COMPARING THE PERFORMANCE OF CLASSICAL AND DEEP LEARNING TECHNIQUES ON CLASSIFYING WATER AS POTABLE OR NON-POTABLE

Evan Pochtar, Ryan Roche, and Evan Voogd

ABSTRACT

In this paper, we provide a final report on our project, *Comparing the Performance of Classical and Deep Learning Techniques on Classifying Water as Potable or Non-Potable*. We present our results of training sklearn's implementation of SVM and Random Forest on our dataset, with both models performing slightly below 70% on a hold-out test set. We also provide the results of our attempts at modifying the Random Forest architecture to better suit this dataset, comparing it to SVM, Multilayer-Perceptron, and Transformer based approaches.

1 THE PROBLEM

The problem is to predict if water is potable based on pH value, hardness, the amount of dissolved solids, cholarmine level, sulfate level, electrical conductivity, organic carbon level, Trihalomethane level, and turbidity using a dataset obtained from Kaggle - Tharmalingam (2024). We will build three models - a Random Forest classifier, an SVM classifier, and a Neural Network to attempt to solve this problem, and compare their respective performances.

2 WHY IS THIS PROBLEM INTERESTING?

Firstly, this problem is interesting because the dataset itself is incomplete - many rows do not have all columns present. Devising a method that is able to take advantage of the data despite this could be an interesting challenge.

Beyond the dataset itself, the problem is also interesting due to its practical nature. The benefit of a strong model for this task is clear, as lack of clean drinking water is a common issue across many communities in the Global South. Being able to accurately predict if water is safe could lead to expanded water supplies. Additionally, as will be discussed in the Previous Work section, much of the current work on this problem has been focused on Neural Networks, with comparatively fewer papers attempting to solve this task using classical models. Therefore, this would be a good opportunity to see how classical models can compare to Neural Networks, taking a problem that has previously been mostly solved with them but does not rely on architecture-specific data, such as sequential data that would clearly lend itself to deep-learning based models.

As the project progressed, another interesting dynamic emerged. Both SVM with an RBF kernel and Random Forest have had a maximum performance of just under 70%. This led us to believe that we might have quite a bit of room to improve on our training accuracy by revising these two base methods and devising new derivative models that might be better suited to the problem.

3 PREVIOUS WORK

Tiyasha et al. (2020) shows that previous research has demonstrated the effective application of various AI models in water quality assessment and prediction. As highlighted in recent comprehensive reviews, artificial neural networks (ANNs) have shown remarkable success in handling the nonlinear and complex relationships between water quality parameters, while Support Vector Machines (SVMs) have proven particularly effective in classification tasks related to water quality. The literature shows that both neural networks and SVM models have exhibited superior performance

compared to traditional statistical methods, with ANNs particularly excelling at handling missing data and complex parameter interactions. Random Forest classifiers, while less extensively studied in the water quality domain, have emerged as promising tools due to their ability to handle high-dimensional data and capture non-linear relationships between water quality parameters. These studies collectively suggest that machine learning approaches can effectively address the challenges inherent in water quality assessment, including data nonlinearity, parameter interdependence, and complex classification tasks.

Hassan et al. (2021) takes a similar approach to us in attempting to compare several different machine learning techniques suitability to this task, comparing a single-layer Neural Network, a Random Forest model, a Multinomial Logistic Regression model, an SVM model, and a Bagged Tree model. This paper found relatively comparable performance between these models (besides SVM), ranging from 98.65% classification accuracy for the single-layered Neural Network to 99.83% classification accuracy for the Multinomial Logistic Regression model. SVM on the other hand had a lower accuracy of only 96.98%. This paper, however, took a relatively simplistic approach to the problem, not analyzing the effects of different kernels on SVM performance, or any discussion of hyperparameter tuning more generally. Additionally, only a single-layered Neural Network model was attempted, and while Random Forest outperformed it, the paper did not explore if a deep neural network could have greater predictive power.

