smoothingfloat, default=1e-9

- Portion of the largest variance of all features that is added to variances for calculation stability.

Results and discussion

In these sections, I’ll explore the results after evaluating models by training with highlighted algorithms. The results have been processed and measured by accuracy from the validation set and discuss the performance of the model by using the learning curve.

Neural Network

The results of validation model show that :

Accuracy: 88.4375%

Figure 12 The accuracy of the neural network model

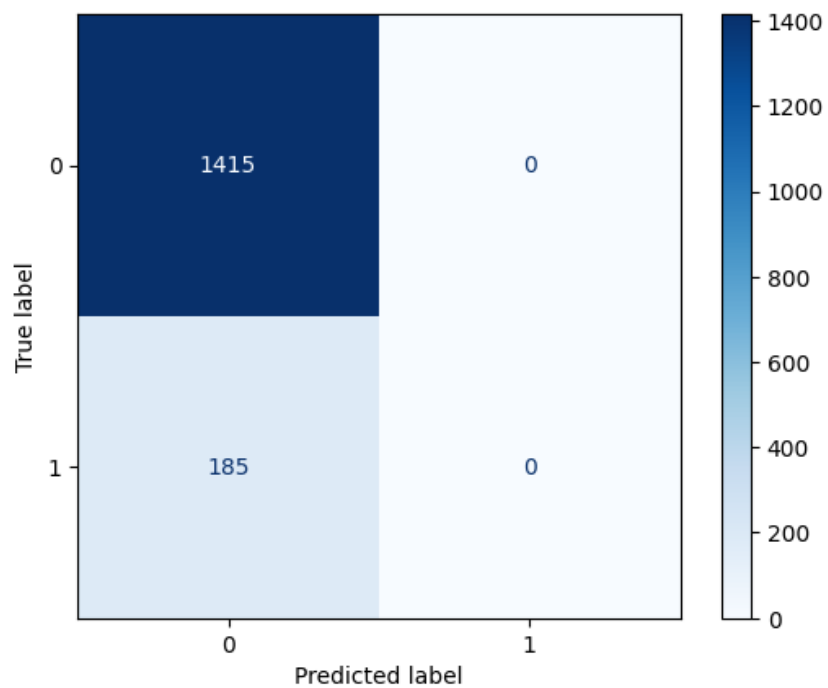


Figure 13 Confusion matrix of the neural network model

	precision	recall	f1-score	support
0.0	0.88	1.00	0.94	1415
1.0	0.00	0.00	0.00	185
accuracy			0.88	1600
macro avg	0.44	0.50	0.47	1600
weighted avg	0.78	0.88	0.83	1600

Figure 14 Classification report of the neural network model

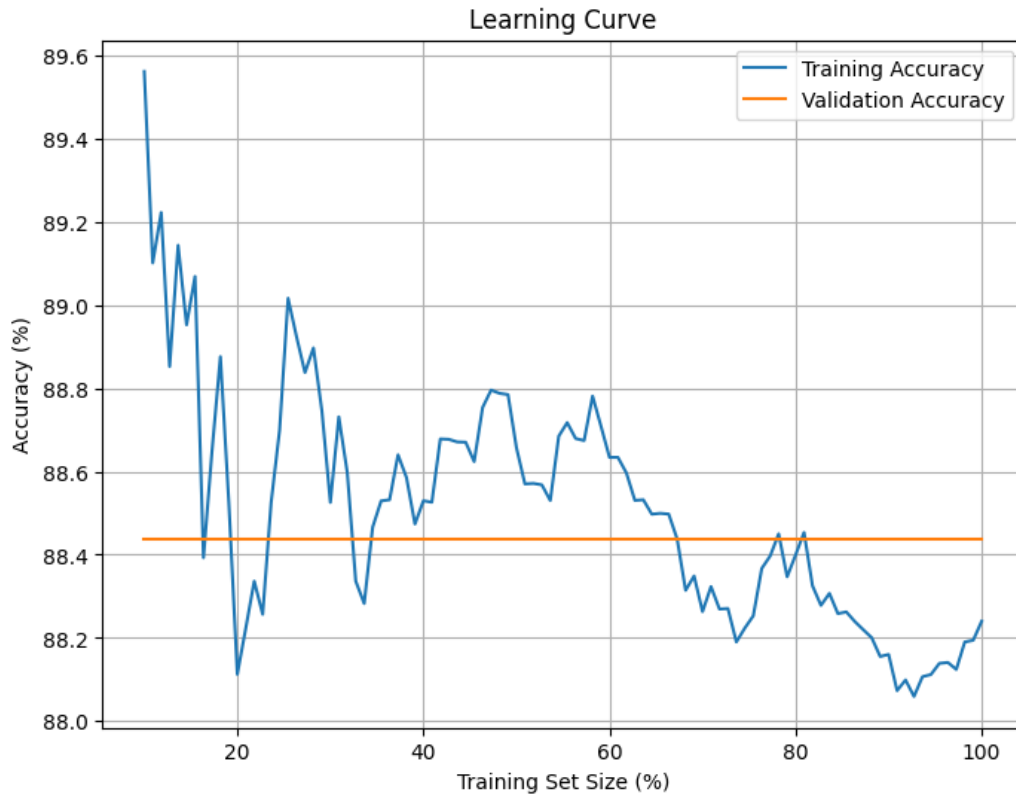


Figure 15 Learning curve on the NN model

From the confusion matrix. We focused on actual versus prediction of True Positive (TP), False Positive (FP), True Negative (TN) and False Negative (FN) as shown in figure 13.

The accuracy of the model = 88.4375%

The interpretation of the confusion matrix is that:

True Positives (TP) = 0 The model did not correctly predict any positive instances.

False Positives (FP) = 185 The model incorrectly predicted 185 instances as positive when they were actually negative.

True Negatives (TN) = 1415 The model correctly predicted 1415 instances as negative.

False Negatives (FN) = 0 The model did not miss any positive instances (because it never predicted any positives).

The model is heavily biased toward predicting only the negative class 0 or always predicting unsafety water and it's never predicted the positive class 1 or safety water, which is a sign of extreme class imbalance or a miscalibrated model. Since all actual positives were misclassified as negatives, the model might be overfitting to the negative class.

Figure 14 shows the classification report informs that the performance of the model is well with class 0 or unsafety water belongs to the confusion matrix

Figure 15 shows the model performance. The training is shown as a blue line which shows the accuracy slowly drop over time of iterations (number of iterations = 500) because of the

training fit while validation as an orange is stable over time of iterations because the model uses a single validation data set or not generalized with more data. The training and validation have a close gap between the graphs that indicate low variance but high bias because the model has hit a performance ceiling. In addition the stable validation graph may have a cause of class imbalance that the dataset has many more negative samples than positive ones so that the model might be learning to always predict 0(unsafety) or feature issue so that the features might not be informative enough to distinguish between positive and negative cases.

