Project

Water Potability Assessment Using Machine Learning Techniques.

MEEG667-011: Digital Technology for Agriculture and Natural Resources

By: Venkata Siva Sai Prathyush Kolli

Student Id: 702706968

Under Guidance of Professor: Yin bao

1. Introduction

Access to clean secure drinking water is the number one necessity for both human life and health. On the other hand, in many parts of the globe, it is contaminated and does not meet the health standards set. Water quality assessment was previously done by gathering water samples from samplers and testing them in a lab. This is a process, which is time consuming, capital intensive and limited in its capacity to observe water sources in real time and on a permanent basis. Along with the increasing need of more detailed water quality data, there is a growing demand for more complex and appropriate monitoring solutions.

This project shows an exclusive way of water quality assessment via machine learning technology that automatically judges the water sources as drinkable or not through the sensors which are analyzing the water. Machine learning provides an advantage when used for predicting water quality in that it is fast, cheap, and highly scalable technique. Sophisticated machine learning models, such as random forests, support vector machines, and neural networks stand out because they can identify and analyze patterns in big data. They can be utilized for various water quality tests based on the data collected. Through the collaboration of machine learning with real-time data procurement systems, we can show the existing pollution and also the tendency for future water quality issues to escalate into public health problems can be anticipated.

The aim is to create a machine learning model that can be used to classify potable versus non-potable water based on chemical data from the measurements taken from the 3000 different water bodies. That is the very goal of this new technology to make the process of whether water is safe to drink or not much quicker. Implementation of this solution can potentially transform how water quality is monitored from small communities to massive cities. Below are the main features from the dataset where the target result is based on.

pH Value: pH is one of important water quality parameters to determine the acid-base state of water. It is whether water is acidic or alkaline, depending upon its acid or alkali state. According to WHO, a pH ranging from 6.5 to 8.5 is safe for use in drinking water.

Hardness: Aside from dissolved gases, water hardness mostly comes from the dissolution of these salts from geological deposits into the water. The older the water, the longer it has been in contact with these materials, the softer it will be. This traditional view of the means by which water precipitates soap was based on its reaction with calcium and magnesium ions.

Solids (Total Dissolved Solids – TDS): TDS is the overall quantity of inorganic and some organic compounds dissolved in water including salts such as potassium, sodium and bicarbonates, chlorides, magnesium and sulphates . These substances are mainly incensing the water aesthetics. The higher TPD values propose that water is rich in minerals and hence the water is not good for drinking and needs to be treated before drinking.

Chloramines: Chloramines are applied worldwide as a disinfectant. They are made when amonia is added to chlorine during the water treatment. Maximal level of chlorine determined for drinking water is 4 mg/L. Chloramines are used as a replacement of some harmful by-products formed when chlorine is used only.

Sulphate: Sulphates are present in a wide range of environmental media and chemicals are widely used by the chemical industries. There are high amount of sulphate measures in sea water, held around 2700mg/L, and the amount in fresh water ranges from 3-30mg/L.

Conductivity: Volatile salts / ions, most of them alkaline metal like Na, K, Mg, Ca, release these ions to the aquifers when attacked by any kind of organic acid, making water a conductor of electric current. Pure water is, however, a poor electrical conductor. The higher the conductivity of drinking water the lower the water quality. The standard set by WHO of 400μS/cm makes it clear that drinking water.

Organic Carbon: Total Organic Carbon is usually from natural organic matter and artificial sources. It measures the amount of carbon in organic matter in water. According to the US EPA standard, the treated drinking water should have less than 2 mg/L TOC and source water which is treated before consumption should have less than 4 mg/L TOC.

Trihalomethanes: THMs are the disinfection by-products created when chlorine used in water treatment reacts with naturally occurring organic matter. The amount of THMs can increase with the presence of more organic mater, more chlorine, and warmer water. THM concentrations up to 80 ppm are acceptable in potable water.

Turbidity: Turbidity is the cloudiness or haziness of water, resulting from the presence of suspended solid matter. Not only does it impacts the aesthetics of water, but it also prevents the effective disinfection also. The present study reported that its mean turbidity value is 0.98 NTU that is much lower (clearer water) than the WHO criterion of 5.00 NTU.

Potability: Flag value to signal drinkable or no-drinkable water. If the water is not drinkable, then the test result is indicated as ‘0’, and otherwise the result is ‘1’. This binary output considerably saves human labor readjusting messy test reading results and redirects water safety issues to public health care. Tip: The output ‘0’ is often referred to as ‘NOT’, ‘FAILURE’ or ‘ERROR’.

2. Objectives

The primary purpose of this project was to perfect the machine learning models to classify water potability. The main objective is to come up with a system that would be foundational in providing accurate readings to facilitate quick and data-driven decision making system.

- **Data Preprocessing and Cleaning**: Make sure that the water quality data have been properly cleaned and are ready to be analyzed. This linked the process of dealing with missing data, outliers and noise to create a data set which can be used for model training and testing.
- **Exploring Machine Learning Algorithms:** Research and assess appropriate machine learning algorithms aimed at pinpointing the optimal method of classifying water as potable or not. In this research, analysis of both linear and nonlinear approaches was performed to know what distinguishes one from another regarding provided dataset.
- **Training and testing of Machine Learning Models**: Lead comprehensive training sessions with the chosen machine learning models. I needed to implement the models and test their accuracy on a variety of indices to confirm the unseen data is well supported effectively.

