Predicting Water Potability with Machine Learning and Deep Learning Techniques

Hannah Johnston
School of Information Technology
Carleton University
Ottawa, Canada
hannahjohnston@cmail.carleton.ca

Mahitha Sangem
Systems and Computer Engineering, Data Science
Carleton University
Ottawa, Canada
mahithasangem@cmail.carleton.ca

Abstract— **Safe drinking water is important for everyone. We built several binary classifiers to recognize water potability (determine whether water is potable or not). We compared the performance of traditional machine learning and deep learning techniques for the prediction of water potability. We focused on F1-score, due to the imbalance in the data and the high cost of both false negatives and false positives. Imputation with trimmed mean and ADASYN yielded improved results for most of our models. Deep learning models slightly outperformed traditional machine learning, particularly CNNs with ADASYN and ensemble methods, but the results were not sufficiently accurate for real-world predictions. The limited size and synthetic composition and size of the dataset may have limited model performance. Further research is needed to improve and solidify performance through modified architectures (including hybrid CNN models), additional real-world data, and repeated testing.**

Keywords—Artificial intelligence, Machine learning, Boosting, Deep learning, Ensemble learning, Random forests, Neural networks, Convolutional neural networks, Recurrent neural networks.

I. Introduction

In predicting water potability, our goal is safe drinking water for everyone. On a national, regional, and local level, this is significant as a health and development issue. Investment in water supply and sanitation can also produce a new economic advantage as access to potable drinking water reduces negative health effects and medical expenses more than it costs to implement [1].

Our goal was to build a binary classifier to recognize water potability (water is potable or not). We explored traditional machine learning models, ensemble, and deep learning methods.

A. *Dataset*

We used a dataset from Kaggle [2], that contains 3276 different water samples (rows). It includes 10 attributes, with nine water quality metrics: pH, hardness, solids, chloramines, sulfate, conductivity, organic carbon, trihalomethanes and turbidity. The last column is the target variable, potability, where 1 means the water is potable and 0 means it is non-potable. Full details are available in the appendix.

B. *Prior work*

There have been many attempts to classify water quality, using a variety of machine learning and deep learning techniques [3–6]. Decision Trees, SVM, KNN, Naïve Bayes, MLP, LSTM, and GRU) have been applied to different datasets, with different features and sample collection periods. A subset of the research has been conducted using the same Kaggle dataset, so we have focused on those in this paper. Chafloque et al. (2021) implemented a dense, sequential neural network with five hidden layers (with 32, 64, 32, 16, and 8 neurons respectively) [7]. They used ReLU for the first 6 layers and the sigmoid activation function on the output layer to achieve validation results with approximately 69% accuracy. Yusuf et al. (2022) used several machine learning models to classify safe water quality and achieved a top accuracy of 83.78% with Random Forest [8].

Haq et al. further built on these results, allegedly achieving 97.23% accuracy on our same dataset with Decision Trees and k-fold cross-validation with 25% test size [9]. We were unable to replicate or even approach these results using similar models and were consequently skeptical of their validity. We believe data leakage may have been an issue as they illustrated performing a train-test split following the rest of their dataset preprocessing. The authors did not respond to our request for more information.

C. *Relevant metrics*

While some of the prior work considered only accuracy [7], F1-score is a more useful metric for this data, because of its imbalance (more 0s than 1s). Metrics other than accuracy (e.g., precision, recall, f1-score) are valuable due to the higher cost and impact of both false negatives and false positives: either potable water wastage or negative health impacts of consuming non-potable water.

II. Methods

A. *Analysis & Pre-processing*

1) *Feature analysis*

We plotted histograms, univariate analysis, and box plots for all features. We also calculated the count, mean, standard deviation, minimum, maximum, and 25%, 50%, and 75% values.