XGBoost

The results of validation model show that :

Accuracy: 96.50%

Figure 16 The accuracy of the XGBoost model

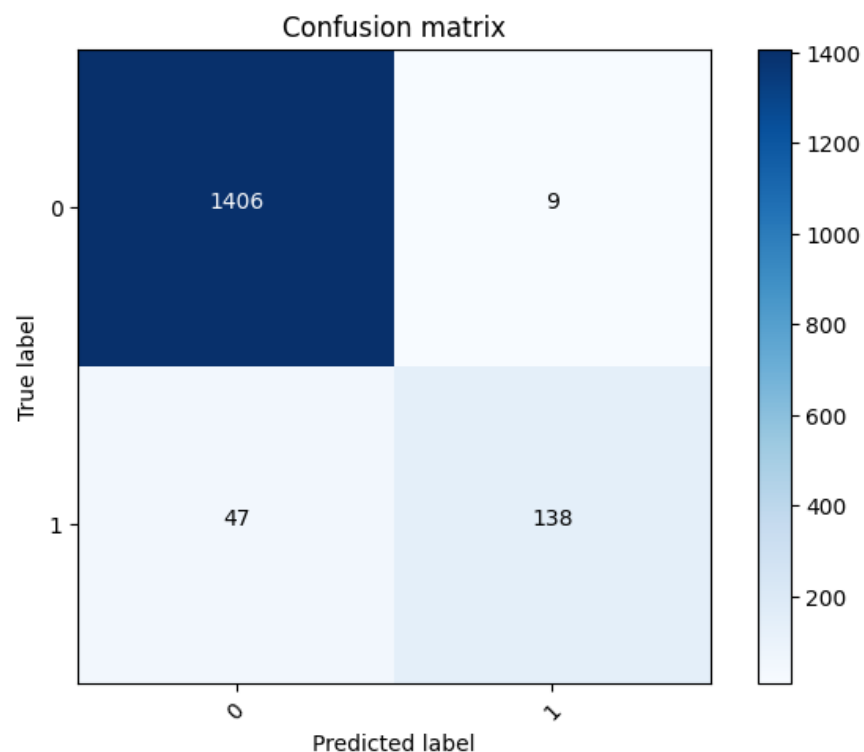


Figure 17 Confusion matrix of the XGBoost model

	precision	recall	f1-score	support
0.0	0.97	0.99	0.98	1415
1.0	0.94	0.75	0.83	185
accuracy			0.96	1600
macro avg	0.95	0.87	0.91	1600
weighted avg	0.96	0.96	0.96	1600

Figure 18 Classification report of the XGBoost model

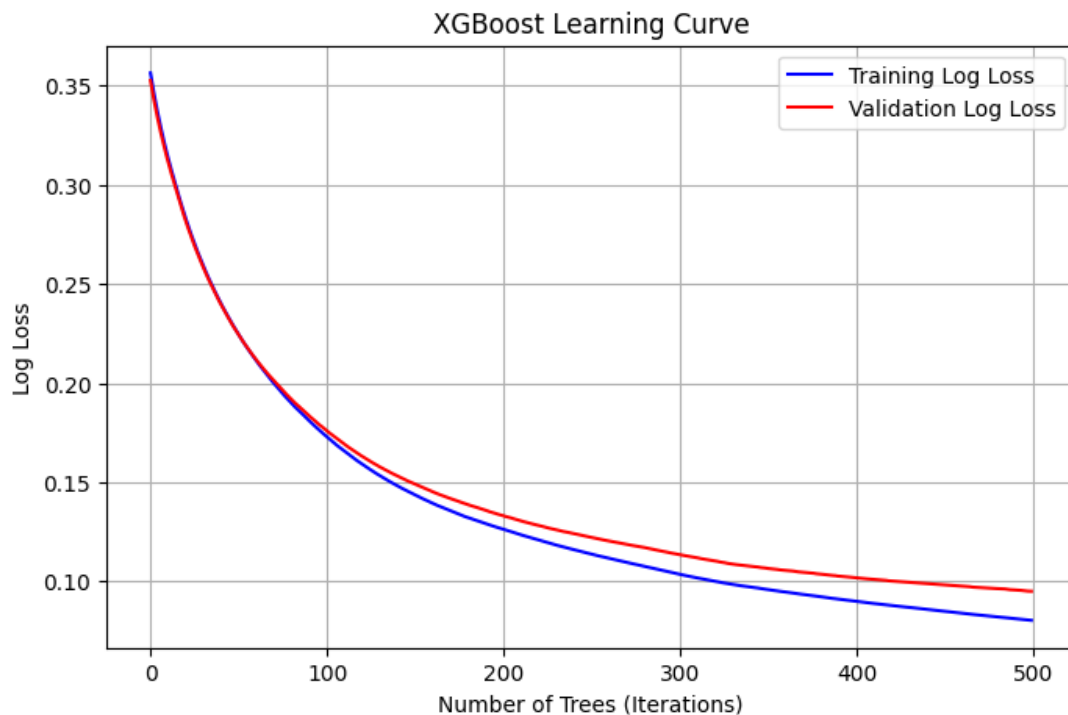


Figure 19 Learning curve on the XGBoost model

The accuracy of the model = 96.50%

The interpretation of the confusion matrix is that:

True Negatives (TN) = 1406 The model correctly predicted class 0 when the actual class was 0.

False Positives (FP) = 47 The model incorrectly predicted class 0 when the actual class was 1.

False Negatives (FN) = 9 The model incorrectly predicted class 1 when the actual class was 0.

True Positives (TP) = 138 The model correctly predicted class 1 when the actual class was 1.

The model seems to perform very well, especially in identifying class 0 or unsafety water and it also has a very high accuracy. The performance for class 1 or safety water is good but less impressive than for class 0. The matrix suggests a significant class imbalance in the dataset. This makes high overall accuracy easier to achieve, but the relatively higher number of False Negatives (47) for the minority class 1 might be a concern.

Figure 18 shows the classification report informs that the performance of the model is well with both of class 0 and class 1 with high precision, recall and F1-score.