Abuzir & Abuzir (2022) additionally compares different machine learning techniques, even citing Hassan et al. in its literature review. The paper compares the J48, Naïve Bayes, and multi-layer perceptron algorithms against each other, using multiple metrics including root mean squared error, receiver operating characteristic area, and precision-recall curve to evaluate their performance. As opposed to focusing exclusively on comparing the performance of the respective models against each other, the paper additionally evaluates how the performance of the models change when differing amounts of PCA components are used in lieu of the raw data samples. The MLP model was consistently the best-performing in terms of classification accuracy. However, the gap in accuracy between it and the other models shrank considerably when PCA was used, going from a 3% gap between it and the next-most accurate model when all features were used, to 0.15% in the PCA test with the most principal components used. Aside from the addition of PCA, the paper does not do much else to add to the discussion regarding the effectiveness of different network architectures for classifying water potability. No mention was made of whether deeper neural networks could produce more accurate classifications (the MLP used only had one hidden layer), and the missing values in the training data were simply filled in with normalized values- no models that are robust to missing features were evaluated.

4 GOAL

The goal of our project was to compare the efficacy of various classical models to a neural network, which seems to be the standard model of choice for this prediction task. After compiling our preliminary results, our goal was to develop a novel variation on both SVM and Random Forest to see if we are able to improve their performance compared to the base models, as well as the standard tool for this class, a neural network.

5 RESULTS

5.1 RANDOM FOREST

5.1.1 RESULTS OF BASE MODEL

We trained and optimized hyperparameters for a Random Forest predictor using the sklearn library. Our hyperparameter tuning finalized on a model using the Gini criterion, a maximum depth of 168, and 244 estimators. Testing on the holdout test set (20% of the total data) led to a classification accuracy of 68%, significantly less than that found by Hassan et al. (2021). This leads us to believe that modifying the model should have room to gain superior performance over using the base sklearn model. This low accuracy seemed to be driven primarily by potable water being misclassified as non-potable water, with a recall of 34%, versus a recall of 88.1% for non-potable water.

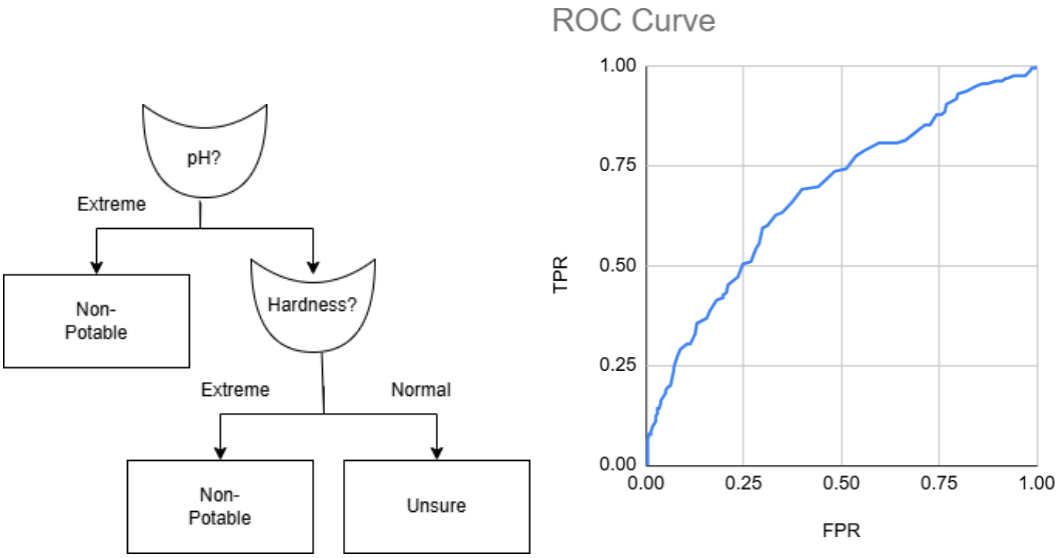

Figure 1: Figures for Random Forest. Left - Individual decision tree within Random Forest, trained on pH and hardness. Right - Receiver Operating Characteristics Curve of Random Forest model within Learning-Cutoff Random Forest Model, x-axis is false positive rate, y-axis is true positive rate. Random predictor can be visualized as a straight line from the bottom left to top right, perfect predictor would trace the left and top sides.

