# The classification of water quality for consumption using Machine Learning

## *1. Introduction*

### *1.1 Background and Significance of the Problem*

In today's era where data and technology play an increasingly vital role, the use of Machine Learning (ML) to enhance and improve human quality of life has become indispensable. One of the critical challenges that the global community faces is ensuring safe and adequate drinking water for everyone. In this context, the classification of water quality for consumption using Machine Learning has become a significant and highly demanded topic.

Research and development in Machine Learning have opened new possibilities in water quality inspection and assessment, using techniques such as image processing, deep data analysis, and composite learning algorithms. The application of Machine Learning in the classification and prediction of water quality not only aids in detecting and preventing the consumption of contaminated water but also enhances the efficiency of water resource management.

This study focuses on exploring and analyzing how Machine Learning can assist in classifying water quality, the effectiveness of various models in predicting water quality, and the potential challenges encountered in this process. The outcomes of this research are expected to lead to the development and application of Machine Learning technologies in more efficiently detecting and improving the quality of drinking water in the future.

This introduction emphasizes the importance of using Machine Learning in improving the quality of drinking water and discusses the research direction and demand in this topic.

### *1.2 Objective*

This research focuses on evaluating water quality by examining and analyzing important parameters related to water quality assessment, to determine whether the water is suitable for drinking. Moreover, this research includes the development and improvement of quality data management processes to be suitable for Machine Learning applications. This encompasses data cleaning, appropriate selection and preparation of data, and improving data quality to enhance the efficiency of Machine Learning models. Finally, this study will apply Machine Learning algorithms to develop accurate and reliable methods for classifying water quality, effectively combining quality data management with Machine Learning technology.

## ***2. Methods***

### *2.1 Data set*

The dataset obtained from the Kaggle website, accessible at https://www.kaggle.com/datasets/adityakadiwal/water-potability,

The dataset contains a total of 3,276 entries and is used to assess the suitability of water for drinking based on various indicators of water quality, such as pH, water hardness, total dissolved solids, chloramines, sulfate, conductivity, organic carbon, trihalomethanes, and turbidity. The 'Potability' column indicates whether the water is safe to drink (1) or not (0).

```
RangeIndex: 3276 entries, 0 to 3275
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ph               2785 non-null   float64
 1   Hardness         3276 non-null   float64
 2   Solids           3276 non-null   float64
 3   Chloramines      3276 non-null   float64
 4   Sulfate          2495 non-null   float64
 5   Conductivity     3276 non-null   float64
 6   Organic_carbon   3276 non-null   float64
 7   Trihalomethanes  3114 non-null   float64
 8   Turbidity        3276 non-null   float64
 9   Potability       3276 non-null   int64
dtypes: float64(9), int64(1)
memory usage: 256.1 KB
None
Shape of the Dataset = (3276, 10)
```

### 2.1.1 Water quality parameters

1. pH: The pH level of the water, indicating its acidity or alkalinity.

2. Hardness: The hardness of the water, indicating the amount of calcium and magnesium minerals.

3. Solids (Total dissolved solids - TDS): The concentration of dissolved substances in water, including minerals, salts, and heavy metals.