- **Optimization of Machine Learning Models**: Hyper parameter optimization together with cross-validation can be used for a fine-tuning of the selected models. The objective was to improve the accuracy and the capabilities of the models in predicting the safety of consuming water.
- **Implement a user-friendly visualization interface to display the results and insights derived from the model**: Design visualization tools that easily interpret and illustrate the results of water potability evaluation. This objective specifically concerned on the making of the data transparent and understandable.

3. Materials and Methods

1. **Computational Environment**: Google Colab with T4 GPU is used as the primary computing platform. It is based on the cloud service that offers free CPU and GPU resources that are essential for handling complex computations of big data and advanced deep learning models. This environment offers python programming that works well with Google Drive and makes it easy to open datasets and scripts.

2. **Data Preprocessing**: Data cleaning and alignment is a very crucial step. This is done to prepare the data for machine learning tasks.

a. **Handling Missing Values**: The features that commonly contain missing values in the data are pH, sulfate and organic carbon. In order to preserve the integrity of data, missing values are imputed using the mean of respective columns.

Data sample with missing values.

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637	66.420093	3.055934	0
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	100.341674	4.628771	0
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0
5	5.584087	188.313324	28748.687739	7.544869	326.678363	280.467916	8.399735	54.917862	2.559708	0
6	10.223862	248.071735	28749.716544	7.513408	393.663396	283.651634	13.789695	84.603556	2.672989	0

Fig1

Percentile of missing values in dataset of each feature.

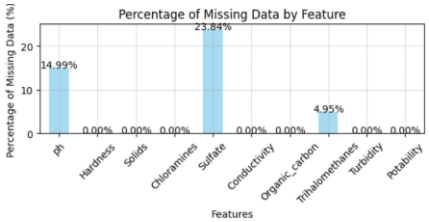


Fig2

The bar chart above demonstrates the quantity of incomplete data in the different attributes of a data set with 3200 data points. The majority of features are complete with just some missing

values for "ph" and "Chloramines" with a combined total of 14. 99% and 23. 84% missing data, respectively. This implies that these features are missing a large chunk of data or information, even partially, calling for techniques like imputation or removal of the data for further analysis. The presence of data gaps leads to the chance of having bias in your models or losing the accuracy of statistical analysis.

b. **Outlier Management**: The outlying points were removed by utilizing the Winorization technique (a statistical procedure that replaces extreme outliers in group by percentile extreme values so that extreme values are outside 5th and 85th percentile). In this way, the influence of the disputed data was significantly lowered without completely eliminating those points altogether. The bar chart below depicts the level of outliers present in each feature of the dataset, expressed as a percentage.

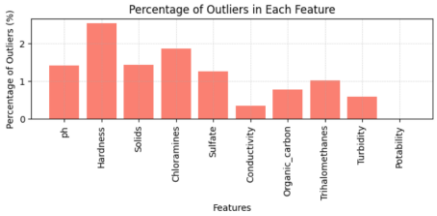


Fig3

It highlights that features like 'ph' and 'Hardness' have a higher chance of outlier values compared to other features of data. From the two images (fig4 and fig5) showing pH value, we can see the effect of outliers. Fig 4 displays outliers at the edge of the pH scale, while Fig 5 shows a more data without outliers.

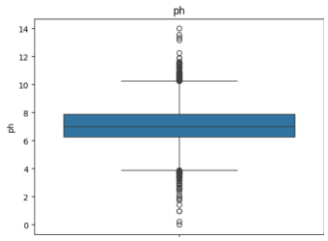


Fig4

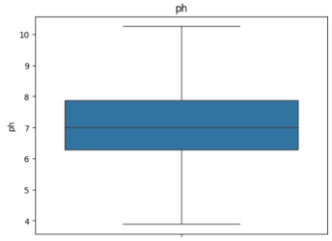


Fig5

The proper handling of outliers is essential for precise data interpretation since they can skew statistical measures and analysis.

c. **Feature Scaling**: It is important for this dataset to standardize variables like Solids, sulfates which vary widely in their ranges and units. I have rescaled features with standardization technique, so that they have a mean of 0 and a standard deviation of 1. This normalization was also essential for models that are sensitive to the scale of input data, such as SVM and KNN.

d. **Imbalanced Dataset**: The below figure(fig6) depicts an imbalanced dataset where 61% of instances belongs to class "0" (blue bar) and 39% belongs to class "1" (green bar). Such datasets pose challenges for machine learning models as they tend to be biased towards non-potable class, which in turn results in poor performance in the potable class yielding lower accuracy.

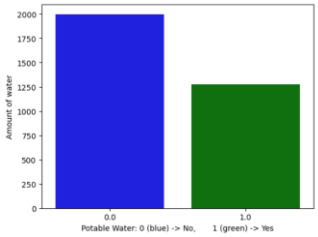


Fig6

Then I tried to balance the classes using SMOTE (Synthetic Minority Over-Sampling Technique); however, my model's accuracy worsened. SMOTE generates synthetic samples of the minority class by interpolating between the minority instances and their nearest neighbours. Despite using SMOTE, I didn't use it in my final solution apparently; probably because of the noise introduced or because the synthetic samples used aren't realistic enough to improve the performance accuracy of my models.

