CS542 Team 2 Water Potability

Keshav Maheshwari **Weiqi Ji** **Renjian Zheng** **Yiqin Zhang**

Abstract

Water potability is one of the most crucial aspects of maintaining a healthy population. For centuries, this has been a problem especially in developing countries and societies. However, due to the recent advancements in machine learning, it is now possible to simply collect a few data points about the water in question to predict whether or not it is potable with at least 70% accuracy. This paper explores three supervised machine learning models through which we predicted water potability with the help of the dataset attached. The three models are K-Nearest Neighbors (KNN), Logistic Regression (LR), and Support Vector Machines (SVM). Among these models, KNN came out on top with a 66% accuracy.

1 Introduction

Water potability is significant for how we can lift the developing countries of the world out of poverty and improve their quality of life. About 1.1 billion people lack access to potable water across the globe. These public health emergencies need to be addressed with a strong scientific lens, especially one that has the potential to eradicate this issue once and for all.

There have been various machine learning models in the industry used to predict water potability such as Support Vector Machines (SVM), Neural Networks (NN), Deep Neural Networks (Deep NN), K-Nearest Neighbors (KNN), etc. The primary dataset used by most reliable models is the one we used for this project, retrieved from the Kaggle database.

We used KNN, LR, and SVM to make our predictions. More specifically, we employed the resources available to us through the Python Pip packages namely SciPi and SciKitLearn. Further, we had the luxury to use at our disposal the paramount resources provided by the Boston University Shared Computing Cluster (SCC) which is designated primarily for running machine learning and data heavy models.

It is also important to note the context of our project and an overview of how we arrived at the results mentioned in the following sections. Our initial intuition was to produce our hand-made in-house machine learning models. After dozens of hours of Python computations we realized it was best to leave these models to the experts- Python libraries such as SciKit and SKLearn. These libraries produced slightly worse results but also balanced the data prediction in a more believable way, which meant that the data was being processed more correctly than the in-house models we made. We decided to stick with the SKLearn prediction models and our code will be mirroring exactly that.

It is also important to mention that for our presentation, we had run all our models with skewed data, which meant we could not trust the accuracy results of the models as they were heavily biased on the negative side for water potability. However, during the time between our presentation and now, we have been able to successfully correct the data skewness in the water potability dataset, which meant we could rely on our accuracy scores with much higher confidence, since the data imbalance had been eliminated.

2 Methods

The objective of this project is to create a model to predict water potability using machine learning techniques including K-Nearest Neighbors (KNN), Logistic Regression (LG), Support Vector Machines (SVM). The workflow is in Figure 1.

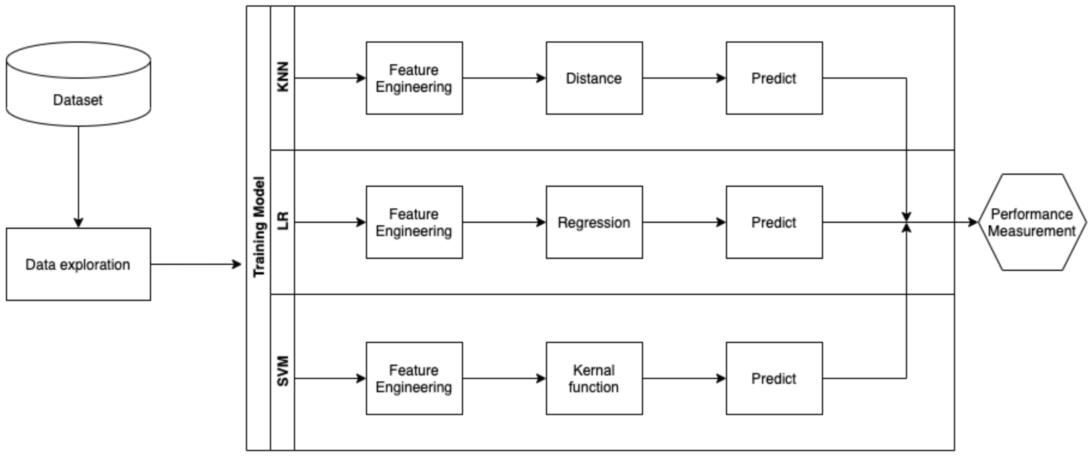


Figure1: Methodology flow

3 Data

3.1 Dataset

Before we dive deep into data processing, we want to introduce you to our data. The water potability dataset was acquired from Kaggle https://www.kaggle.com/adityakadiwal/water-potability (accessed on 18 June 2021).