4. Chloramines: The amount of chloramines in water, compounds used for water disinfection.

5. Sulfate: The amount of sulfate ions in water, which can come from both natural sources and human activities.

6. Conductivity: The electrical conductivity of the water, indicating the amount of dissolved minerals.

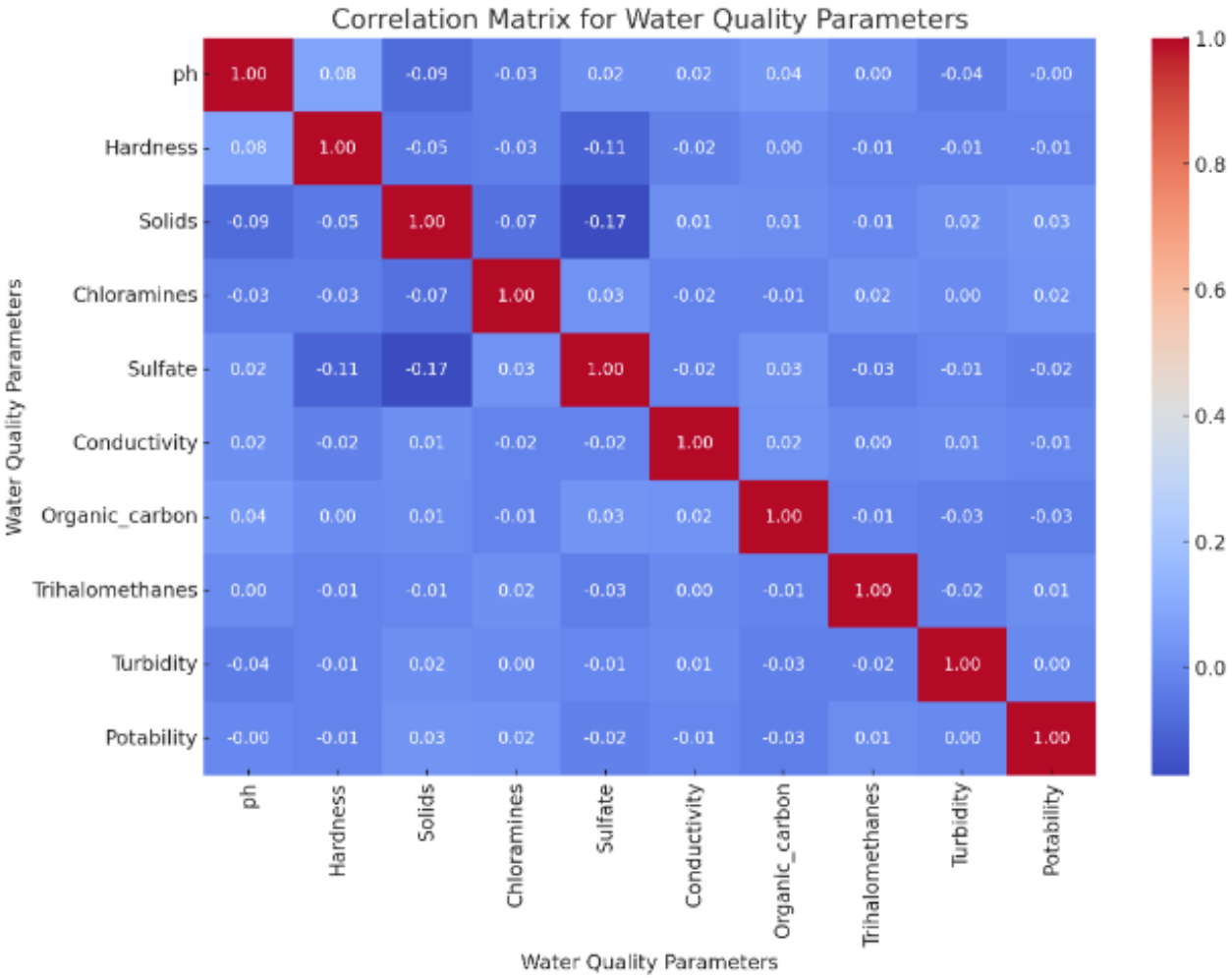
7. Organic carbon: The amount of Total Organic Carbon (TOC) in water.

8. Trihalomethanes: The amount of trihalomethanes, by-products of water disinfection with chlorine.

9. Turbidity: The turbidity level of the water, indicating the amount of suspended particles.

10. Potability: This number indicates whether the water is potable or not (0 means not potable, 1 means potable).

### 2.1.2 Correlation water quality parameters




1. pH: Does not have a strong correlation with any other field; the highest is a weak positive correlation with Organic_carbon (0.0435).