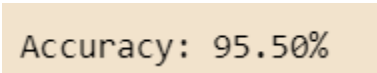
Figure 19 shows the learning curve of the model with log loss function(a measure of error, where lower is better). Both the training (blue line) and validation loss (orange line) decrease significantly as the number of trees increases from 0 up to around 150-200 trees. After this point, the rate of decrease slows down considerably, and the curves start to flatten out, indicating the model's performance is stabilizing. The training loss consistently decreases and ends up slightly lower than the validation loss. This is expected, as the model is directly optimized on the training data. The validation loss also decreases steadily and appears to be leveling off, suggesting the model is generalizing well to unseen data.

The gap between the training loss and validation loss is relatively small and doesn't appear to be widening significantly as more trees are added. This pattern is generally indicative of a good fit. The model is learning effectively (loss is decreasing), and it's generalizing well to the validation set without strong signs of overfitting

Decision Trees

From the documentation of Scikitlearn, the decision Tree has a criterion parameter function to measure the quality of a split. Supported criteria are “gini” for the Gini impurity and “entropy”. In this model entropy compared gini criteria functions to measure the differentiation of accuracy.

Decision tree with entropy as a criteria



Accuracy: 95.50%

Figure 20 The accuracy of the Decision Trees model with entropy

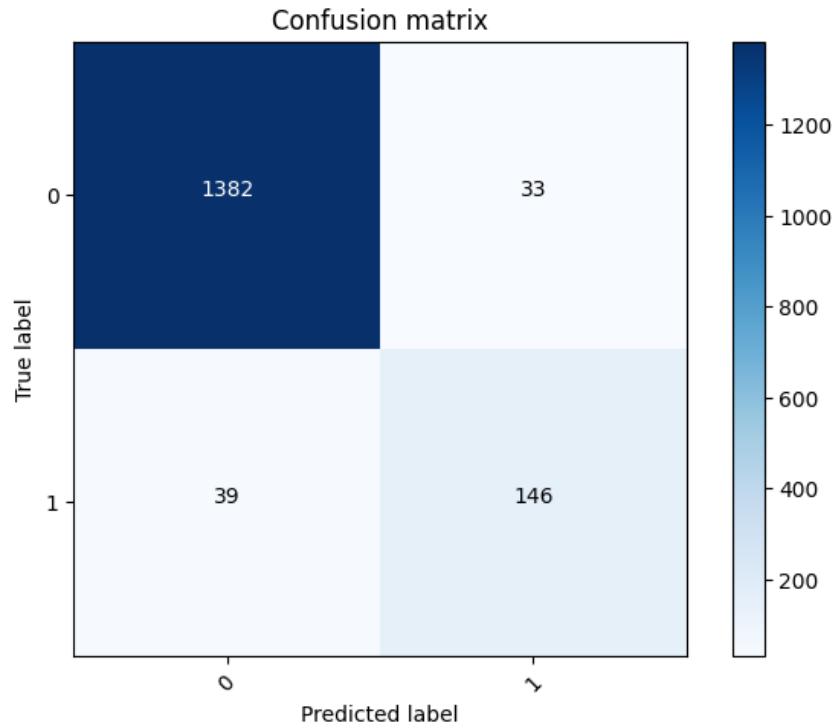


Figure 21 Confusion matrix of the Decision Trees model with entropy

	precision	recall	f1-score	support
0.0	0.98	0.98	0.98	1415
1.0	0.83	0.82	0.82	185
accuracy			0.96	1600
macro avg	0.90	0.90	0.90	1600
weighted avg	0.96	0.96	0.96	1600

Figure 22 Classification report of the Decision Trees model with entropy

The accuracy of the model = 95.50%

The interpretation of the confusion matrix is that:

True Negatives (TN) = 1382 The model correctly predicted class 0 when the actual class was 0.

False Positives (FP) = 39 The model incorrectly predicted class 0 when the actual class was 1.

False Negatives (FN) = 33 The model incorrectly predicted class 1 when the actual class was 0.

True Positives (TP) = 146 The model correctly predicted class 1 when the actual class was 1.

Decision Tree model with entropy achieves high overall accuracy, driven largely by its strong performance on the majority class 0. Its ability to correctly identify the minority class 1 (Recall).

Figure 22 shows the classification report informs that the performance of the model is well with both of class 0 and class 1 with high precision, recall and F1-score.

Decision tree with Gini as a criteria

Accuracy: 95.75%

Figure 23 The accuracy of the Decision Trees model with Gini

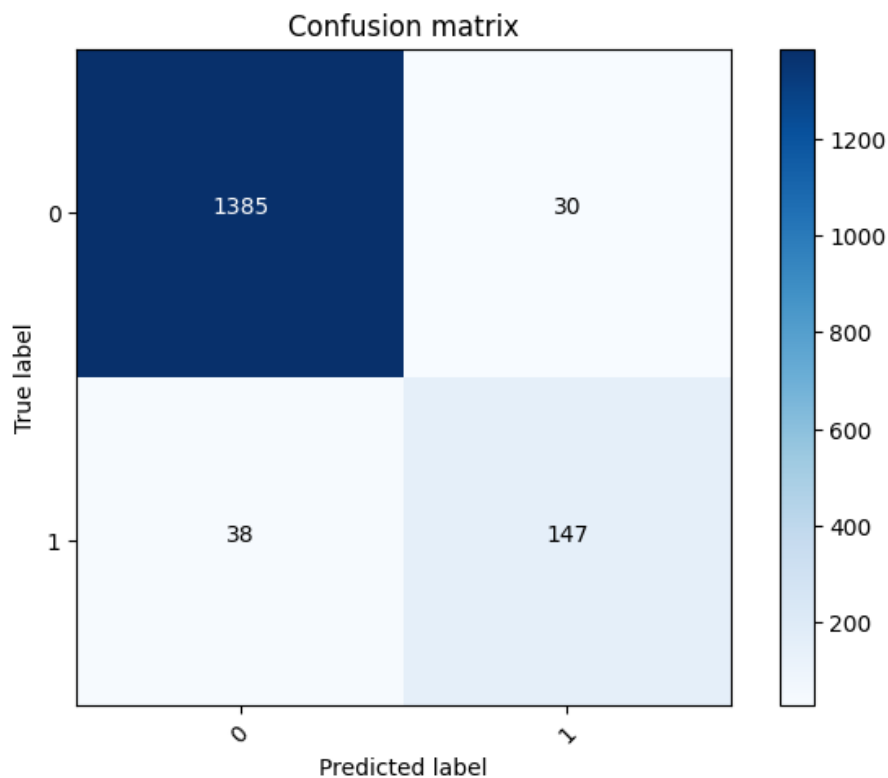


Figure 24 Confusion matrix of the Decision Trees model with Gini

	precision	recall	f1-score	support
0.0	0.98	0.98	0.98	1415
1.0	0.84	0.82	0.83	185
accuracy			0.96	1600
macro avg	0.91	0.90	0.91	1600
weighted avg	0.96	0.96	0.96	1600

Figure 25 Classification report of the Decision Trees model with Gini

The accuracy of the model = 95.75%

The interpretation of the confusion matrix is that:

True Negatives (TN) = 1385 The model correctly predicted class 0 when the actual class was 0.