This dataset contains 3276 data points with nine important features: ph, Hardness, Solids, Chloramines, Sulfate, Conductivity, Organic_carbon, Trihalomethanes, Turbidity. There are missing data only in three features, 781 missing data in sulfate, 491 in ph, and 162 in Trihalomethanes, respectively. Figure 2 shows how the missing values are distributed.

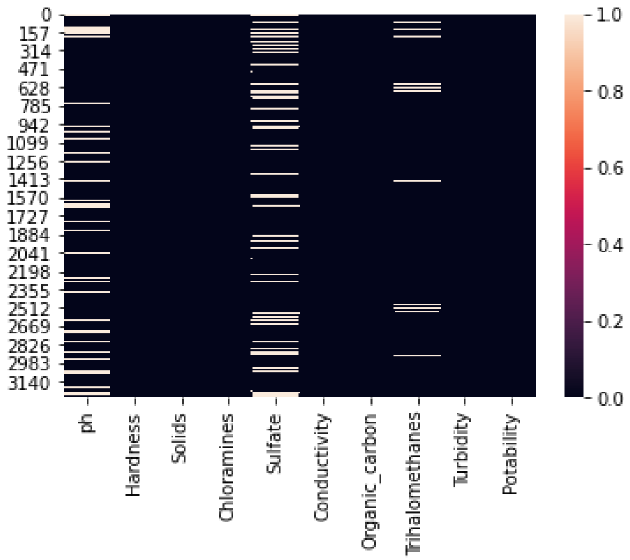


Figure 2: NaN Density Map

3.2 Feature Engineering

3.2.1 Imputation

For imputation of values in the dataset, we use standard practices used throughout the industry. Since we did not have any non-numerical values, we did not have to worry about imputing those. All the missing data for the numerical values were imputed through the calculation of the local conditional mean based. For the conditional mean, we simply calculated the mean for all the values in the column that also reported the same classification as the missing value to be imputed.

3.2.2 Normalization

Normalization of data was also done through standard practices seen throughout industry. We used z-scaling and minmax scaling which helped get the values between -1 and 1.

3.2.3 Train, Test, Split

Since our dataset was a single complete unsplit dataset, we had to manually split to produce our train and test sets. We chose to have a distribution of 80% train data and 20% test data.

3.2.4 Resampling for Skewness Correction

As with any imbalance dataset, we had to make sure the preprocessing would correct for any prior skewness. We used SKLearn's resample function to downsample the negative classification as it was overrepresented heavily in the original data.