5.1.2 Issues with base model

For the random forest model, we devised a modification to handle the issue that our models are able to identify non-potable water easily, but often misclassifies potable water as non-potable. Our theory, in terms of the random forest model's reason for this occurring falls into the actual architecture of a random forest. Only one feature needs to be problematic in order to make a water source non-potable. For instance, if the water has a high level of organic compounds, it won't be safe to drink, no matter the pH level. As random forest excludes features when training each individual classifier, this means that a portion of the non-potable training data for each classifier is not going to contain the information that places it into this class. E.g. if a training point has a high level of organic compounds, but a safe pH and hardness level, a classifier where the only selected features are pH and hardness will erroneously learn that this data's pH and hardness are unsafe.

This explains why the classifiers are able to identify non-potable water reasonably well. Consider Figure 1a, which shows one hypothetical decision tree trained on the features pH and hardness. If the water has an extreme pH or hardness, this tree should be able to identify this easily, as its pH and hardness will be significantly different from that of the potable training data. However, if its pH and hardness is normal, the model is much more unsure. This rightmost branch (Normal pH and normal hardness) has been trained on both non-potable and potable data, however, as their pH and hardness are normal, it is effectively training on random noise. Therefore, it likely will simply return either potable or non-potable with a likelihood for an individual predictor of approximately:

$$P(\text{predicts potable}) = \frac{\text{Number of Potable Data Points}}{\text{Number of Data Points with normal pH and hardness}}$$

For non-potable water, some proportion of the trees are trained on the features that are driving its non-potability, so the probability of the entire random forest predicting non-potable is likely:

$$P(\text{true non-potable prediction}) = \sum_{i \in [N(p_{\text{extreme}})]} \frac{1}{N} + \sum_{i \in [N(1-p_{\text{extreme}})]} \frac{p_i}{N}$$

where N is the number of classifiers, p_{extreme} is the percentage of individual classifiers trained on the extreme features for a particular point, and p_i is the percentage of non-potable points with normal

values for the features that classifier i was trained on. The equivalent expression for a true potable prediction is:

$$P(\text{true potable prediction}) = \sum_{i \in [N]} \frac{1 - p_i}{N}$$

Note that, in the training data, only around 40% of the training points are potable. While $1 - p_{\text{extreme}}$ is higher than this, as some proportion of the points are going to have extreme values for the trained features, this is still a relatively low probability. Meanwhile, as 60% of the data is non-potable, the second term of the true non-potable prediction will already be close to the 50% cutoff, while the first term will usually be large enough to guarantee it being above 50%.

5.1.3 REVISED MODEL

With this in mind, our idea was to attempt to learn this "noise" rate, e.g. the proportion of false non-potable predictions on potable data, so a different cutoff could be used that is above the background noise while still being low enough that the first term of the true non-potable equation could be sufficient to result in a non-potable prediction. With this in mind we devised the following algorithm:

Algorithm 1 Learning-Cutoff Random Forest

Input: training set $S = \{(\boldsymbol{x}_i, y_i)\}_{i \in [N]}$, $N_{\text{estimators}} \in \mathbb{N}$, $D_{\text{max}} \in \mathbb{N}$, criterion $\in$ [gini, entropy, log_loss], $v_{\text{size}} \in (0, 1)$.

split S into S_{train} and S_{cutoff}, such that $|S_{\text{cutoff}}| = v_{\text{size}}|S|$
train random forest model on S_{train}, with $N_{\text{estimators}}$ estimators, the given criterion, and a max depth of D_{max}.
probs $\leftarrow$ proportion of trees predicting class 1 for each point in S_{cutoff}.
best_cutoff $\leftarrow 0$
best_accuracy $\leftarrow 0$
for $i = 0, ..., N_{\text{estimators}}$ **do**
    Create new array predictions of length $|S_{\text{cutoff}}|$.
    for $j = 0, ..., |S_{\text{cutoff}}|$ **do**
        predictions[j] $\leftarrow$ probs[j] $> \frac{i}{N_{\text{estimators}}}$
    end for
    accuracy $\leftarrow$ accuracy score of predictions
    if accuracy $>$ best_accuracy **then**
        best_cutoff $\leftarrow \frac{i}{N_{\text{estimators}}}$
        best_accuracy $\leftarrow$ accuracy
    end if
end for

Output: Classifier that takes $\boldsymbol{x}$, passes it into random forest, and returns 1 if the proportion of trees predicting class 1 for $\boldsymbol{x}$ is above best_cutoff, and 0 otherwise.