Correlation Matrix:

A correlation matrix(fig7) is a frequently used visualization of correlations between the elementary features of a dataset. You can see this dataset for your own analysis, where don't forget to follow the learning steps below. Most of the cells have small distances to zero, which means that the features of the dataset are practically non-linearly correlated. For example, features such as pH and Hardness (very weak positive linear relationship), Sulfates and Solids (very weak negative linear relationship).

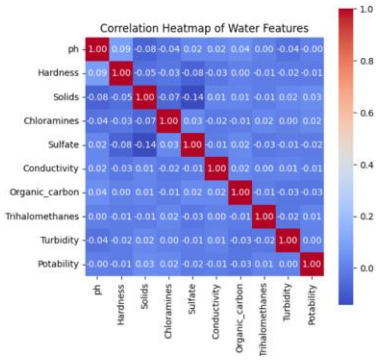


Fig7

Correlation matrix says that there is no linear relationship between the features that can explain the target variable. So, Linear model may not work in this case. Therefore, we should try with probability based models.

4. Model Development and Evaluation

A variety of machine learning models were explored to identify the optimal approach for classifying water as potable or non-potable. I have considered X, Y as features and target data, where X = water.drop(['Potability'], axis=1) and Y = water['Potability']. After experimenting with various data splits, I observed that an 80-20 split provided the best accurate model. The 80-20 split consistently yielded the best results, ensuring robust model performance. The models used in this project are:

a. **Logistic Regression**

A linear model used for binary classification tasks, which is straightforward to implement and interpret. Even though it's clear from the correlation matrix that linear model doesn't suit for the dataset, I have tried with it.

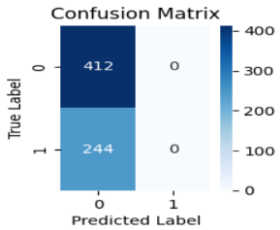
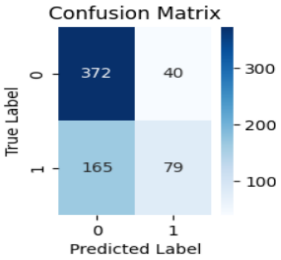
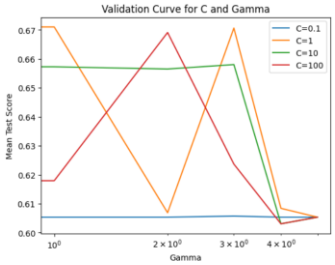


Fig8

By viewing the confusion matrix(Fig8), I have come to understand that the model doesn't fit. As the confusion matrix shows a significant number of false negatives (244 cases incorrectly predicted as non-potable that are actually potable) and no false positives, this result highlights a potential area of improvement for increasing model sensitivity to detect potable class or data.

b. **Support Vector Machine (SVM):** An effective classifier that works well for clear margin of separation and high dimensional spaces. The confusion matrix shows (Fig9) a relatively high number of false negatives compared to true positives, which suggests that the model is more conservative, erring on the side of predicting non-potability.





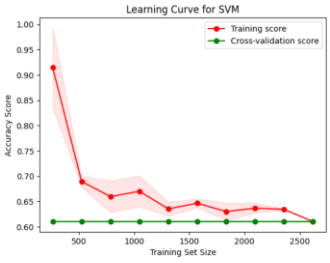


Fig9 Fig10 Fig11

Performance Metrics: (Fig10) Best SVM Parameters: {'C': 1, 'gamma': 'scale'} suggests that a relatively simple model without aggressive regularization (C=1) and scaling for the gamma parameter is done by getting the accuracy score for different values of C and gamma for this dataset as shown. Cross-validation Score of 0.67 and Test Accuracy of 68.75% suggest moderate model performance with the model accurately predicting the potability status of about two-thirds of the test samples. Cross-validation Score of 0.61 from external validation aligns closely with the test performance, indicating the model's generalizability.

Learning Curve Analysis: (Fig11) The learning curve indicates a sharp initial accuracy on the training set, which rapidly declines as more data is introduced, and then stabilizes. The cross-validation score starts much lower and gradually converges towards the training score, although it remains consistently lower throughout. This pattern suggests that the model might be experiencing some overfitting with smaller datasets, which becomes less pronounced as more training data is used. The SVM model demonstrates a fair level of effectiveness with the given dataset, managing a balance between overfitting and underfitting as seen in the learning curve.

c. **Decision Trees**: Simple to understand and interpret. Decision trees mimic human decision-making more closely than other models. For the best result the 3 tunable parameters have been analyzed in the below accuracy graph figure Fig12 to see accuracy and have selected (max_depth 4, min_samples_leaf 50, min_split_samples 90)

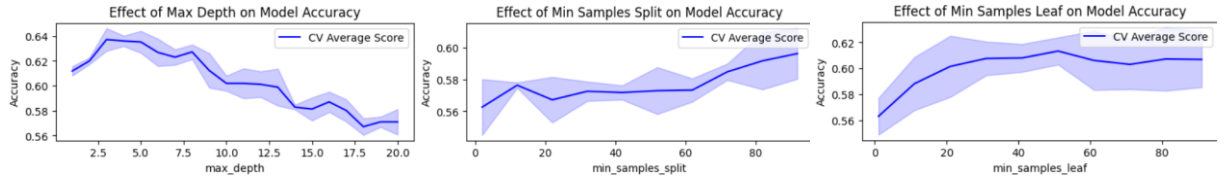


Fig12

According to confusion Matrix analysis (Fig13) very few actual potable cases were accurately identified, which is concerning for practical applications where missing a potable classification could be critical.