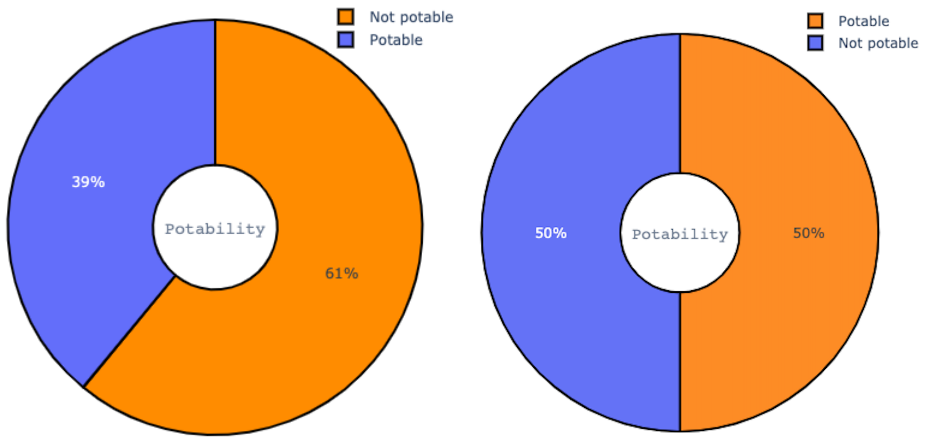


Figure 3: Water Potability Chart Before and After Resampling

4 Training Models

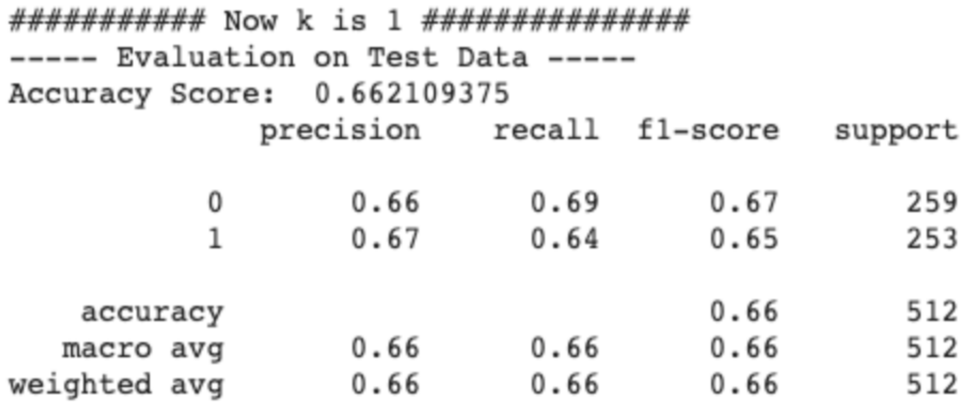
4.1 K-Nearest Neighbors (KNN)

The KNN algorithm is one of the major traditional machine learning algorithms used for classification. The KNN algorithms use K-neighbor values to find the closest point between the objects. The Euclidean Distance Metric (Di) was applied to find the nearest neighbor in the features vector.

$$D_i = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

where x1 , x2 , y1 , and y2 are variables for input data.

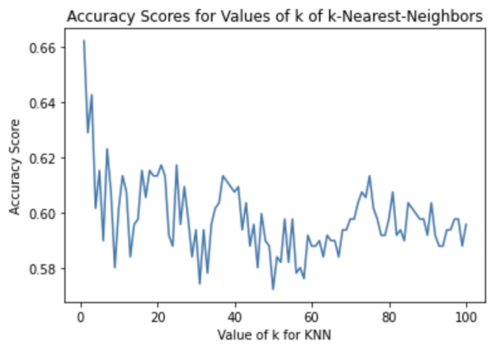
The K-value is used to find the closest points in the feature vectors. Sort to get the closest K-neighbors and get the most frequent classification among these K neighbors, and then return the predicted unique classification value. We employed different techniques to train KNN models. We increase k from 1 to 100 and get the best performance at k = 1 (after we have balanced the dataset).



```
########### Now k is 1 ###############
----- Evaluation on Test Data -----
Accuracy Score:  0.662109375
              precision    recall  f1-score   support

           0       0.66      0.69      0.67       259
           1       0.67      0.64      0.65       253

    accuracy                           0.66       512
   macro avg       0.66      0.66      0.66       512
weighted avg       0.66      0.66      0.66       512
```



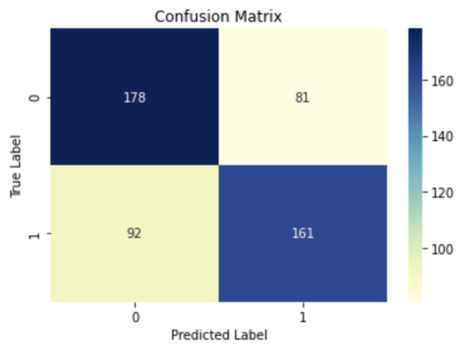


Figure 4. KNN Results

4.1.1 KNN Observations

We see above in our KNN graph in which values of K range from 1 to 100. It is rare to see a dataset which has the most optimal K as being 1. However, we reconfirmed online with other researchers and they received similar values of the most optimal K. We speculate this might be because of the stark similarities between our training and testing datasets.