False Positives (FP) = 38 The model incorrectly predicted class 0 when the actual class was 1.

False Negatives (FN) = 30 The model incorrectly predicted class 1 when the actual class was 0.

True Positives (TP) = 147 The model correctly predicted class 1 when the actual class was 1.

In summary, this Decision Tree model using the Gini criterion performs slightly better across most metrics (Accuracy, Precision, Recall, Specificity) compared to the Decision Tree using entropy shown in the previous image. It has high accuracy, particularly for the majority class 0, and achieves a good balance between precision and recall for the minority class 1.

Figure 25 shows the classification report informs that the performance of the model is well with both of class 0 and class 1 with high precision, recall and F1-score.

The Gini criteria gave a more accuracy than entropy so in the decision trees model I likely to use Gini index as a based of the model and build the decision tree for the algorithm decision making.

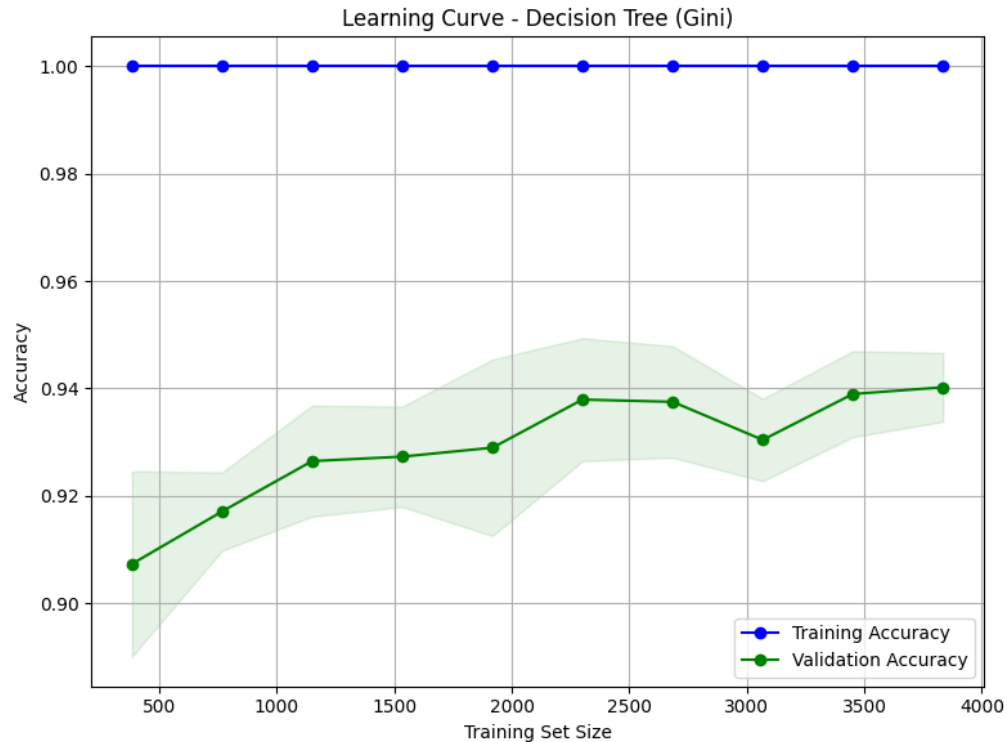


Figure 26 Learning curve on a Decision Tree model

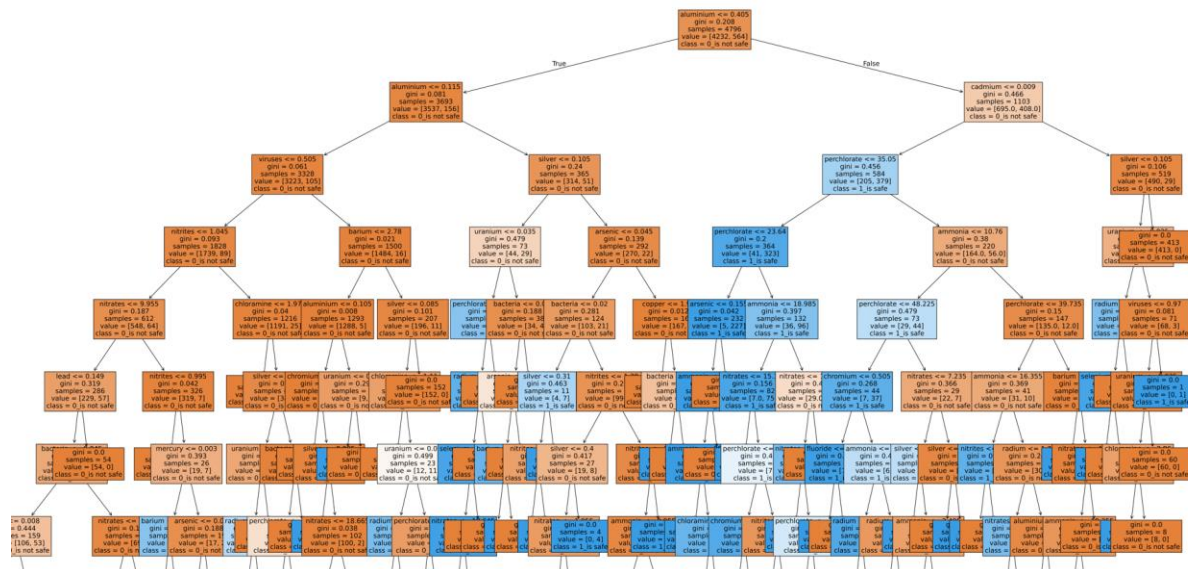


Figure 27 Example of a Decision Trees architecture by using Gini index

From figure 26 shows the learning curve of training and validation line. The training accuracy (Blue Line) starts at 1.00 for a small training set size and remains consistently. This indicates that the model perfectly fits the training data across all training set sizes. The validation accuracy (Green Line) starts at a lower value for a small training set size and generally

increases as the training set size grows. There is a significant gap between the training accuracy and the validation accuracy. This indicates a high variance problem or overfitting. The model learns the training data extremely well but does not generalize as effectively to unseen data.