2. Hardness: Similarly shows weak correlations with all other fields, with no significant relationships.

3. Solids: Shows a very weak positive correlation with Potability (0.0337), suggesting that higher solid contents might have a slight association with water being potable.

4. Chloramines: Has a very weak positive correlation with Potability (0.0238) as well.

5. Sulfate: Its strongest correlation is a weak negative one with Solids (-0.1718), indicating that higher levels of solids might be associated with lower sulfate levels.

6. Conductivity: Very weak correlations with all other fields, indicating no strong relationships.

7. Organic_carbon: No significant correlation with other fields, the highest being a weak positive with ph.

8. Trihalomethanes: Weakly correlated with other fields, with no significant relationships.

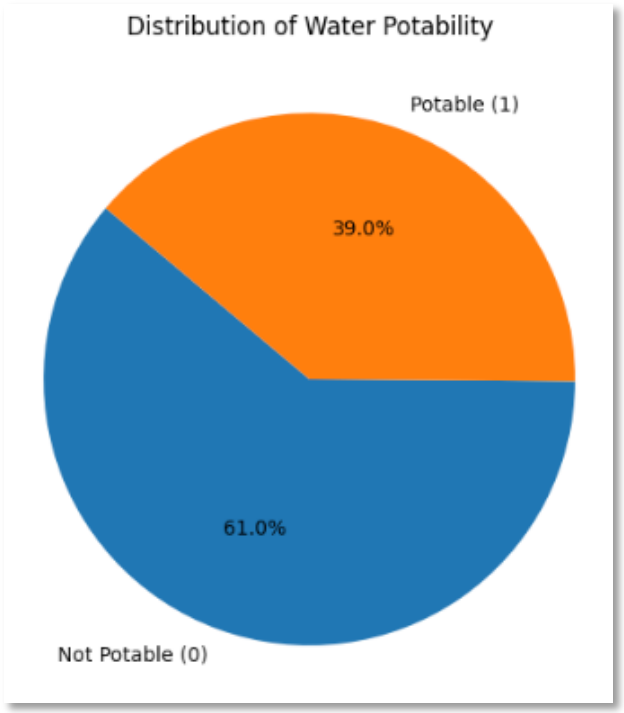
9. Turbidity: Similarly, shows weak correlations with no significant relationships.

10. Potability: This is the target variable, and it has very weak correlations with all other water quality indicators, with the highest correlation being a very weak positive with Solids (0.0337).

These relationships suggest that there are no strong linear correlations between the individual water quality indicators and whether the water is potable or not according to this dataset. It's important to note that correlation does not imply causation, and these weak correlations do not necessarily mean that there are no other types of relationships or that these indicators are not important in determining water potability. Advanced statistical or machine learning methods may reveal more complex relationships.

### 2.1.3 Data balance

A pie chart displaying the distribution of water potability. According to this image:



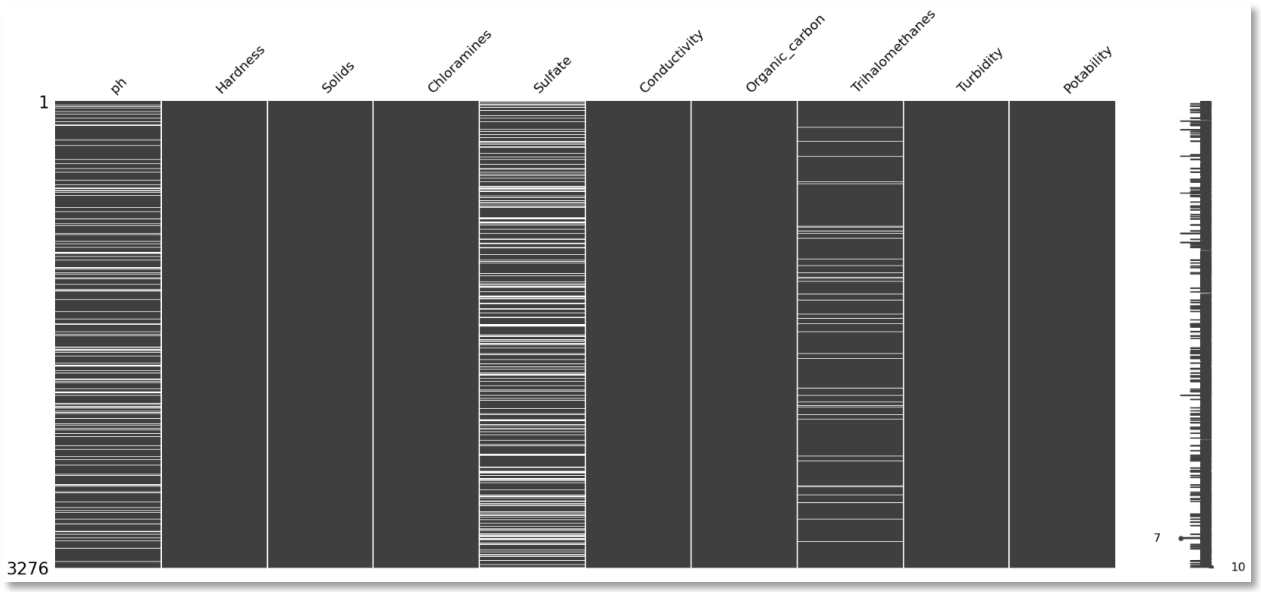

The blue color represents the portion of non-potable water (Not Potable), which accounts for 61.0% of the dataset.

The orange color represents the portion of potable water (Potable), which accounts for 39.0% of the dataset.

This proportion indicates that in this dataset, there is a larger amount of non-potable water compared to potable water. Understanding this distribution of water potability may be important in developing and evaluating models that predict water quality. It also highlights the balance or imbalance in the dataset that may affect the performance of machine learning models.

## *2.2 Quality Data Management*

### 2.2.1 Managing Missing Values




```
ph                491
Hardness            0
Solids              0
Chloramines         0
Sulfate           781
Conductivity        0
Organic_carbon      0
Trihalomethanes   162
Turbidity           0
Potability          0
dtype: int64
```

*Managing Missing Values (Replace missing value with 0)*

```
df.fillna(0, inplace=True)
```

```
ph                0
Hardness          0
Solids            0
Chloramines       0
Sulfate           0
Conductivity      0
Organic_carbon    0
Trihalomethanes   0
Turbidity         0
Potability        0
dtype: int64
```

*Managing Missing Values (Delete all data in a row when missing values are found.)*

Remain 2011 row (Missing 1265 = 38.61%)

Before

```
RangeIndex: 3276 entries, 0 to 3275
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ph               2785 non-null   float64
 1   Hardness         3276 non-null   float64
 2   Solids           3276 non-null   float64
 3   Chloramines      3276 non-null   float64
 4   Sulfate          2495 non-null   float64
 5   Conductivity     3276 non-null   float64
 6   Organic_carbon   3276 non-null   float64
 7   Trihalomethanes  3114 non-null   float64
 8   Turbidity        3276 non-null   float64
 9   Potability       3276 non-null   int64
dtypes: float64(9), int64(1)
memory usage: 256.1 KB
None
```

```
df = df.dropna()
```

After

```
Int64Index: 2011 entries, 3 to 3271
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ph               2011 non-null   float64
 1   Hardness         2011 non-null   float64
 2   Solids           2011 non-null   float64
 3   Chloramines      2011 non-null   float64
 4   Sulfate          2011 non-null   float64
 5   Conductivity     2011 non-null   float64
 6   Organic_carbon   2011 non-null   float64
 7   Trihalomethanes  2011 non-null   float64
 8   Turbidity        2011 non-null   float64
 9   Potability       2011 non-null   int64
dtypes: float64(9), int64(1)
memory usage: 172.8 KB
None
```