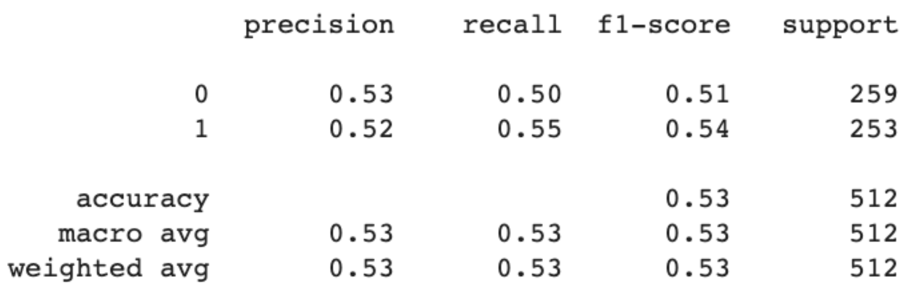
Moreover, the KNN algorithm is easy to implement. The main advantage is simplicity especially because there is no need to tune several parameters, or make additional assumptions. Reliability is another advantage here, as KNN produces the highest accuracy for us, and is also used as a significant resource for industry projects used to predict models that are of significant importance.

Disadvantages of this model can be that they are slow and computationally intensive, at least for us. To produce a graph of up to 100 neighbors, it was predicted to take 12 hours until completion, but to be fair, we were running our in-house code that we used to predict. However, parallel programming and SCC's 32 core computers saved the day in 6 minutes. In the end though, we ended up using SKLearn's KNN as it produced better accuracy and almost a 10 fold improvement in performance.

4.2 Logistic Regression Model (LR)

Logistic regression is a classification algorithm based on the logistic function and the sigmoid function, it is especially a good method to use for dependent variables that are binary. After we impute the dataset and fill in the missing data, we can then visualize the dataset and map the outputs to the given parameters of the inputs through regularization. Using the sigmoid function, the model will be able to classify categorical results, in this case between 0 and 1, by comparing the outputs with the curve of the sigmoid function. In our case, we have implemented a simple logistic regression model using libraries for scikit learn, and we were able to produce somewhat less-than optimal results.

$$Sigmoidal : F(x) = \frac{1}{1 + exp(-x)}$$



```
              precision    recall  f1-score   support

           0       0.53      0.50      0.51       259
           1       0.52      0.55      0.54       253

    accuracy                           0.53       512
   macro avg       0.53      0.53      0.53       512
weighted avg       0.53      0.53      0.53       512
```

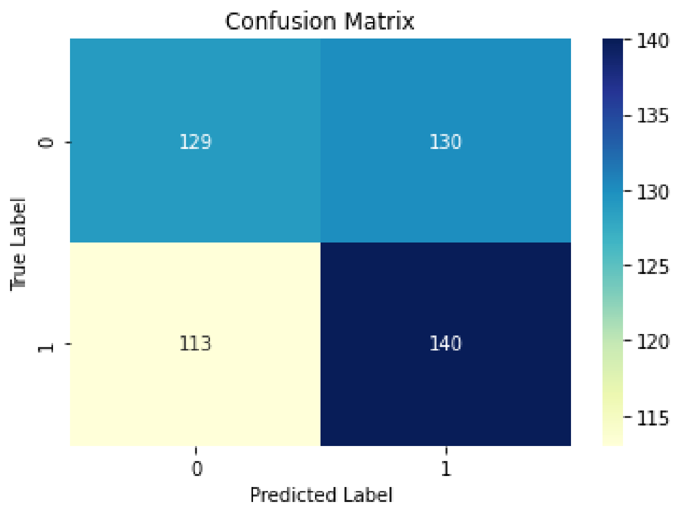


Figure 5: Logistic Regression Results

4.2.1 Logistic Regression Observations

The logistic regression model we have implemented produced less than desired results because it is producing an accuracy that is barely better than random guessing (which would have returned 50/50). However, our results show that the accuracy of our logistic regression is around 53%, which is up to par with some of the current logistic implementations by fellow researchers on the same dataset. However, because this data is not linearly separable, we analyzed that this is detrimental to the performance of the logistic regression because it would have a hard time regularizing the input features as vectors.