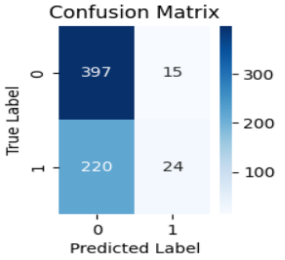
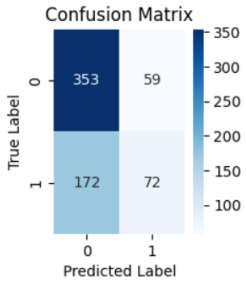


Fig 13 Fig14

Learning Curve Interpretation(Fig14): The curve shows that the training score is very high and remains flat, suggesting that the model is perfectly fitting the training data. Cross-validation curve starts significantly lower and displays some fluctuation as training data increases, but it generally remains much lower than the training score. This pattern indicates a clear overfitting issue where the model learns the training data too well but fails to generalize effectively to unseen data. Performance Metrics: Best Parameters: {'max_depth': 2, 'min_samples_leaf': 90, 'min_samples_split': 2}. These parameters indicate a simple decision tree model, suggesting that the model is under-complex for the dataset.

CV Score (0.62) and Accuracy (0.64) both scores are aligned and confirm the model's moderate overall performance. The relatively close scores indicate that the model's performance is somewhat consistent across different test folds and the final test set. The Decision Tree model, as currently configured, shows underperformance, particularly in detecting the potable class.

d. **K-Nearest Neighbors** (KNN): A non-parametric method that performs well on classification problems based on similarity measures between data points. We have concluded the best parameters as {'metric': 'manhattan', 'n_neighbors': 15, 'weights': 'distance'} indicate that a Manhattan distance metric with 15 neighbors and weighing by distance gives the best results during cross-validation after profound trail and test.



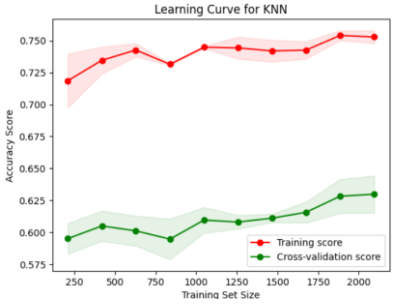


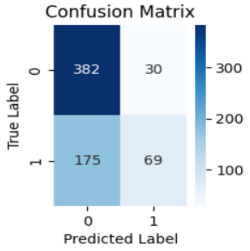
Fig15 Fig16

As per confusion matrix(Fig15) Analysis the model identified 72 cases of potability accurately, which is relatively low compared to the false negatives, suggesting issues with sensitivity of data.

Learning Curve Interpretation: (Fig16) The training score starts high and remains relatively stable as the number of training instances increases. This consistency suggests that the model can learn effectively from the training data given the complexity controlled by the parameters. Cross-validation score starts much lower than the training score, indicating potential overfitting with smaller sets of training data. However, it improves as more data is included, suggesting that increasing the dataset size could help improve model generalization.

CV Score (0.63) and Accuracy (0.65) both scores reflect model performance. The close alignment of CV score and accuracy suggests consistent performance across different subsets of the dataset and the test set.

e. **Random Forest**: An ensemble technique that improves on the simplicity of decision trees by adding robustness and reducing the risk of overfitting. After number of iterations, parameters are {'max_depth': 10, 'min_samples_leaf': 1, 'min_samples_split': 2, 'n_estimators': 100} which suggest a complex model that is tuned to balance bias and variance effectively, given the dataset. According to the confusion matrix (Fig17) the model has identified a modest number of potable cases accurately but clearly shows room for improvement in sensitivity.



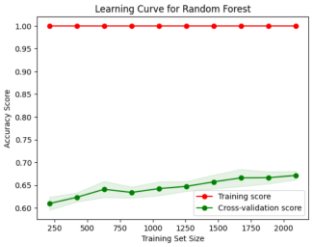


Fig17 Fig18

Learning Curve Interpretation (Fig18): The training score is very high and flat, which indicates that the model can memorize or fit very well to the training data. This is typical for Random Forest models, which are robust and less prone to overfitting due to their ensemble nature. The cross-validation score is considerably lower than the training score, initially, but it gradually improves as more training data is used. This pattern suggests that while the model can potentially overfit to smaller training sets, it generalizes better with more data. However, the persistent gap

between the training and validation scores might indicate the model's limitations in generalizing beyond the training data.

CV Score (0.670992) and Accuracy (0.687500) these scores are relatively close and confirm the model's ability to perform consistently across different subsets of the dataset and in the test set, respectively. The Random Forest model demonstrates solid performance with certain strengths, such as high accuracy in identifying non-potable samples and robustness against overfitting.

f. **XGBoost**: An implementation of gradient boosted decision trees designed for speed and performance, which is particularly effective for structured data. After the visualizations of the graphs below figure19 of tuning parameters with the accuracy, I have got the highest accuracy with best parameter values: No of estimators 80, learning rate 0.1 and max depth 9.