We also plotted feature correlations and found no high intercorrelation between features. All features had a similar, but low, correlation with the target variable “potability”, so we kept and considered all nine features. It appeared difficult to achieve

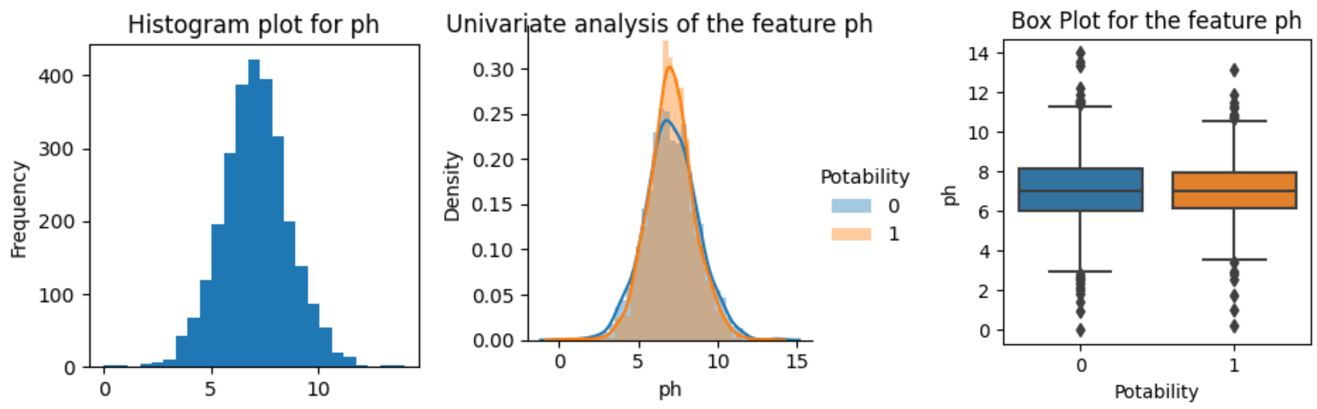

Fig. 1. Sample histogram, univariate analysis, and box plots for the pH feature

high scores because of the negligible correlation of the features with the target variable.

2) Pre-processing

To avoid any potential data leakage issues, we first split data into train/validate/test to avoid data leakage from scaling and upsampling. We noted that all of the 0s and 1s were together in the dataset, so we used shuffle to avoid issues when splitting. We used stratify while splitting the data to get the same proportions of 0s and 1s in all three sets (train, validation, and test). We used standard scaling to bring feature values to a fixed range. We noticed class imbalance (0-1999 and 1-1278), which we addressed with upsampling using Adaptive Synthetic Sampling (ADASYN) [10].

We plotted Kernel Density Estimation (KDE) for every feature to observe distribution with target Potability. Due to high variation and outliers, we initially dropped null values. However, we later performed imputation using the trimmed mean approach. Trimmed mean is a method of averaging that removes a small designated percentage of the largest and smallest values prior to calculating the mean (to avoid bias in mean due to outliers) [11]. This led to slightly better overall results across many (but not all) of the models.

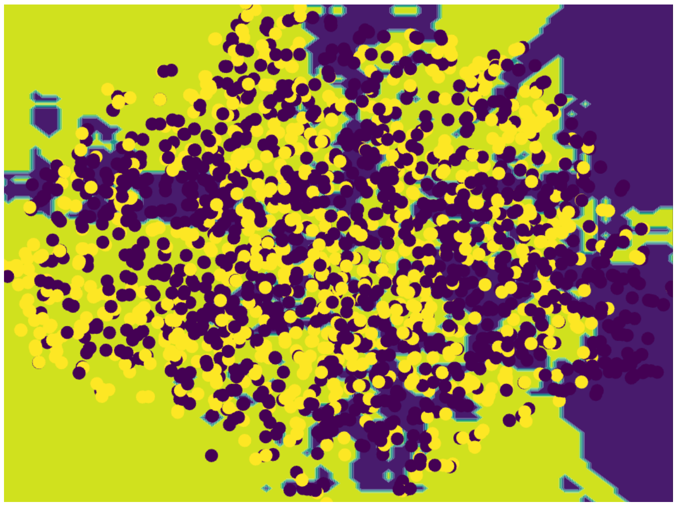

Fig. 2. Decision Boundary after dimensionality reduction (using t-SNE) for Voting Hard

B. Model Building

1) Initial Classifiers

We initially implemented K-Nearest Neighbours (KNN), Support Vector Machine (SVM), Decision Tree (DT), a basic Multilayer Perceptron (MLP), Logistic Regression and Naive Bayes algorithms.

We performed hyperparameter tuning using Bayes search for each of the above classifiers. We followed this by hyperparameter tuning with ADASYN upsampling using pipeline (to handle the imbalance) on validation data. We then fit the best estimator with training data.