4.3 Support Vector Machines (SVM)

4.3.1 Support Vector Classifier (SVC)

The SVC model was the first model we tried (without the kernel function). The model used to minimize the cost function:

$$J(\mathbf{w}) = \frac{1}{2}||\mathbf{w}||^2 + C\left[\frac{1}{N}\sum_{i}^{n} \max\left(0,\ 1 - y_i * (\mathbf{w} \cdot x_i + b)\right)\right]$$

by calculating the gradient of the cost function:

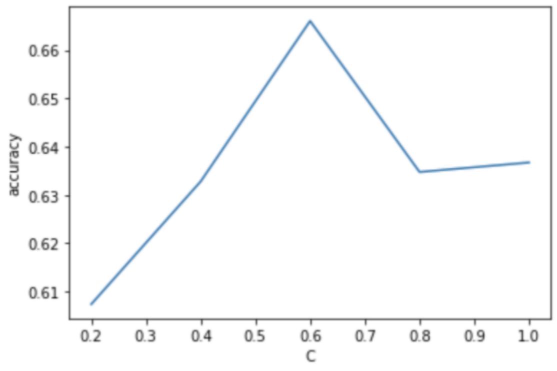
$$\nabla_w J(\mathbf{w}) = \frac{1}{N}\sum_{i}^{n} \begin{cases} \mathbf{w} & \text{if } \max(0,\ 1 - y_i * (\mathbf{w} \cdot x_i)) = 0 \\ \mathbf{w} - C y_i x_i & \text{otherwise} \end{cases}$$

4.3.2 SVM Training

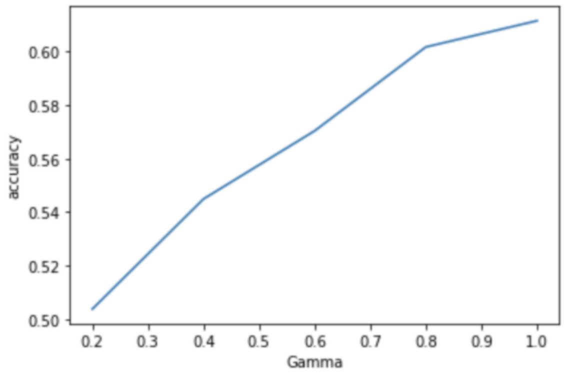
For SVM, two kernel functions, polynomial kernel function and radial basis function, were used separately. For each kernel function, two parameters, C and Gamma, were tuned separately. In our case, we implemented the SKLearn SVM model with different kernel functions each time.