Our idea was that the cutoff with the highest score on the holdout set would likely be high enough to account for the noise, but low enough to not outweigh the trees with the extreme features.

5.1.4 RESULTS

Our modification was only able to slightly outperform the base random forest model on the test set, with an accuracy score of 69.7% versus the base model's 68%. Looking at the recall rates for the two classes, our model did not have the effect that we expected, with a recall for class 1 of 92.3% (versus 88.1% in the base model) and a recall for class 0 of 30.7% (versus 34% in the base model). We especially did not expect the performance to diminish on class 0, we expected, if anything, for this to involve a tradeoff between misclassifications moving from class 1 to class 0, trading a small decrease in class 1 recall to gain a larger increase class 0 recall.

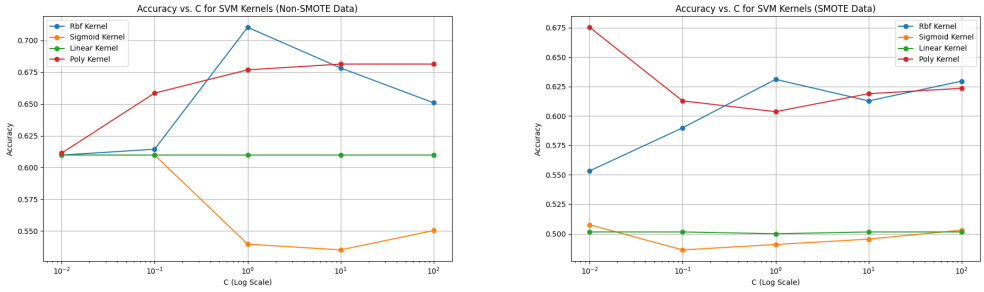

Figure 2: Left - Comparison of SVM kernel performance (RBF, Sigmoid, Linear, Polynomial) across different C values for non-SMOTE data, Right - Comparison of SVM kernel performance (RBF, Sigmoid, Linear, Polynomial) across different C values for SMOTE-applied data.

To explain this result, we can look closer into the internals of the model. The model chose a recall score of 50%, exactly the standard cutoff. It's possible that, with our limited data set, that the decrease in training data from the cutoff-optimizing set resulted in a worse performing inner model at predicting class 1 than the random forest model trained on the full training set, and with the Random Forest's cutoff optimal already, no effect was seen by the cutoff itself.

To try and understand better why this model did not perform as expected, we created a ROC curve, shown in Figure 1b. By using an ROC curve, we are able to visualize the trade-off between false and true positives made by adjusting our cutoff. The Learning-Cutoff Random Forest model essentially picks a point along the ROC curve to optimize this trade-off, optimizing for accuracy score. For our Learning-Cutoff Random Forest model to outperform the base Random Forest model, there would need to be some sort of plateau, representing the point where TPR can be increased with a low increase in cost of higher FPR. However, this does not exist in the curve, instead being relatively consistent - hence the model's choice of 50%. With this in mind, it is clear why changing the cutoff alone is not enough to make up for the lack of performance of the Random Forest model, and the performance is likely due to the structure of the model not being a good fit for the data, as explained earlier in this section.