The learning curve suggests that the Decision Tree model is overfitting the training data. While it achieves perfect accuracy on the training set, its performance on the validation set is considerably lower and plateaus.

As show in figure 27 The architecture of the decision tree after proceeding the model for decision making that high complexity without limited to the deep of each branch.

Decision Trees and Random Forests

Accuracy: 95.38%

Figure 28 The accuracy of the Decision Trees and Random Forests model

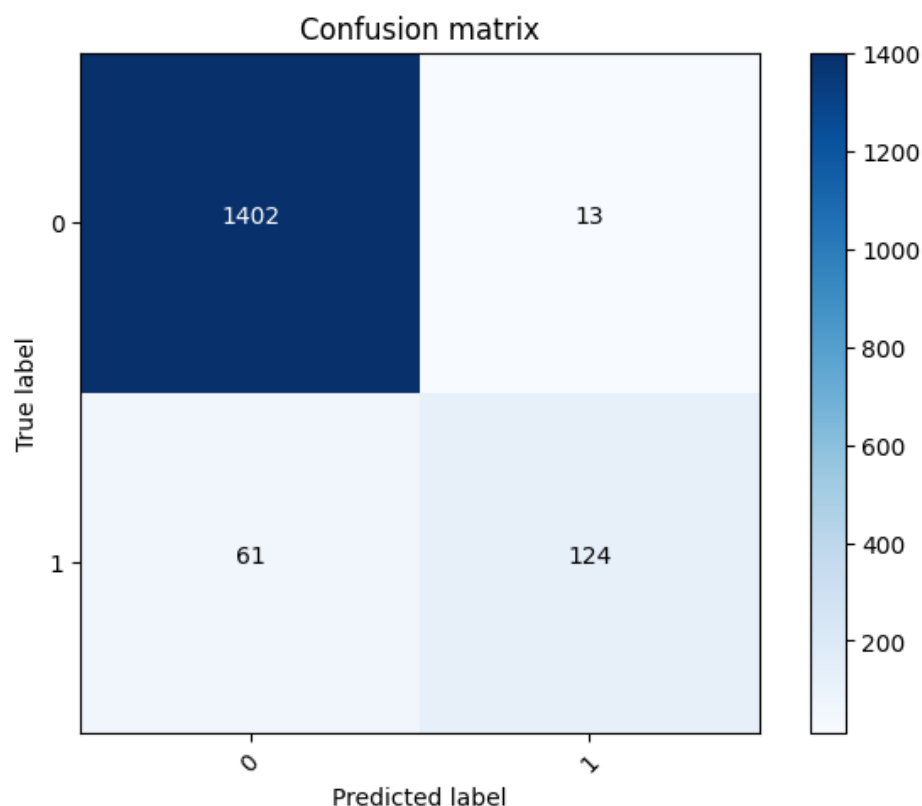


Figure 29 Confusion matrix of the Decision Trees and Random Forests model

	precision	recall	f1-score	support
0.0	0.96	0.99	0.97	1415
1.0	0.91	0.67	0.77	185
accuracy			0.95	1600
macro avg	0.93	0.83	0.87	1600
weighted avg	0.95	0.95	0.95	1600