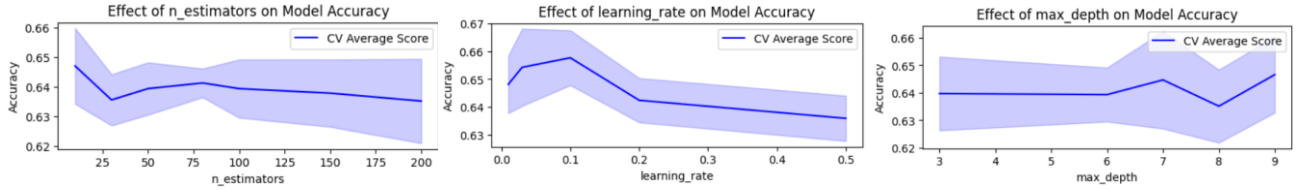


Fig19

According to the confusion matrix (Fig20), the model accurately predicted the potable class 100 times. This confusion matrix suggests a moderate level of performance with a substantial number of false negatives, indicating the model might be erring on the side of predicting the majority class (non-potable). This could be due to class imbalance of dataset.

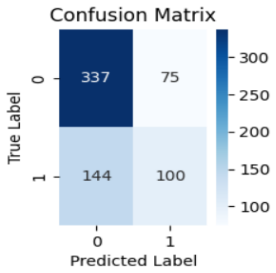


Fig20

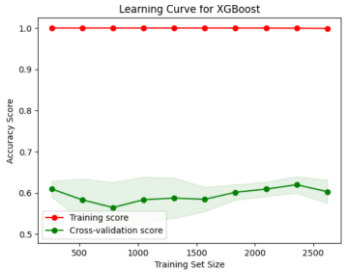


Fig21

Learning Curve (Fig21) shows a high training score (red line) that remains consistent across the training set sizes, suggesting that the model fits the training data well. The cross-validation score (green line) is significantly lower than the training score at smaller training sizes indicating potential overfitting. However, as the training size increases, the cross-validation score increases, suggesting that adding more data helps the model generalize better, but there's still a notable gap between training and validation performance.

CV Score 0.657630 and Accuracy 0.666159 both metrics indicate the model's generalization ability. The CV Score derived from cross-validation helps mitigate the impact of the random chance in the train-test split, offering a more robust estimate of model performance. These scores, while moderate, suggest that the model performs uniformly across different data subsets.

g. **Neural Networks**: Capable of modeling complex nonlinear relationships through layers of neurons, offering flexibility and power in modeling diverse datasets. According to the confusion

matrix Fig22, it shows 298 true negatives and 127 true positives indicating the instances accurately predicted as non-potable and potable respectively.

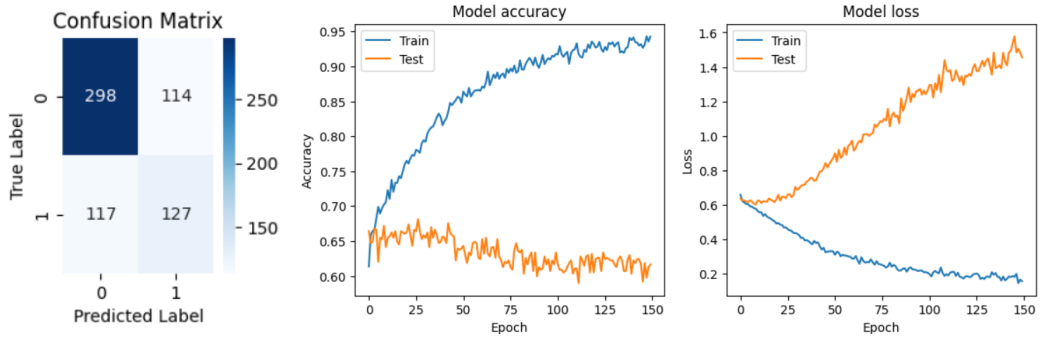
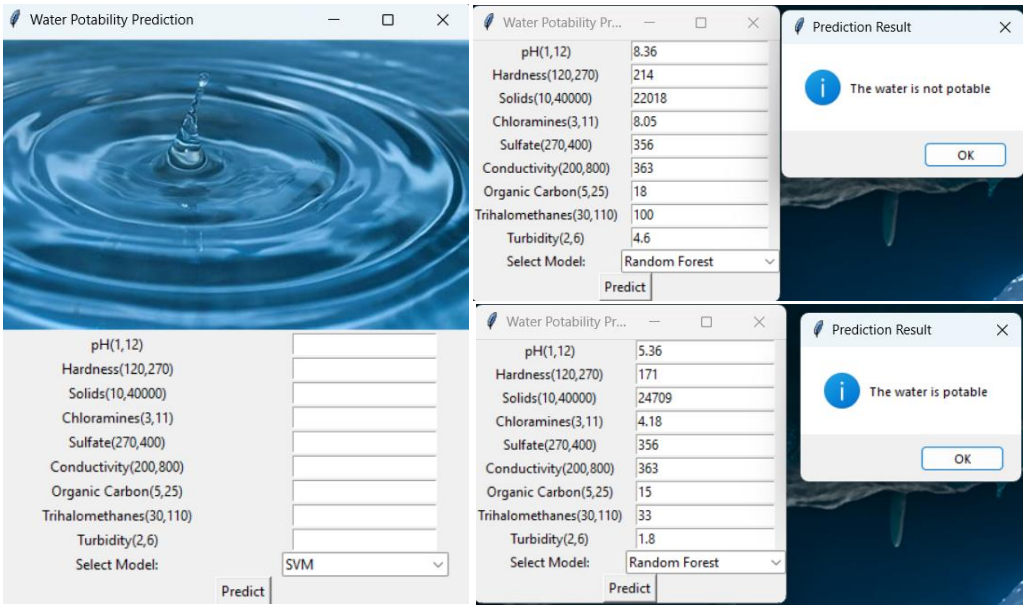