5.2 Support Vector Machine

The results demonstrate that the RBF kernel achieved the highest accuracy across most scenarios, particularly when using $C = 1$. On the original dataset, the RBF kernel produced an accuracy of 71%, outperforming the other kernels. This success can be attributed to the RBF kernel's ability to capture complex, non-linear relationships between features, which likely better reflects the underlying decision boundary for potability classification. Interestingly, while the RBF kernel's performance decreased slightly when SMOTE was applied, resulting in an accuracy of 63%, it showed a more balanced recall between the two classes. For instance, on the SMOTE-enhanced dataset, recall for the minority class (class 1) improved significantly from 33% in the original dataset to 60%, suggesting that SMOTE effectively mitigated the dataset's imbalance, even if it slightly reduced overall precision.

The regularization parameter C had a significant impact on the performance of all kernels, with $C = 1$ consistently providing the best results for both the RBF and polynomial kernels. This indicates that a moderate trade-off between margin size and misclassification worked well for this dataset. Lower values of C, such as $C = 0.01$, often led to underfitting, as observed most clearly in the original dataset showing that most algorithms hovered around the same accuracy of 63%. Although, interestingly, the polynomial kernel performed very well with a low value of C=0.01, having the highest accuracy in the SMOTE dataset at 67.5%. Conversely, very high values of C, such as $C = 10$, sometimes overemphasize fitting the training data, reducing generalization. The parameter C is critical in SVMs because it controls the balance between decision margin size and training error. Lower values prioritize a larger margin but tolerate more misclassifications, favoring simpler models, while higher values force the model to minimize misclassification, potentially leading to

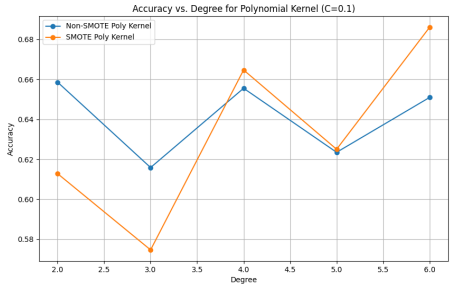

Figure 3: Accuracy of polynomial kernel SVM across varying degrees (2–6) with and without SMOTE preprocessing (C=0.1)

overfitting. In non-linear kernels like RBF, $C = 1$ strikes an effective balance, achieving good generalization and fitting the data's complexity.

Across all tested configurations, the RBF kernel consistently delivered the best results, particularly for the original dataset where its ability to model non-linear patterns proved crucial. The sigmoid kernel performed poorly, particularly for the minority class, with some configurations failing entirely to classify potable water correctly. For example, with $C = 0.01$ on the original dataset, recall for class 1 was 3%, indicating severe underfitting. Similarly, the linear kernel achieved a maximum accuracy of 61%, even with SMOTE applied, reflecting its inability to capture the dataset's non-linear structure. The polynomial kernel provided reasonable performance, achieving an accuracy of 68% on both the original and SMOTE datasets. However, its recall for class 1 remained low, suggesting it was less effective in addressing the dataset's class imbalance.

The class imbalance in the original dataset significantly impacted recall for the minority class. In the original dataset, recall for class 1 was as low as 33% with the RBF kernel and 3% with the sigmoid kernel. Applying SMOTE improved recall for class 1 across all kernels, although this often came at the expense of reduced precision for the majority class. This trade-off between recall and precision is expected when synthetic data introduces slight noise into the dataset. Despite this, the RBF kernel with $C = 1$ on the original dataset emerged as the best configuration, achieving the highest overall accuracy of 71% and a weighted F1-score of 0.67. This kernel likely performed well because it captured the non-linear patterns in the dataset effectively, while $C = 1$ provided an appropriate balance between margin maximization and minimizing misclassification. These results emphasize the importance of kernel selection, tuning hyperparameters like C, and addressing class imbalance when optimizing SVM models.