To further improve the performance, we chose five of the top-performing classifiers to use as base learner models in ensemble techniques with stacking and voting. We used the following implementations: boosting and bagging classifiers like XGBoost (eXtreme Gradient Boosting), AdaBoost, Gradient Boost and Random Forest classifiers as meta classifiers in each stacking ensemble implementation. We implemented soft and hard voting.

2) Additional Deep Learning Classifiers

We built several additional deep learning models, including Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), and additional Deep Neural Network (DNN), MLP models that we refer to as (DNN) in the project to distinguish them from our earlier MLP attempts.

For each of CNN, RNN, and DNN, we ran grid search with cross validation of 3, 50 epochs, f1 scoring, and tested both with and without ADASYN. Once we identified the best parameter settings, we reran the best estimators for 100 epochs.

Because precision, recall, and f1-score were not built-in functions for all the models we used, we defined them manually. We implemented early stopping on validation f1 and learning rate reduction. When the validation loss plateaus for more than 5 epochs, we reduce the learning rate by 0.5.

We experimented with batch size and found that batch sizes of 8 generally performed better. We defaulted to this number for CNN and RNN, but as DNN was on occasion performing better at 16, left it in the grid search as a parameter.

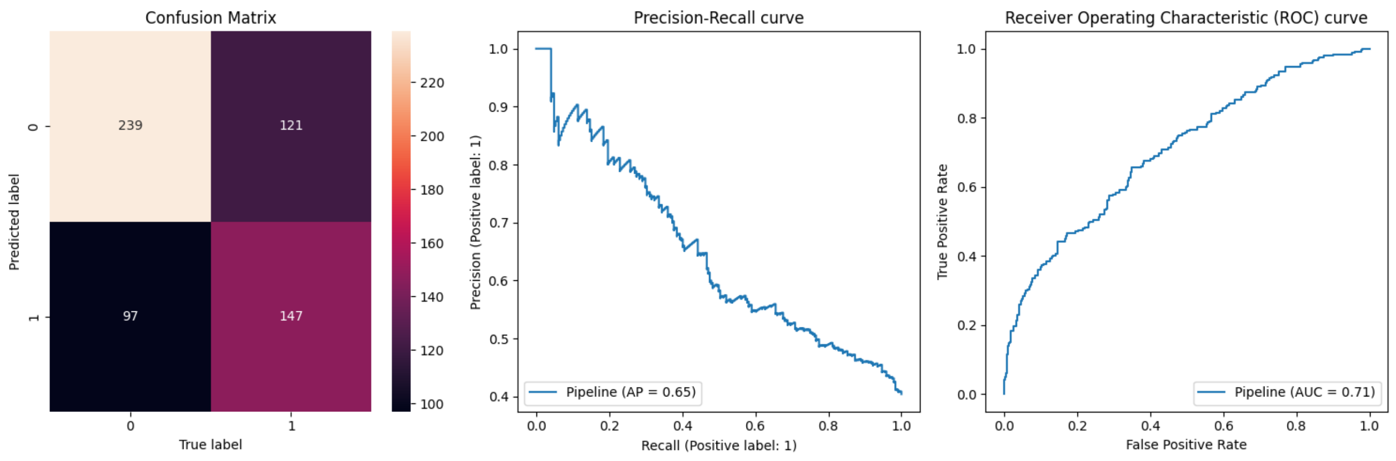

Fig. 3. Sample Confusion matrix and Precision-Recall, Receiver Operating Characteristic (ROC) for MLP with ADASYN and hyperparameter tuning