*Managing Missing Values ( Replace Missing Value with condition - ph )*

ตารางมาตรฐาน pH น้ำดื่มในประเทศไทย

| พารามิเตอร์ | กรมอนามัย [1] | อย.[2] | สมอ.[3] |
|---|---|---|---|
| คุณภาพน้ำทางกายภาพ/ฟิสิกส์ | | | |
| - ความเป็นกรด-ด่าง(pH) | อยู่ระหว่าง 6.5-8.5 | อยู่ระหว่าง 6.5-8.5 | อยู่ระหว่าง 6.5-8.5 |
| - ความขุ่น (turbidity) | ไม่เกิน 5 NTU | ไม่เกิน 5 ซิลิกาสเกล | ไม่เกิน 5 NTU |
| - สี (Colour) | ไม่เกิน 15 หน่วย แพลทินัม-โคบอลต์ | ไม่เกิน20ฮาเซนยูนิต | ไม่เกิน 5 หน่วย แพลทินัม-โคบอลต์ |
| - กลิ่น | ไม่กำหนด | ต้องไม่มีกลิ่น แต่ไม่รวมถึงกลิ่นคลอรีน | ไม่กำหนด |

(6.8 + 8.5) / 2

```
mean_ph_non_potable = df[df['Potability'] == 0]['ph'].mean()
if ((df['Potability'][x] == 0)) : df['ph'][x] = mean_ph_non_potable
elif ((df['Potability'][x] == 1) ) : df['ph'][x] = 7.5
```

```
Mean ph non potable : 7.0854
```

*Managing Missing Values ( Replace Missing Value with condition - Sulfate )*

```
print('Conditonal Statements to fill in the Missing Values of Sulfate Column')
print("\n")
print('if Potability = 0')
condition_1_mean_sulfate = df[(df['Potability'] == 0)][['Sulfate']].mean()
print("Sulfate : {:.4f}".format(float(condition_1_mean_sulfate)))

print("\n")
print('if Potability = 1')
condition_2_mean_sulfate = df[(df['Potability'] == 1)][['Sulfate']].mean()
print("Sulfate : {:.4f}".format(float(condition_2_mean_sulfate)))
```

```
if Potability = 0
Sulfate : 334.5643


if Potability = 1
Sulfate : 332.5670
```

*Managing Missing Values ( Replace Missing Value with condition - Trihalomethanes)*

```
# Calculate the mean of 'Trihalomethanes' for each group of 'Potability'
mean_trihalomethanes_potable = df[df['Potability'] == 1]['Trihalomethanes'].mean()
mean_trihalomethanes_not_potable = df[df['Potability'] == 0]['Trihalomethanes'].mean()

# Replace missing 'Trihalomethanes' values for the potable group
df.loc[(df['Potability'] == 1) & (df['Trihalomethanes'].isnull()), 'Trihalomethanes'] = mean_trihalomethanes_potable

# Replace missing 'Trihalomethanes' values for the non-potable group
df.loc[(df['Potability'] == 0) & (df['Trihalomethanes'].isnull()), 'Trihalomethanes'] = mean_trihalomethanes_not_potable

# Print the mean values for each 'Potability' group
print('Mean Trihalomethanes for Potable water:', mean_trihalomethanes_potable)
print('Mean Trihalomethanes for Non-Potable water:', mean_trihalomethanes_not_potable)
```

```
Mean Trihalomethanes for Potable water: 66.53968374070116
Mean Trihalomethanes for Non-Potable water: 66.30355527306088
```

### 2.2.2 Managing Outlier

Tukey's Rule is a statistical method used to detect outliers in a dataset. This method utilizes the interquartile range (IQR), which is the difference between the 25th percentile (Q1) and the 75th percentile (Q3), to establish a range for considering potential outliers.

The IQR measures the spread of the middle 50% of the data. Tukey's Rule defines the boundaries for outliers by multiplying the IQR by 1.5 and then adding or subtracting this value from Q1 and Q3 to find the normal range.