5.2.1 Polynomial Kernel Degree Analysis

An analysis of the polynomial kernel's degree revealed notable findings, particularly for C=0.1. On the non-SMOTE dataset, degree 2 achieved an accuracy of 65.85%, outperforming degrees 3, 5, and 6, which scored 61.59%, 62.35%, and 65.09%, respectively, while degree 4 performed similarly at 65.55%. For the SMOTE-augmented dataset, higher-degree polynomials excelled, with degree 6 achieving the highest accuracy of 68.60%, followed by degree 4 at 66.46% and degree 2 at 61.28%. These results suggest that SMOTE shifted the performance hierarchy, benefiting higher degrees due to the synthetic data's effect on the decision boundary complexity. The strong performance of degree 2 on the non-SMOTE dataset highlights its suitability for capturing the underlying structure without overfitting, particularly in the presence of limited and imbalanced data. In contrast, higher-degree polynomials likely introduced unnecessary complexity. While SMOTE enhanced the performance of more flexible kernels, it slightly diminished the effectiveness of degree 2, likely due to changes in the data distribution. This analysis underscores the critical role of kernel hyperparameter selection, demonstrating how degree 2 balances simplicity and generalization, making it effective for datasets prone to overfitting and class imbalance.

5.3 NEURAL NETWORKS

5.3.1 MULTI-LAYER PERCEPTRON

A custom multi-layer perceptron neural network was built and evaluated after training with different hyperparameter combinations. The neural network consisted of five fully-connected layers, including the input and output layers. The three hidden layers are of dimensions 32, 24, and 16, respectively, with each hidden layer performing batch normalization with a ReLU activation function followed by a dropout layer. From the grid search, it was determined that the best combination of hyperparameters was a learning rate of 10^{-3}, dropout probability of 0.2, and batch size of 128, yielding an accuracy of 66%. This indicates that the model is somewhat able to establish an understanding of the patterns in the data, but is limited by its linear structure.

5.3.2 TABTRANSFORMER

Amazon's TabTransformer, proposed in Huang et al. (2020), is widely considered to be the one of the best available models for tabular data. The network takes a split-head approach, passing categorical features into a transformer and concatenating them with the continuous features before running them through a multi-layer perceptron. Since the features in the water quality dataset were all continuous, the transformer component of the model was effectively vestigial in this application. Unsurprisingly, this restricts the model's potential for performance, only yielding an accuracy of 57.11% on the data. This indicates that the model is unable to adequately capture the non-linear patterns in the data, which is to be expected given that the only "active" part of the model is the multi-layer perceptron. The transformer does most of the heavy lifting in this model, but since we have no categorical features to give it, it isn't much use in this application.

6 CONCLUSION

Contrary to some of the previous literature on the subject, all three classes of models - Random Forests, SVM, and Deep Learning approaches - perform roughly equally on this classification problem, ranging from 66-71% in peak performance. Similarly, both Random Forest and SVM exhibit poor recall rate for potable water. Some portion of this could be due to the structure of the models, specifically the Random Forest and the TabTransformer model, being ill-suited for this particular problem. Despite this, some portion of the performance is likely due to the limited sample size, especially for potable water, as shown by the improved class 1 recall rate of SVM when employing SMOTE. This result however, could be safer in certain situations, as the overpredicting of non-potable water is preferable to labeling non-potable water as potable. Also in this dataset, it isn't outlined what necessarily is contributing to the classification of potable water, which could allow us to identify more about the problem itself. Overall, our project did not find any significant differences between classical and deep learning approaches for this particular problem, although we were able to identify some conceptual issues with the Random Forest model being applied to this problem that might make it less performant given more data. In the future, a closer examination of the data itself to see if one of the measurements is overrepresented, or what exactly the cause of this "cap" of a 71% accuracy rate really is.

7 METHODS

7.1 RANDOM FOREST

The Random Forest Model and Learning-Cutoff Random Forest models were tuned using SciKit-Learn's GridSearchCV tool, with a 3-fold Cross-Validation approach tuning the maximum depth of the individual predictors (from 1 to 250), the number of individual predictors (from 1 to 250), the criterion (Gini, Entropy, and Log Loss), and additionally the proportion of the training set to hold out to optimize the cutoff (values ranging from 0.33 to 0.1).