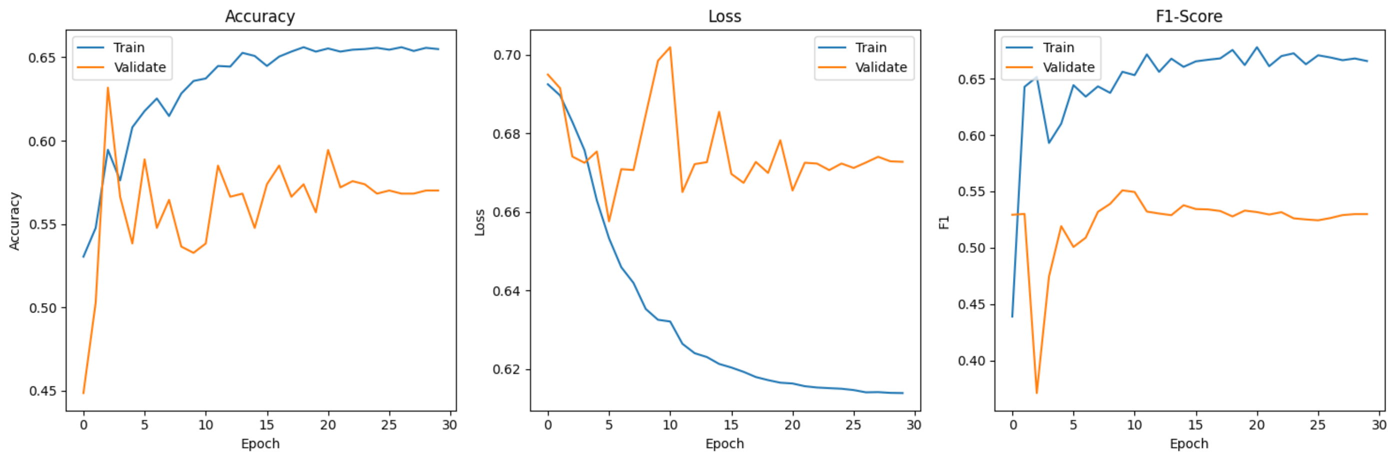

Fig. 4. Sample Accuracy, Loss, and F1-score for CNN with ADASYN and hyperparameter tuning (with early stopping and learning rate reduction)

a) Convolutional Neural Network (CNN)

We defined the input shape of the CNN as (3, 3, 1) so it could be used with Conv2D. We reshaped the training and validation data to fit this input shape. We created a Sequential model with two 2D convolutional layers with kernel sizes of 2 and 3 respectively. We used GridSearchCV to select the number of filters per layer, the amount of dropout. The output of the second convolutional layer was then flattened and passed through a Dense layer with a variable number of units and a ReLU activation function. We added another Dropout layer before the final, single-neuron output layer with a sigmoid activation function. We compiled the model with binary cross-entropy loss and the Adam optimizer.

b) Recurrent Neural Network (RNN)

We defined a RNN model with two SimpleRNN layers. We used grid search to determine the number of units in the first and second layers. Finally, we implemented a Dense output layer with a single unit and a sigmoid activation function.

c) Deep Neural Network (DNN)

We implemented a DNN with a variable number of layers, neurons, and batch size to tune with grid search. Through prior experimentation, we selected an initial learning rate of 0.01 and a decay of 0.0003.

d) Manual Hard-Voting Ensemble

We implemented a manual, hard-voting ensemble by taking the mode of the predicted labels for the best performing models: the CNN, RNN, and DNN (both with ADASYN and hyperparameter tuning), plus the previous Soft-Voting result.

III. Results

A. *Metrics and Visualization*

We compared the harmonic mean of precision and recall (F1-score), Precision, Accuracy and Recall. We prioritized F1-score because of the serious consequences of both false positives and false negatives: water wastage and danger to public health.

We plotted Area Under Curve for Receiver Operating Characteristics (AUC ROC), Precision-Recall curves and confusion matrices for all of the above classifiers.

We also plotted the Decision boundary after dimensionality reduction (using t-SNE) [13] for the potable and non-potable classes. The diagram helps illustrate the challenge of identifying clusters, due to the lack of correlation, as it visualizes a lack of clear boundaries between the two possible results (potable and non-potable).

For the second set of deep learning classifiers, we also used history to plot f1-score, accuracy, and loss over time. This was helpful to understand the behaviour of the models.

B. *Model Parameters*

The best parameters for each of the models were as follows (note: although we tested them independently, the resulting parameters were consistent with and without ADASYN):

- CNN: 72 and 36 filters for the first and second layers, respectively, 0.05 dropout, and 36 dense units.
- RNN: layer 1 of size 36 units and layer 2 of size 9 units.
- DNN: 6 layers with 6 neurons each and a batch size of 8.

C. *Model Performance*

Manual hard-voting resulted in the best f1-score (63.21). CNN with hyperparameter tuning provided the best accuracy (70.52. Stacking with AdaBoost resulted in the highest precision (72.72). Decision Tree was the best-performing individual classifier from the traditional machine learning set in terms of f1-score (57.55).