According to Tukey's Rule:

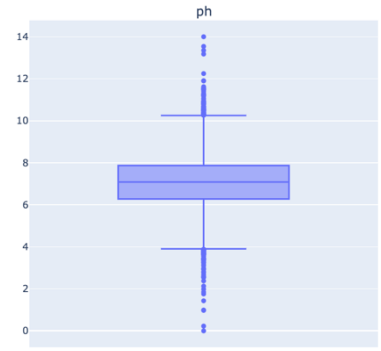
- Any value below Q1 - 1.5 * IQR or
- Any value above Q3 + 1.5 * IQR

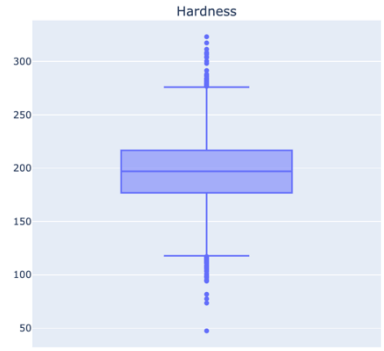
is considered an outlier. This method effectively identifies values that lie outside the main distribution of data and is commonly used in data analysis to decide which values should be considered part of the dataset or excluded.

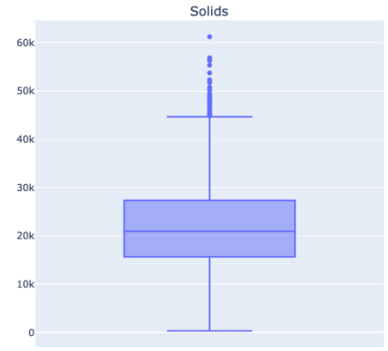
```
Q1 = df[column].quantile(0.25)
Q3 = df[column].quantile(0.75)
IQR = Q3 - Q1
outlier_lower = Q1 - 1.5 * IQR
outlier_upper = Q3 + 1.5 * IQR
```