The base Random Forest model used in the implementation of Learning-Cutoff Random Forest, as well as for the initial results of the Random Forest model is SciKit-Learn's implementation. The models were evaluated on a holdout test set of 20% of the dataset (with a seed of 42). The ROC

MLP Model	
Learning Rate (α)	$0.01, 0.001, 0.0001$
Dropout Probability (p)	$0.1, 0.2, 0.3$
Batch Size (b)	$32, 64, 128$

TabTransformer	
Learning Rate (α)	$10^{-3}, 10^{-4}, 7.5\text{E}-5, 5\text{E}-5$
Depth (L)	$3, 4, 5, 6$
Embedding Dimension (d)	$16, 24, 32$

Table 1: Hyperparameter Values Tested for Neural Networks

Curve (Figure 1b) uses the data from the holdout set used to optimize the cutoffs, as it visualizes the trade-off that the model was optimizing.

7.2 Support Vector Machines

To classify potable and non-potable water, a Support Vector Machine (SVM) was utilized, focusing on identifying the best-performing kernel and the effects of class balancing through SMOTE (Synthetic Minority Oversampling Technique) and regularization parameter C. The dataset, with missing values imputed using column means, was split into stratified training and testing sets (80/20 ratio) to preserve class proportions. Features were standardized using StandardScaler for zero mean and unit variance. Four SVM kernels—linear, RBF, sigmoid, and polynomial—were tested on both the original and SMOTE-augmented datasets, with the polynomial kernel further evaluated across degrees 2 to 6. The parameter C, which balances training error minimization and decision margin size, was systematically varied ($C \in [0.01, 0.1, 1, 10, 100]$) to optimize performance. Accuracy scores were computed on the test set, and trends were visualized to analyze the influence of kernels, SMOTE, and C on model performance. This comprehensive approach provided a robust evaluation of SVM configurations for water potability classification.

7.3 Neural Networks

The Multi-Layer Perceptron model was optimized via a manual grid search to tune the learning rate, probability of the dropout layers, and batch size hyperparameters of the model. For each combination of hyperparameter values from the table, the model was trained for 200 epochs using a randomly distributed, four-to-one split of training and validation data pulled from the original dataset. PyTorch's BCELoss was used as the criterion for the training of this model, with an Adam optimizer used for training.

For the TabTransformer, the model was optimized in a similar manner. Four hyperparameters of the model were tuned in the grid search, those being the learning rate, model depth (the number of transformer encoder layers), and dimension (size of the feature embeddings). The model was trained for 200 epochs with each combination of hyperparameter values, using PyTorch's Adam optimizer with BCEWithLogitsLoss as the criterion.

The models trained with the best-performing hyperparameter combination for each respective model type were evaluated using a holdout set of the dataset that was excluded from the training data. The holdout data was randomly selected using Scikit-Learn's train_test_split function with a seed of 42.

References

Saleh Y. Abuzir and Yousef S. Abuzir. Machine learning for water quality classification. *Water Quality Research Journal*, 57(3):152–164, 05 2022. ISSN 2709-8044. doi: 10.2166/wqrj.2022.004. URL https://doi.org/10.2166/wqrj.2022.004.

Md. Mehedi Hassan, Md. Mahedi Hassan, Laboni Akter, Md. Mushfiqur Rahman, Sadika Zaman, Khan Md. Hasib, Nusrat Jahan, Raisun Nasa Smrity, Jerin Farhana, M. Raihan, and Swarnali Mollick. Efficient prediction of water quality index (wqi) using machine learning algorithms. *Human-*

Centric Intelligent Systems, 1:86–97, 2021. ISSN 2667-1336. doi: 10.2991/hcis.k.211203.001. URL https://doi.org/10.2991/hcis.k.211203.001.

Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. Tabtransformer: Tabular data modeling using contextual embeddings, 2020. URL https://arxiv.org/abs/2012.06678.

Laksika Tharmalingam. Water quality and potability, 02 2024. URL https://www.kaggle.com/datasets/uom190346a/water-quality-and-potability.

Tiyasha, Tran Minh Tung, and Zaher Mundher Yaseen. A survey on river water quality modelling using artificial intelligence models: 2000–2020. *Journal of Hydrology*, 585:124670, 2020. ISSN 0022-1694. doi: https://doi.org/10.1016/j.jhydrol.2020.124670. URL https://www.sciencedirect.com/science/article/pii/S002216942030130X.