Figure 30 Classification report of the Decision Trees and Random Forests model

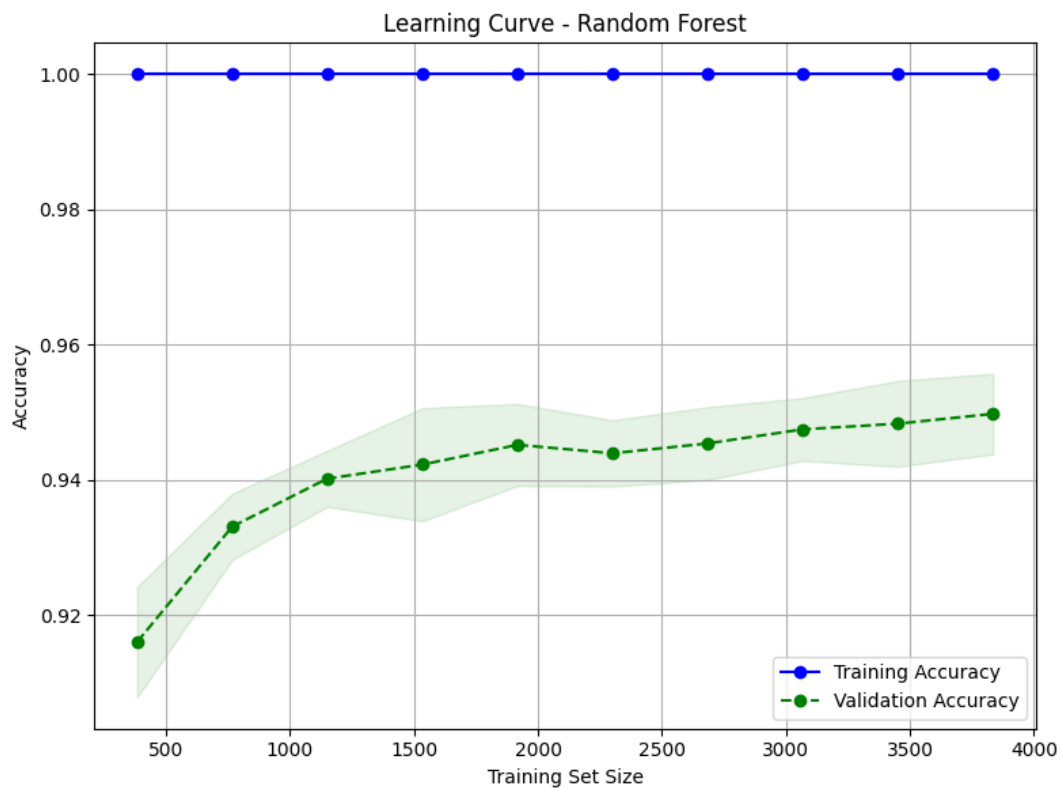


Figure 31 Learning curve on the Decision Trees and Random Forests model

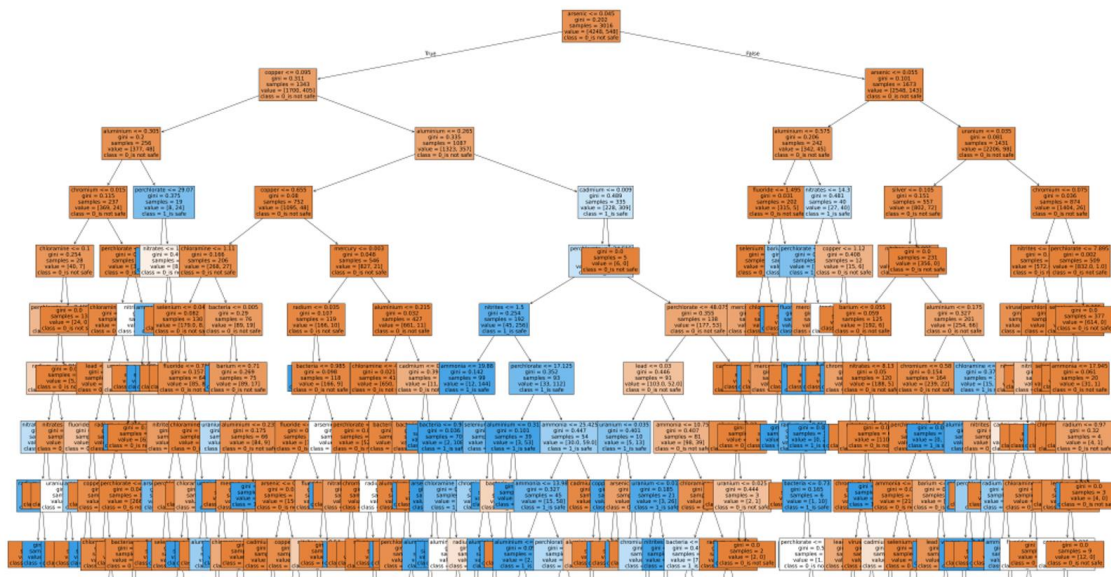


Figure 32 Example of the Decision Trees and Random Forests architecture

The accuracy of the model = 95.38%

The interpretation of the confusion matrix is that:

True Negatives (TN) = 1402 instances were correctly predicted as belonging to class 0.

False Negatives (FN) = 13 instances were incorrectly predicted as belonging to class 1, but they actually belong to class 0.

False Positives (FP) = 61 instances were incorrectly predicted as belonging to class 0, but they actually belong to class 1.

True Positives (TP) = 124 instances were correctly predicted as belonging to class 1.

The model performs very well at correctly identifying instances of class 0. The model shows a strong ability to classify instances of class 0 but has some room for improvement in correctly classifying instances of class 1.

Figure 30 shows the classification report information that the model performance well with both of class 0 and class 1 with high precision, recall and F1-score.

Figure 31 Show the learning curve of the model that the training accuracy (Blue Line) starts at 1 and remains consistently at 1.00. This indicates that the model perfectly fits the training data across all training set sizes. The validation accuracy (Green Line) starts at a lower value for a small training set size. The validation accuracy generally improves, showing an upward trend. There is a persistent gap between the training accuracy and the validation accuracy. This suggests that the model is overfitting the training data. While it learns the training data extremely well, this perfect learning doesn't generalize perfectly to unseen data (the

validation set) so suggests that adding significantly more training data might not lead to substantial improvements in the model's generalization performance.

The learning curve indicates that the Random Forest model is overfitting the training data, achieving perfect accuracy on the training set but a lower and plateauing accuracy on the validation set. Further increasing the training data size is unlikely to significantly improve generalization performance. Addressing the overfitting issue is the next step for improving the model.

Figure 32 shows the model of trees for decision making with decision trees and random forests algorithm that high complexity.

Naïve Bayes

Accuracy: 84.12%

Figure 33 The accuracy of the Naïve Bayes model

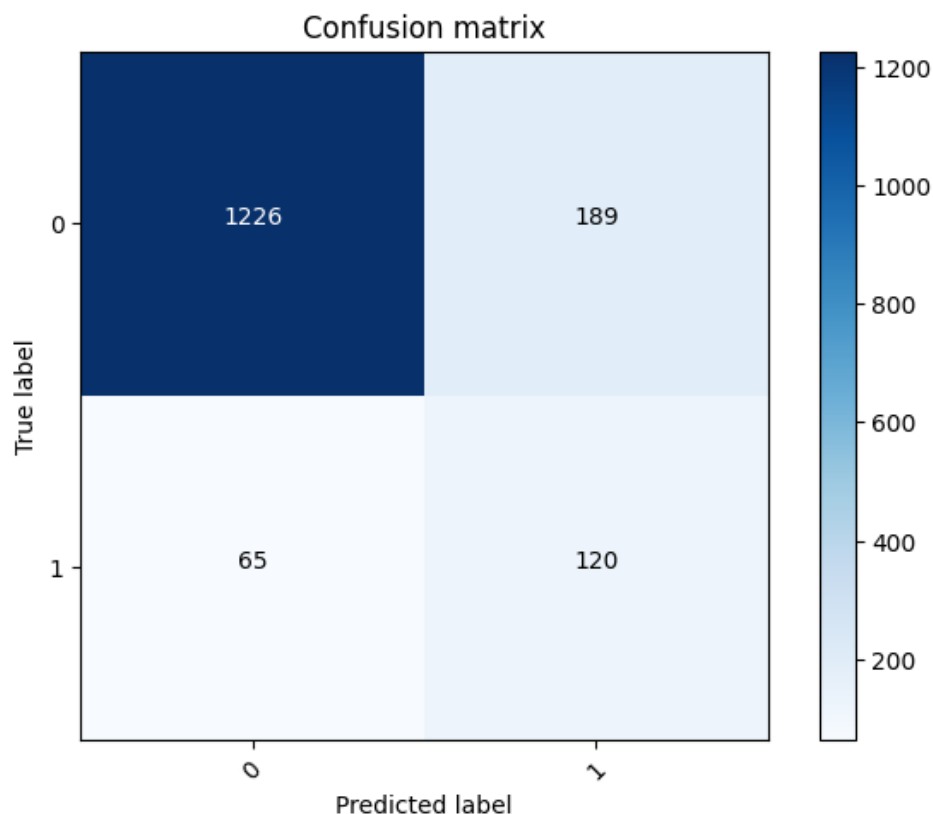


Figure 34 Confusion matrix of the Naïve Bayes model

	precision	recall	f1-score	support
0.0	0.95	0.87	0.91	1415
1.0	0.39	0.65	0.49	185
accuracy			0.84	1600
macro avg	0.67	0.76	0.70	1600
weighted avg	0.88	0.84	0.86	1600