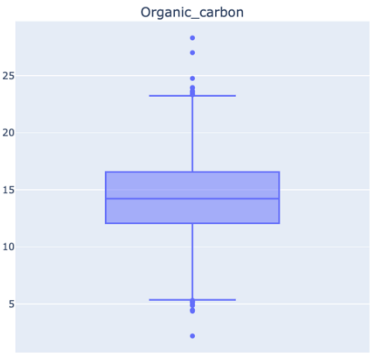
6538135 Suganya Photawin

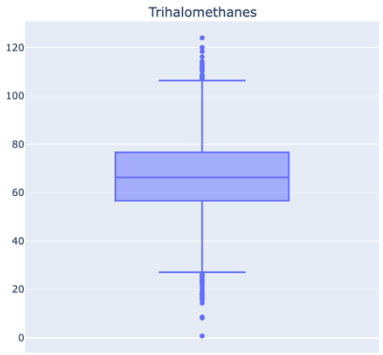
*Outlier over view (Before)*

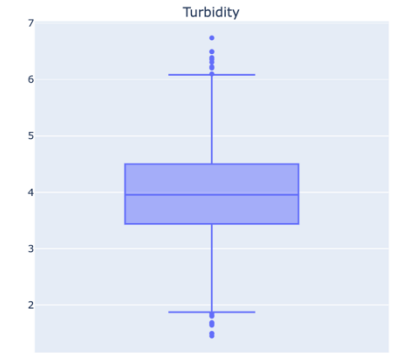


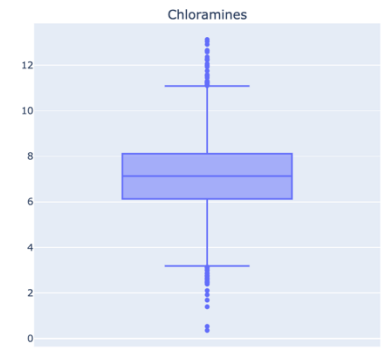



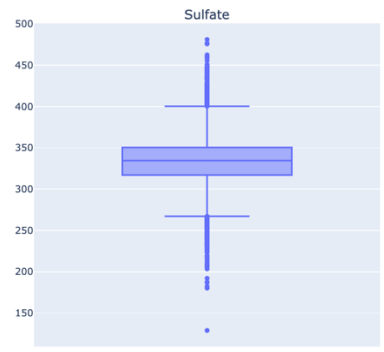



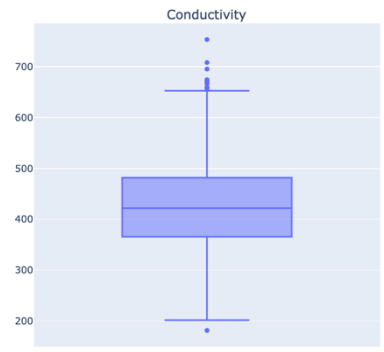



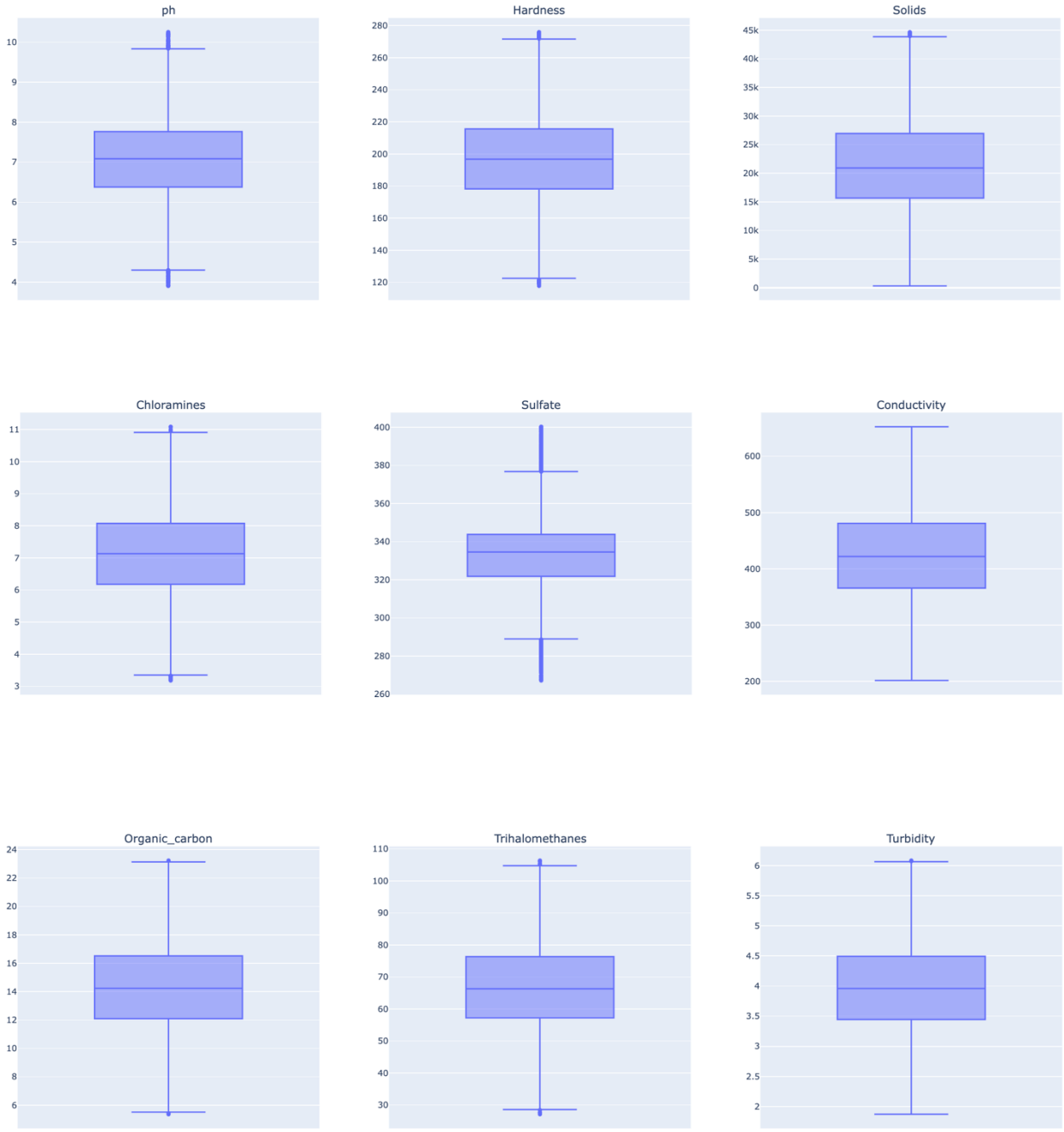



*Outlier over view (After)*



### 2.2.3 Feature Engineering

```
df['Hardness_Sulfate'] = df['Hardness'] * df['Sulfate']

df['Chloramines_OrganicCarbon'] = df['Chloramines'] * df['Organic_carbon']

df['Solids_Sulfate_ratio'] = df.apply(lambda row: row['Solids'] / row['Sulfate']
                                      if row['Sulfate'] > 0 else None, axis=1)
df['ph_Hardness'] = df['ph'] * df['Hardness']
```

| Hardness_Sulfate | Chloramines_OrganicCarbon | Solids_Sulfate_ratio | ph_Hardness |
|---|---|---|---|
| 75505.501517 | 75.774616 | 56.418973 | 1451.726415 |
| 43300.287457 | 100.723120 | 55.684538 | 480.945936 |
| 75021.444822 | 156.471513 | 59.508867 | 1816.117313 |
| 76506.892200 | 148.586079 | 61.695917 | 1782.893330 |
| 56166.050134 | 75.667432 | 57.971347 | 1646.615390 |

## *2.3 Machine Learning Training Model*

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=101)
```

```
models =[("LR", LogisticRegression(max_iter=1000)),("SVC", SVC()),('KNN',KNeighborsClassifier(n_neighbors=10)),
         ("DTC", DecisionTreeClassifier()),("GNB", GaussianNB()),
        ("SGDC", SGDClassifier()),("Perc", Perceptron()),("NC",NearestCentroid()),
        ("Ridge", RidgeClassifier()),("NuSVC", NuSVC()),("BNB", BernoulliNB()),