1. Polynomial Kernel Function

    - Tune for C values. Here we use C = 0.2, 0.4, 0.6, 0.8, 1.0. Gamma stays as 1.0

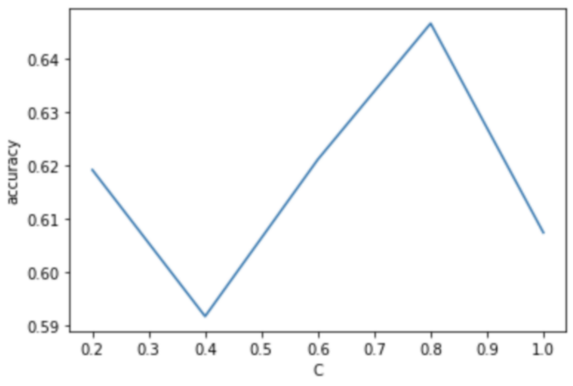


    - Tune for gamma. Here we use gamma = 0.2, 0.4, 0.6, 0.8, 1.0. C stays as 1.0

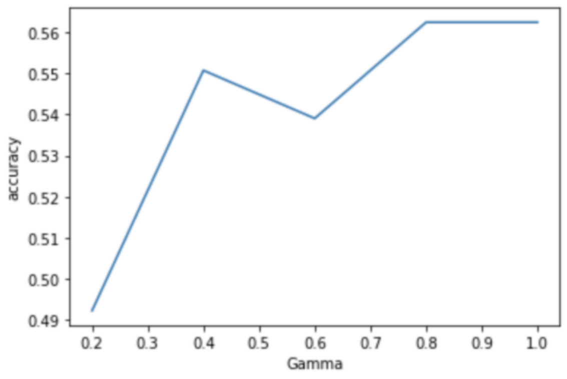


2. Radial Basis Function

    - Tune for C values. Here we use C = 0.2, 0.4, 0.6, 0.8, 1.0. Gamma stays as 1.0



- Tune for gamma. Here we use gamma = 0.2, 0.4, 0.6, 0.8, 1.0. C stays as 1.0



4.3.3 Model Result

SVC does not work well with our dataset. The model was predicting all class 0's instead of some 0's and some 1's. One of the potential reasons is that our dataset is not linearly separable, so without a kernel function, there is no best fit hyperplane.