Figure 35 Classification report of the Naïve Bayes model

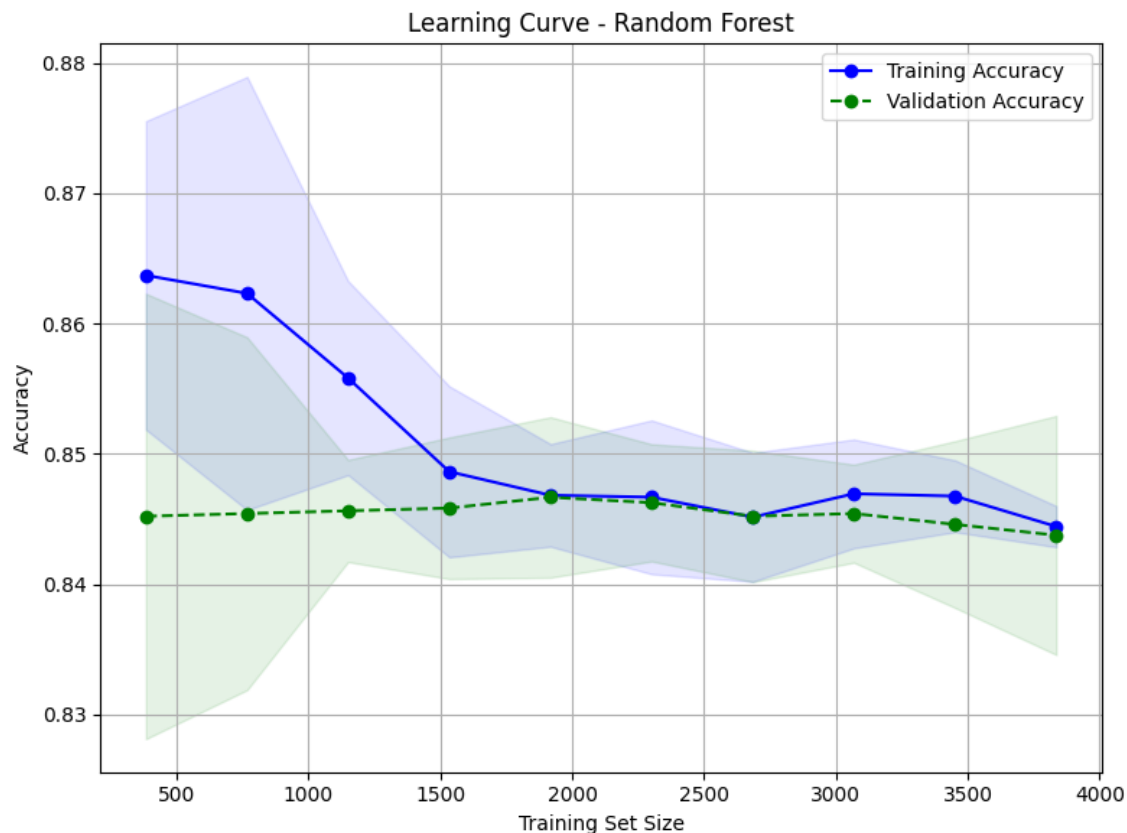


Figure 36 Learning curve on the Naïve Bayes model

The accuracy of the model = 84.12%

The interpretation of the confusion matrix is that:

True Negatives (TN) = 1226 instances were correctly predicted as belonging to class 0.

False Negatives (FN) = 189 instances were incorrectly predicted as belonging to class 1, but they actually belong to class 0.

False Positives (FP) = 65 instances were incorrectly predicted as belonging to class 0, but they actually belong to class 1.

True Positives (TP) = 120 instances were correctly predicted as belonging to class 1.

The Naïve Bayes model shows a good ability to classify instances of class 0 but has a noticeable number of misclassifications for both classes, with a higher tendency to misclassify class 0 as class 1.

Figure 35 shows the classification report informs that the performance of the model is well with both of class 0 and class 1 with high precision, recall and F1-score.

Figure 36 shows that the training accuracy (Blue Line) starts at a relatively high value for a small training set size and generally decreases as the training set size increases. The validation accuracy (Green Line) starts at a lower value for a small training set size and shows a slight upward trend initially. There is a gap between the training accuracy and the validation accuracy throughout initial of training sizes. The training accuracy consistently outperforms the validation accuracy. This suggests the presence of some overfitting, although it's not as extreme as in a scenario where training accuracy is near perfect.

The Naïve Bayes model indicates a moderate degree of overfitting and suggests that increasing the training data size alone is unlikely to significantly boost performance. Further improvements might require exploring model tuning.

Model Selection

Model selection is an essential phase in the development of powerful and precise predictive models in the field of machine learning. Model selection is the process of deciding which algorithm and model architecture is best suited for a particular task or dataset. It entails contrasting various models, assessing their efficacy, and choosing the one that most effectively addresses the issue at hand.

This method is to select the most accurate model on the validation set. The result shows that the XGBoost model gave the highest accuracy (lowest validation error), which is 96.50%. The results are summarized on the table below.

Table 3 Summary of the model's interpretation

Models	Validation accuracy	Confusion Matrix
Neural Network	88.4375%	TP = 0, FP = 185, TN = 1415, FN = 0
XGBoost	96.50%	TP = 138, FP = 47, TN = 1406, FN = 9
Decision Trees with entropy	95.50%	TP = 146, FP = 39, TN = 1382, FN = 33
Decision Trees with Gini	95.75%	TP = 147, FP = 38, TN = 1385, FN = 30
Decision Trees and Random Forests	95.38%	TP = 124, FP = 61, TN = 1402, FN = 13
Naïve Bayes	84.12%	TP = 120, FP = 65, TN = 1226, FN = 189

The test set will be selected to perform this XGBoost model and then process to get the results. The result shows in the next step of model prediction.

Model Prediction

Perform the XGBoost algorithm that has been analyzed on the test set to get the result.

```
Shape of new data (single sample): (1, 20)
Prediction for the new sample: [1]
The true prediction is [1]
Accuracy: 81.94%
Confusion Matrix of Validation:
[[1299  116]
 [ 173   12]]
```

Figure 37 Testing the XGBoost algorithm

Figure 37 shows that the XGBoost model test with input of x variable which has 20 features (single sample) with actual class label (y variable) = 1. The set of input is followed by this [2.72,26.81,0.01,0.28,0.005,5.37,0.8,1.62,0.96,0,0,0.138,5.66,1.93,0,22.32,0.2,0.01,0.01,0.06]. The model prediction for the sample shows predicted class = 1 which is correct predicted class. The classification accuracy of the test set is 81.94%