Fig22 Fig23

The accuracy plot Fig23 shows the training accuracy increasing steadily and leveling off close to 95%, while the test accuracy is significantly more volatile and peaks at around 70%. This discrepancy suggests overfitting, where the model learns the training data well but generalizes poorly to new, unseen data. The loss plot further emphasizes this point, where the training loss decreases close to zero, indicating good performance on training data. In contrast, the test loss increases significantly after around 25 epochs, further reinforcing the overfitting issue.

5. System Integration and Visualization

The best-performing model was integrated into a Python-based system using the Tkinter library for developing the graphical user interface (GUI). This interface allows users to input new data and receive predictions on water potability. Additionally, visualization tools were implemented to graphically represent the results, making them accessible and easy to understand for all users. This methodological approach not only provided a thorough understanding of various machine learning models but also ensured the practical application of these models through a user-friendly system, demonstrating the project's commitment to applying advanced analytics to solve real-world problems effectively.



6. Results

In evaluating machine learning models for water potability classification, several metrics are crucial. RMSE and MAE gauge prediction errors, with lower values indicating more accurate predictions. The F1 Score assesses the balance between precision and recall, where higher scores reflect better model performance. Variance measures a model's sensitivity to dataset fluctuations, where lower values denote stability. The Cross-Validation Score evaluates the model's ability to generalize, with higher scores preferred. Finally, Accuracy measures the overall correct predictions with higher values showing superior model performance. These metrics collectively guide the optimal model selection for practical application.

Model	Rmse	Mae	F1 Score	Variance	CV Score	Accuracy
Logistic Regression	0.609	0.372	0.48	0.01	0.61	0.62
SVM	0.559	0.3125	0.6542	0.14	0.61	0.67
XGBoost	0.58	0.63	0.65	0.11	0.65	0.66
KNN	0.59	0.35	0.61	0.15	0.63	0.64
Decision Tree	0.59	0.35	0.54	0.05	0.62	0.64
Random Forest	0.56	0.312	0.64	0.12	0.67	0.68
Neural network	0.59	0.35	0.64	0.23	0.66	0.67

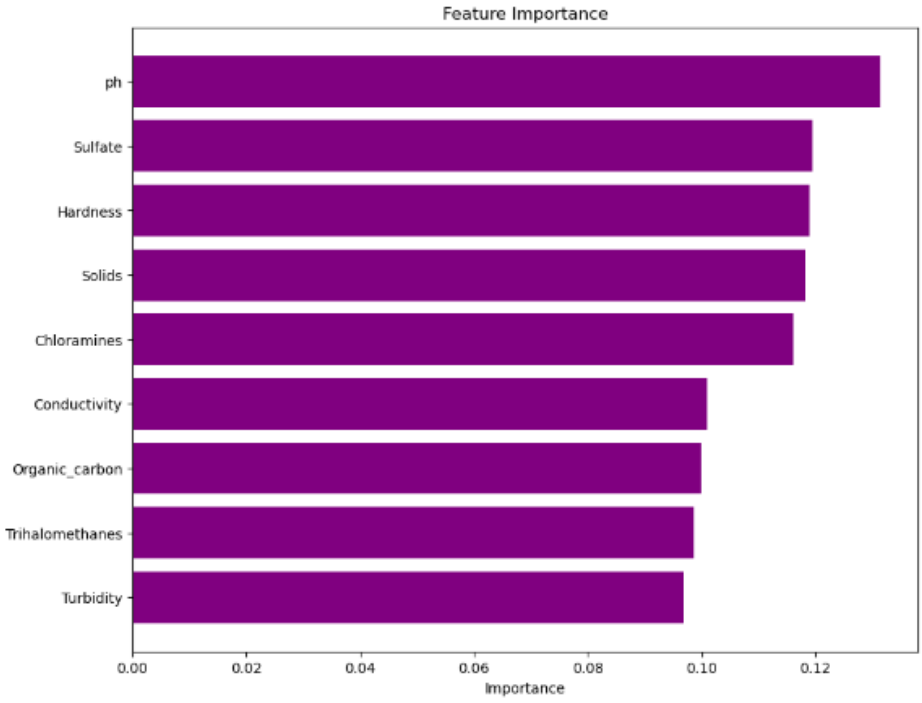
Among the six machine learning models evaluated for water potability classification, the Random Forest model exhibited the best overall performance based on the metrics considered. It achieved the lowest RMSE of 0.56 and the lowest MAE of 0.312, indicating highly accurate predictions with minimal error. Additionally, the Random Forest model scored an F1 Score of 0.64, showcasing a strong balance between precision and recall. It also demonstrated good stability with a variance of 0.12, suggesting less sensitivity to fluctuations in the dataset compared to other models. Notably, it achieved the highest Cross-Validation Score of 0.67 and the highest accuracy of 68%.

These results underscore the Random Forest model's superior capability in effectively classifying water potability, making it the most reliable choice among the models tested.