However, SVM gave back optimal predictions of about 64% accuracy. Here is the result using Radial Basis Function, with parameter C = 1.0 and Gamma = 1.0. The confusion matrix in figure 6 shows that after we fix the problem of imbalance highlighted during our presentation, the prediction is more balanced and reliable.

```
Accuracy Score:  0.640625
              precision    recall  f1-score   support

        -1.0       0.61      0.76      0.68       252
         1.0       0.69      0.52      0.60       260

    accuracy                           0.64       512
   macro avg       0.65      0.64      0.64       512
weighted avg       0.65      0.64      0.64       512

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

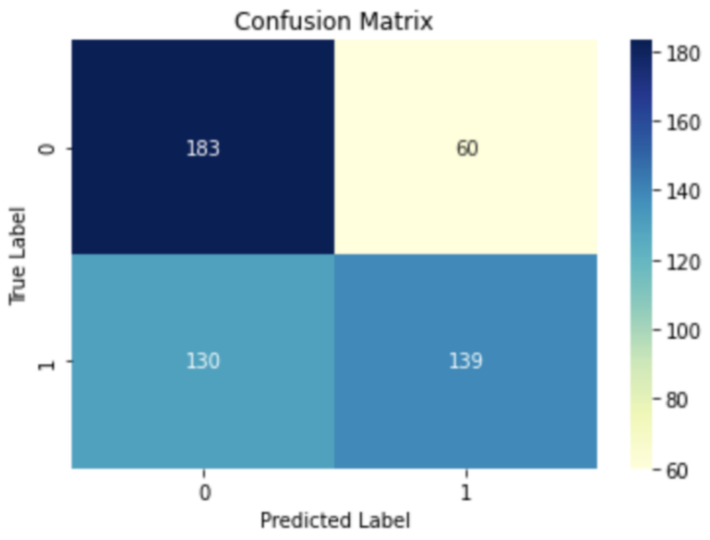


Figure 6: SVM Results

4.3.4 SVM Observations

SVC does not work well with nonlinearly separable dataset. When we implemented our own SVC model (without kernel function) on the dataset, we got undesirable results, which is the reason why we switched to SKLearn SVM model, which is good for solving non-linearly separable datasets.

We also want to note that SVM is quite different from KNN, as there are a couple of parameters that can be tuned, i.e. C, gamma, and different kernel functions to choose from.

5 Results and Discussion

For our results and discussion, it is quite important to mention the various performance measurements we recorded throughout our prediction models. Please note that these performance measurements include variables such as accuracy, precision, recall, F-score, etc. Further, we want to emphasize for the formulae below that TP refers to true positive, TN refers to true negative, FP refers to false positive and FN refers to false negative.

1. **Accuracy** shows the number of correct predictions over all predictions.

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

2. **Precision** is the number of true predictions over all predictions of the positive classification.

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

3. **Recall** shows the number of predictions that were ranked positive that were actually predicted correctly.

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

4. **F1 Score** calculates the balance between the precision and the recall so that we can have a better idea of what the final accuracy looks like.

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

5.1 Results

KNN performs the best on this classification appropriate dataset, with an accuracy of 0.66, precision of 0.67, recall of 0.64, and F1 score of 0.65.

LR performs the worst on this classification appropriate dataset, with an accuracy of 0.53, precision of 0.52, recall of 0.55, and F1 score of 0.54.

SVM performs the second best on this classification appropriate dataset, with an accuracy of 0.64, precision of 0.69, recall of 0.52, and F1 score of 0.60.

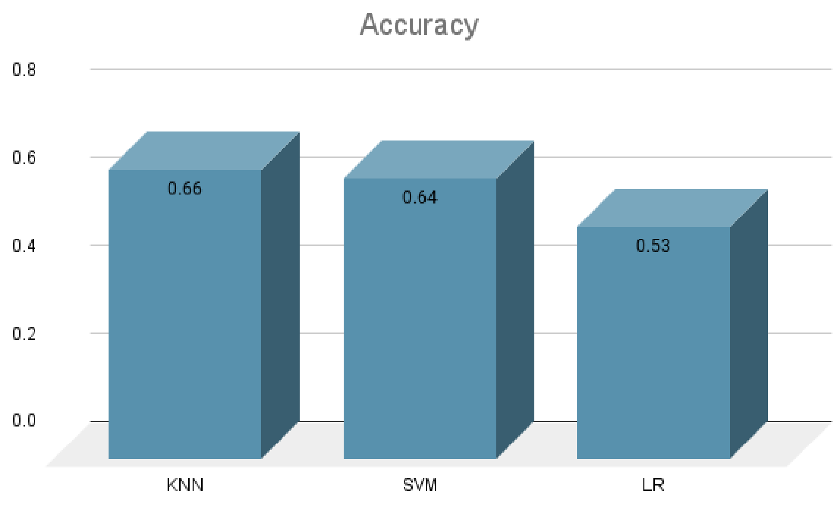


Figure 7: Performance plot for KNN, SVM, LR

6 Conclusion

To conclude, we want to reiterate that we were able to achieve industry-wide accuracy as seen in our fellow researchers' papers on Kaggle. We were satisfied with the results received through KNN and SVM, however, we would not choose Logistic Regression for a dataset like this in order to produce reliable results.

We also want to emphasize that we chose this water potability dataset due to its rising importance in the global discourse more recently. We want to contribute to ensuring a healthy and safe environment, especially for people in developing countries. If such advanced machine learning models can one day predict water potability cheaply and efficiently with an accuracy of more than 99%, then half of the world's problems would be solved overnight. We aim to be machine learning scientists who will contribute to the improvement of conditions throughout the world, because we know that machine learning gives us that power. With great power comes great responsibility, and we cannot let it go to waste.

References

[1]Beyer, K.; Goldstein, J.; Ramakrishnan, R.; Shaft, U. When is "nearest neighbor" meaningful? In Proceedings of the International Conference on Database Theory, Jerusalem, Israel, 10–12 January 1999; pp. 217–235.

[2]Theyazn H. H Aldhyani, Mohammed Al-Yaari, Hasan Alkahtani, Mashael Maashi, "Water Quality Prediction Using Artificial Intelligence Algorithms", Applied Bionics and Biomechanics, vol. 2020, Article ID 6659314, 12 pages, 2020. https://doi.org/10.1155/2020/6659314

[3]Hmoud Al-Adhaileh, M.;Waselallah Alsaade, F. Modelling and Prediction of Water Quality by Using Artificial Intelligence. Sustainability 2021, 13, 4259. https://doi.org/10.3390/su13084259