As our data was neither image, grid-based, nor time series, we were not expecting useful results from either the CNN or the RNN [12]. But CNN had the best f1-score for an individual classifier and RNN second best. We expected better performance from the DNN. The original MLP performed better, suggesting that our choice of parameter range was far from optimal as we attempted refinements with the sequential model.

For a complete comparison of classifier performance metrics see Table 1.

IV. Discussion

Several deep learning models outperformed traditional machine learning, but not by large amounts. Further gains were sometimes possible through ensemble methods, though these were not achieved consistently. Further repeated measures and validation could provide us with more stable and reliable results, allowing us to draw more confident conclusions regarding model performance.

The biggest surprise was the performance of the CNN and RNN models. Because the data was not grid-based or sequential, we did not expect any gains beyond the traditional MLP. However, the CNNs with ADASYN proved to be one of our best performing models in terms of f1-score.

Although the original dataset contained sufficient data for analysis and machine learning, some deep learning techniques would likely benefit from even more data. We were also limited by the blank cells in the dataset, which further reduced the amount of usable data. On further investigation of the Kaggle dataset, we learned that it was synthetically generated, which may also have negatively impacted metric quality. Even assuming the viability of the synthetic data, there are likely other

TABLE I. F1-score, Precision, Recall, and Accuracy, for Each Classifier, Sorted by F1-score

Classifier	F1-score	Precision	Recall	Accuracy
Manual Hard Voting	63.21	56.01	72.54	65.89
CNN with ADASYN and hyperparameter tuning	62.61	54.38	73.77	64.40
RNN with ADASYN and hyperparameter tuning	59.62	48.46	77.46	57.62
Voting-Soft	59.33	52.35	68.44	62.09
CNN with hyperparameter tuning	58.60	67.74	51.64	70.53
Decision Tree Classifier with ADASYN and hyperparameter tuning	57.55	40.40	100.00	40.40
MLP Classifier with ADASYN and hyperparameter tuning	57.42	54.85	60.25	63.91
Voting-Hard	57.14	54.68	59.84	63.74
SVM Classifier with ADASYN and hyperparameter tuning	55.09	43.40	75.41	50.33
Random Forest Classifier with ADASYN and hyperparameter tuning	54.79	52.43	57.38	61.75
K Nearest Neighbors Classifier with ADASYN and hyperparameter tuning	54.61	49.66	60.66	59.27
MLP Classifier with hyperparameter tuning	52.71	66.05	43.85	68.21
Stacking with XGBoost Classifier	51.39	59.04	45.49	65.23
RNN with hyperparameter tuning	50.82	58.92	44.67	65.07
DNN with ADASYN and hyperparameter tuning	50.64	41.58	64.75	49.01
DNN with hyperparameter tuning	50.25	42.11	62.30	50.17
Stacking with Decision Tree Classifier	50.10	47.94	52.46	57.78
Logistic Regression Classifier with ADASYN and hyperparameter tuning	49.11	40.53	62.30	47.85
Stacking with Gradient Boosting Classifier	48.95	68.38	38.11	67.88
Stacking with AdaBoost Classifier	48.22	72.73	36.07	68.71
SVM Classifier with hyperparameter tuning	45.86	39.58	54.51	48.01
Stacking with Random Forest Classifier	45.70	51.01	41.39	60.26
Logistic Regression Classifier with hyperparameter tuning	45.34	40.92	50.82	50.50
K Nearest Neighbors Classifier with hyperparameter tuning	44.10	47.20	41.39	57.62
Random Forest Classifier with hyperparameter tuning	40.32	58.59	30.74	63.25
Naive Bayes Gaussian classifier with hyperparameter tuning	34.71	61.46	24.18	63.25
Naive Bayes Gaussian Classifier with ADASYN and hyperparameter tuning	34.71	61.46	24.18	63.25
Decision Tree classifier with hyperparameter tuning	20.28	69.05	11.89	62.25

features or factors not captured by the dataset that more significantly impact water potability than those available.

V. Conclusions

Given our maximum f1-score of 63.21 (via the manual hard voting ensemble), our current models are not sufficiently accurate to handle predictions of real-world water potability classification, where wrong predictions could lead to significant health consequences or water wastage.