7. Discussion

The findings highlight the effectiveness of using machine learning for water quality analysis. The Random Forest model outperformed others likely due to its ability to handle non-linear relationships and interactions between features. However, the models exhibited some overfitting, suggesting a need for more balanced data. Future work could explore more sophisticated ensemble techniques and deeper neural networks to improve model generalizability.

Feature Importance.



A horizontal bar chart representing the feature importance scores derived from a Random Forest classifier. Feature importance is a measure of the contribution of each feature in making predictions in a machine learning model. In this chart, 'pH' has the highest importance score, indicating it is the most significant feature in the model's predictive capability. Following 'pH', 'Sulfate' and 'Hardness' also have high importance scores, showing they are crucial for the model's performance. Other features such as 'Solids', 'Chloramines', 'Conductivity', 'Organic_carbon', 'Trihalomethanes', and 'Turbidity' have progressively lower importance scores, suggesting they are relatively lesser but still relevant contribution to the model. Each bar's length corresponds to the importance score of the respective feature, with longer bars indicating higher importance. The uniform purple color provides a clear visual distinction, making it easy to compare the relative importance of the features at a glance.

Explanation of terms.

1. RMSE(Root Mean Square Error): Measures the average magnitude of the errors between predicted and actual values.

2. MAE (Mean Absolute Error): Represents the average absolute difference between predicted and actual values.

3. F1 Score: Score of the harmonic mean of precision and recall.

4. Variance: Indicates the dispersion of the prediction errors around the mean prediction.

5. CV (Coefficient of Variation) Score: The ratio of the standard deviation to the mean, used to measure relative variability.

6. Accuracy: The ratio of correctly predicted observations to the total observations.

Effects on the model:

RMSE and MAE: Lower values indicate better model performance in terms of prediction accuracy.

F1 Score: Higher values indicate better model performance in handling imbalanced classes.

Variance: Lower variance suggests more consistent predictions.

CV Score: Lower CV indicates more stability in model performance.

Accuracy: Higher accuracy means the model is correctly predicting more observations.

8. Conclusion

In this project, the Random Forest classifier emerged as the best-performing model, effectively handling the data provided. Despite potential issues of data imbalance and the need for more positive result data, all models demonstrated similar accuracies, indicating that the constructed models performed well with the available dataset. The feature importance analysis revealed that chemical properties, particularly pH and sulfate levels, play a critical role in the model's predictions. Additionally, factors such as hardness, solids, and disinfection significantly contribute to the model's performance. Overall, the models constructed for this study are robust and reliable, effectively leveraging the data to make accurate predictions. Future improvements could focus on addressing data imbalance and expanding the dataset to enhance model performance further.

This project confirms the potential of machine learning in environmental monitoring and public health protection. The developed system can classify water potability with reasonable accuracy, providing a valuable tool for regions lacking comprehensive water quality monitoring infrastructure. The approach also offers a blueprint for similar applications in other public health and safety domains.

9. Future Work

1. Data Collection: Increase dataset size and balance, especially by adding more positive result instances to improve model robustness.

2. Training Neural Networks: Implement and train advanced neural networks, such as CNNs or LSTMs, to enhance accuracy and model performance.

3. Model Optimization: Fine-tune model hyper parameters using techniques like grid search or randomized search to achieve optimal performance.

4. Overfitting Prevention: Apply strategies like cross-validation, regularization, and dropout to ensure models generalize well and avoid overfitting.

5. Application Development: Create a user-friendly Tkinter application for easy data input, model training, and prediction visualization, making the tool accessible to non-technical users.