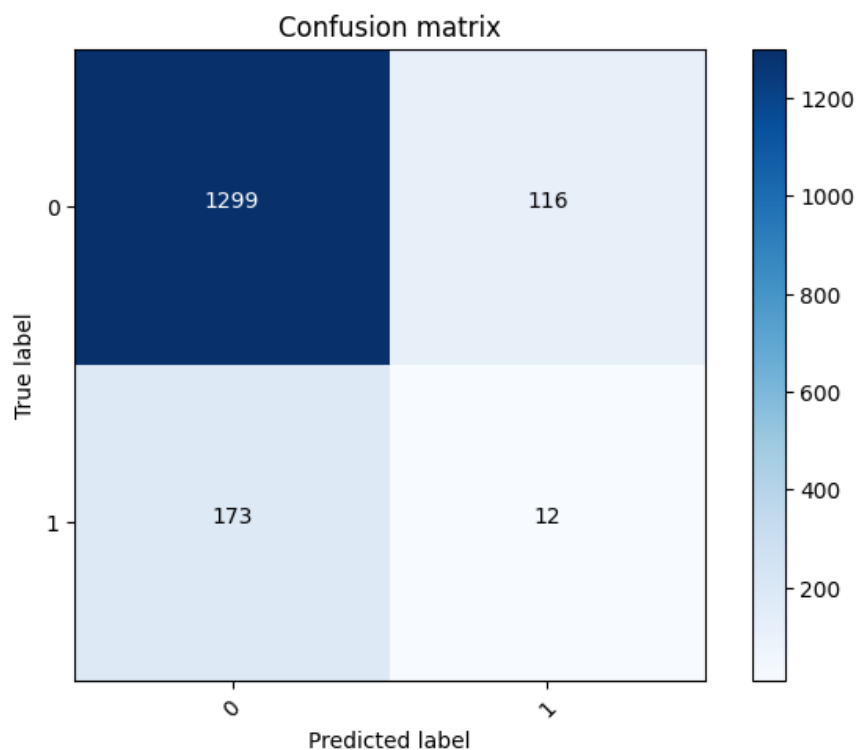


Figure 38 Confusion matrix of the XGBoost prediction on test set

The Confusion matrix interprets that

True Positives (TP) = 12 instances were correctly predicted as belonging to class 1.

False Positives (FP) = 173 instances were incorrectly predicted as belonging to class 0, but they actually belong to class 1.

True Negatives (TN) = 1299 instances were correctly predicted as belonging to class 0.

False Negatives (FN) = 116 instances were incorrectly predicted as belonging to class 1, but they actually belong to class 0.

The model shows a strong ability to correctly classify instances of class 0 but contrast to the number of True Positives is very low, suggesting that the model has poor performance in identifying instances of class 1.

Conclusions and Discussions

This research dedicated to apply proposed machine learning framework to predict the efficiency and effectiveness of the water safety predictions. The proposed machine learning framework conclude of five models; Neural Network, XGBoost, Decision Trees, Decision Trees and Random Forests. The performance of those models have been validated on the assigned dataset consisting of 4,796 samples with 20 features. The obtained results clarified on the validation set with good classification with 88.4375%, 95.75%, 95.38% and 84.12% consequently. However, the model selection is choose to be XGBoost model which has the most accuracy to test on the test set. The results of classification with the 81.94% accuracy. However, the model test on the test set gives the lower accuracy than the validation set. The XGBoost has a better performance compare to the others algorithm because decision trees are prone to overfitting, ensemble methods, like boosting, can often be used to create more robust models. Boosting combines multiple individual weak trees that is, models that perform slightly better than random chance, to form a strong learner. Each weak learner is trained sequentially to correct the errors made by the previous models. After hundreds of iterations, weak learners are converted into strong learners. And the additional of features in the algorithms such as parallel and distributed computing, cache-aware prefetching algorithm and built in regularization.

References

- Awan, A. A. (n.d.). *What are neural networks?* DataCamp. Retrieved April 17, 2025, from <https://www.datacamp.com/blog/what-are-neural-networks>
- Bansal, S. (n.d.). *A very comprehensive tutorial: NN + CNN* [Kaggle notebook]. Kaggle. Retrieved April 17, 2025, from <https://www.kaggle.com/code/shivamb/a-very-comprehensive-tutorial-nn-cnn>
- Chen, T., & Guestrin, C. (n.d.). *XGBoost parameters*. XGBoost Documentation. Retrieved April 17, 2025, from <https://xgboost.readthedocs.io/en/stable/parameter.html>
- Galarnyk, M. (n.d.). *Decision trees visualization* [Jupyter notebook]. GitHub. Retrieved April 17, 2025, from https://github.com/mGalarnyk/Python_Tutorials/blob/master/Sklearn/CART/Visualization/DecisionTreesVisualization.ipynb
- GeeksforGeeks. (2025, February 2). *XGBoost*. Retrieved April 17, 2025, from <https://www.geeksforgeeks.org/xgboost/>
- GeeksforGeeks. (2025, January 16). *Decision tree*. Retrieved April 17, 2025, from <https://www.geeksforgeeks.org/decision-tree/>
- GeeksforGeeks. (2025, January 16). *Random forest algorithm in machine learning*. Retrieved April 17, 2025, from <https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/>
- Goel, A. (2021, May 16). *Binary classification using decision-tree model*. Medium. Retrieved April 17, 2025, from <https://adityagoel123.medium.com/binary-classification-using-decision-tree-model-c4e20c8a5afb>
- Jordan, M. I., & Mitchell, T. M. (2015). Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245), 255–260. <https://doi.org/10.1126/science.aaa8415>
- Matillion. (n.d.). *Data exploration: What it is, techniques, and examples*. Retrieved April 17, 2025, from <https://www.matillion.com/learn/blog/data-exploration>
- ScholarHat. (2023, June 1). *Model selection for machine learning*. Retrieved April 17, 2025, from <https://www.scholarhat.com/tutorial/machinelearning/model-selection-for-machine-learning>
- Scikit-learn developers. (n.d.). *sklearn.ensemble.RandomForestClassifier*. Retrieved April 17, 2025, from <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>
- Scikit-learn developers. (n.d.). *sklearn.metrics.classification_report*. Retrieved April 17, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

- Scikit-learn developers. (n.d.). *sklearn.naive_bayes.GaussianNB*. Retrieved April 17, 2025, from https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html
- Scikit-learn developers. (n.d.). *sklearn.tree.DecisionTreeClassifier*. Retrieved April 17, 2025, from <https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html>
- UN-Water. (n.d.). Water and disasters. Retrieved April 17, 2025, from <https://www.unwater.org/water-facts/water-and-disasters>
- wrecked22. (n.d.). *Basic binary classification using XGBoost* [Kaggle notebook]. Kaggle. Retrieved April 17, 2025, from <https://www.kaggle.com/code/wrecked22/basic-binary-classification-using-xgboost>