Deep learning techniques show promising results as compared to traditional machine learning methods but would benefit from additional, real-world data. In this case, our results may have been limited by both the dataset size and its synthetic composition.

More research is needed on the application of different machine learning and deep learning architectures to further improve performance. We could use or expand the grid search to further improve the deep learning models. CNNs show surprising promise, so future work might also experiment with stacking the deep learning models into a hybrid. Repeated testing would help solidify our findings.

VI. Group Member Contributions

Both team members collaborated on a majority of the project throughout the term. Mahitha led more of the work on the machine learning, stacking, and ensemble models, while Hannah contributed more to the deep learning models and charts.

References

[1] OECD, Benefits of Investing in Water and Sanitation: An OECD Perspective. Paris: Organisation for Economic Co-operation and Development, 2011. Accessed: Mar. 20, 2023. [Online]. Available: https://www.oecd-ilibrary.org/environment/benefits-of-investing-in-water-and-sanitation_9789264100817-en

[2] A. Kadiwal, “Water Quality.” 2021. Accessed: Jan. 30, 2023. [Online]. Available: https://www.kaggle.com/datasets/adityakadiwal/water-potability

[3] N. Radhakrishnan and A. S. Pillai, “Comparison of Water Quality Classification Models using Machine Learning,” in 2020 5th International Conference on Communication and Electronics Systems (ICCES), Jun. 2020, pp. 1183–1188. doi: 10.1109/ICCES48766.2020.9137903.

[4] T. H. H. Aldhyani, M. Al-Yaari, H. Alkahtani, and M. Maashi, “Water Quality Prediction Using Artificial Intelligence Algorithms,” Appl Bionics Biomech, vol. 2020, p. 6659314, 2020, doi: 10.1155/2020/6659314.

[5] A. Blanco, L. V. Del Rosario, K. I. Jose, and M. Alipio, “Deep Learning Models for Water Potability Classification in Rural Areas in the Philippines,” in 2022 IEEE World AI IoT Congress (AIIoT), Jun. 2022, pp. 225–231. doi: 10.1109/AIIoT54504.2022.9817352.

[6] K. Sulaiman, L. H. Ismail, M. A. M. Razi, M. S. Adnan, and R. Ghazali, “Water Quality Classification Using an Artificial Neural Network (ANN),” IOP Conf. Ser.: Mater. Sci. Eng., vol. 601, no. 1, p. 012005, Aug. 2019, doi: 10.1088/1757-899X/601/1/012005.

[7] R. Chafloque, C. Rodriguez, Y. Pomachagua, and M. Hilario, “Predictive Neural Networks Model for Detection of Water Quality for Human Consumption,” in 2021 13th International Conference on Computational Intelligence and Communication Networks (CICN), IEEE, 2021, pp. 172–176.

[8] H. Yusuf, S. Alhaddad, S. Yusuf, and N. Hewahi, “Classification of Water Potability Using Machine Learning Algorithms,” in 2022 International Conference on Data Analytics for Business and Industry (ICDABI), Oct. 2022, pp. 454–458. doi: 10.1109/ICDABI56818.2022.10041667.

[9] M. I. K. Haq, F. D. Ramadhan, F. Az-Zahra, L. Kurniawati, and A. Helen, “Classification of water potability using machine learning algorithms,” in 2021 International Conference on Artificial Intelligence and Big Data Analytics, IEEE, 2021, pp. 1–5.

[10] “ADASYN — Version 0.10.1.” https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.ADASYN.html (accessed Mar. 04, 2023).

[11] “Imputation Method - an overview | ScienceDirect Topics.” https://www-sciencedirect-com.proxy.library.carleton.ca/topics/mathematics/imputation-method (accessed Apr. 12, 2023).

[12] J. Brownlee, “When to Use MLP, CNN, and RNN Neural Networks,” MachineLearningMastery.com, Aug. 11, 2022. https://machinelearningmastery.com/when-to-use-mlp-cnn-and-rnn-neural-networks/ (accessed Apr. 12, 2023).

[13] “sklearn.manifold.TSNE,” scikit-learn. https://scikit-learn/stable/modules/generated/sklearn.manifold.TSNE.html (accessed Apr. 04, 2023).