Appendices A

- Code:

```
#Import Required Libraries

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import xgboost as xgb

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn import preprocessing

from sklearn import metrics

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive_bayes import GaussianNB

from sklearn.linear_model import SGDClassifier

from sklearn.linear_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.preprocessing import StandardScaler,MinMaxScaler,PowerTransformer

from sklearn.metrics import classification_report, accuracy_score

from sklearn import svm

from sklearn.linear_model import LinearRegression

from sklearn.model_selection import KFold
```

```
from sklearn import datasets
from sklearn.metrics import confusion_matrix
from sklearn import tree
import warnings
warnings.filterwarnings("ignore")
water=pd.read_csv("water_potability.csv")
#water.info()
#water.describe()
water.head(10)
# Calculate the percentage of missing water in each column
missing_percentages = water.isnull().mean() * 100
# Plotting the percentage of missing water for each feature
plt.figure(figsize=(7, 2))  # Increase figure size for better readability
ax = missing_percentages.plot(kind='bar', color='skyblue', alpha=0.75)
plt.title('Percentage of Missing water by Feature')
plt.xlabel('Features')
plt.ylabel('Percentage of Missing water (%)')
plt.grid(True, which='both', linestyle='--', linewidth=0.5)
plt.xticks(rotation=45)
for p in ax.patches:
width = p.get_width()
height = p.get_height()
x, y = p.get_xy()
ax.annotate(f'{height:.2f}%', (x + width/2, y + height), ha='center')
plt.show()
# Function to calculate outliers based on the IQR method
def calculate_outliers(df, feature):
Q1 = df[feature].quantile(0.25)
Q3 = df[feature].quantile(0.75)
IQR = Q3 - Q1
```

```
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR
outliers = df[(df[feature] < lower_bound) | (df[feature] > upper_bound)]
return len(outliers)
# Dictionary to hold the percentage of outliers per feature
outliers_percentage = {}
# Calculate percentage of outliers for each feature
for column in water.columns:
num_outliers = calculate_outliers(water, column)
total_entries = len(water[column])
percentage_outliers = (num_outliers / total_entries) * 100
outliers_percentage[column] = percentage_outliers
# Creating a bar chart
plt.figure(figsize=(8, 2))
plt.bar(outliers_percentage.keys(), outliers_percentage.values(), color='salmon')
plt.title('Percentage of Outliers in Each Feature')
plt.ylabel('Percentage of Outliers (%)')
plt.xticks(rotation=90)  # Rotate feature names for better visibility
plt.grid(True, which='both', linestyle='--', linewidth=0.5, alpha=0.6)
#taking mean
print("ph mean:",water["ph"].mean())
print("Sulfate mean:",water["Sulfate"].mean())
print("Trihalomethanes mean:", water["Trihalomethanes"].mean())
#add null values
water['ph'] = water['ph'].fillna(7.0)
water['Sulfate'] = water['Sulfate'].fillna(333.7)
water['Trihalomethanes'] = water['Trihalomethanes'].fillna(66.3)
#outliners
for i in water.columns:
sns.boxplot(water[i])
```

```
plt.title(i)

def outlinefree(water):

Q1, Q3 = np.percentile(water, [25, 75])

return Q1 - 1.5 * IQR, Q3 + 1.5 * IQR

# List of columns to process for outliers, now including 'Potability'

columns_to_process = ['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Conductivity', 'Organic_carbon', 'Trihalomethanes', 'Turbidity', 'Potability']

for col in columns_to_process:

low, up = outlinefree(water[col])

water[col] = np.where(water[col] < low, low, np.where(water[col] > up, up, water[col]))

#saved preprocessed dataset as water_sample.csv

from sklearn.preprocessing import MinMaxScaler

# Loading the dataset

water = pd.read_csv('water_sample.csv')

# Separating the features and the target variable

# Assuming 'Potability' is the target variable

X = water.drop('Potability', axis=1)  # This line will drop the target column from the dataset to isolate features

Y = water['Potability']  # This isolates the target variable

# Initialize MinMaxScaler

scaler = MinMaxScaler()

# Fit and transform the feature data

X_scaled = scaler.fit_transform(X)

# Convert the scaled data back to a DataFrame

X_scaled_df = pd.DataFrame(X_scaled, columns=X.columns)

# Show the first few rows of the scaled DataFrame

print(X_scaled_df.head(20))

#heatmap

data = pd.read_csv('water_sample.csv')

# Calculate the correlation matrix
```

```
corr = data.corr()
# Plotting the heatmap
plt.figure(figsize=(6, 6))  # Set figure size for better readability
sns.heatmap(corr, annot=True, fmt=".2f", cmap='coolwarm', cbar=True, square=True)
plt.title('Correlation Heatmap of Water Features')
#logistic regression
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import mean_squared_error, mean_absolute_error, f1_score
from sklearn.model_selection import learning_curve, validation_curve
# Load your dataset
X = data.drop('Potability', axis=1)
y = data['Potability']
# Scaling features using MinMaxScaler as shown in your provided code
scaler = StandardScaler()
X = pd.DataFrame(X_scaled, columns=X.columns)
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Initialize and train the Logistic Regression model
lr_model = LogisticRegression(max_iter=100, random_state=1)
lr_model.fit(X_train, y_train)
# Predicting the test results
y_pred = lr_model.predict(X_test)
# Metrics Calculation
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
mae = mean_absolute_error(y_test, y_pred)
f1 = f1_score(y_test, y_pred, average='weighted')
cv_scores = cross_val_score(lr_model, X, y, cv=5)
accuracy = accuracy_score(y_test, y_pred)
```

```
variance = np.var(y_pred)
# Metrics Table
metrics_table = pd.DataFrame({
'Metric': ['RMSE', 'MAE', 'F1 Score', 'Variance', 'CV Score (mean)', 'Accuracy'],
'Value': [rmse, mae, f1, variance, cv_scores.mean(), accuracy]})
print(metrics_table)
# Confusion Matrix Plot
plt.figure(figsize=(2, 2.5))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt="d", cmap="Blues")
plt.title('Confusion Matrix')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
# Learning Curve
train_sizes, train_scores, test_scores = learning_curve(lr_model, X, y, cv=5, scoring='accuracy', n_jobs=-1, train_sizes=np.linspace(0.01, 1.0, 50))
train_mean = np.mean(train_scores, axis=1)
train_std = np.std(train_scores, axis=1)
test_mean = np.mean(test_scores, axis=1)
test_std = np.std(test_scores, axis=1)
plt.fill_between(train_sizes, train_mean - train_std, train_mean + train_std, color="r", alpha=0.1)
plt.fill_between(train_sizes, test_mean - test_std, test_mean + test_std, color="g", alpha=0.1)
plt.plot(train_sizes, train_mean, 'o-', color="r", label="Training score")
plt.plot(train_sizes, test_mean, 'o-', color="g", label="Cross-validation score")
plt.title('Learning Curve for Logistic Regression')
plt.xlabel('Training Set Size')
plt.ylabel('Accuracy Score')
plt.legend(loc="best")
#xgboost tuning parameters
from sklearn.model_selection import train_test_split, GridSearchCV
from xgboost import XGBClassifier
```