         ('RF',RandomForestClassifier()),('ADA',AdaBoostClassifier()),
        ('XGB',GradientBoostingClassifier()),('PAC',PassiveAggressiveClassifier())]
```

| | Version1 | Version2 | Version3 | Version4 | Version5 |
|---|---|---|---|---|---|
| RF | 0.660 | 0.706 | 0.784 | 0.787 | 0.823 |
| SVC | 0.649 | 0.720 | 0.687 | 0.639 | 0.698 |
| XGB | 0.611 | 0.672 | 0.785 | 0.814 | 0.825 |
| ADA | 0.544 | 0.583 | 0.772 | 0.776 | 0.813 |
| LR | 0.307 | 0.541 | 0.807 | 0.307 | 0.699 |

### 2.3.1 Model selection

Random Forest (RF): Random Forest is an ensemble model that works by creating multiple decision trees during training and uses 'voting' from each tree to predict the outcome. It is effective in various classification and regression problems and is often more resistant to overfitting than a single decision tree.

Support Vector Machine (SVC): SVC is a machine learning model that seeks to find the best 'hyperplane' in a multi-dimensional space that can separate different classes. It is popular in classification tasks and performs well on high-dimensional datasets.

XGBoost (XGB): XGBoost (eXtreme Gradient Boosting) is an implementation of gradient boosting with a focus on efficiency and speed. This model uses 'boosting,' which is a step-by-step additive learning approach, and is often used in data science competitions due to its high performance.

AdaBoost (ADA): AdaBoost (Adaptive Boosting) is another method of boosting that works by adjusting the weights of samples in the dataset at each training stage. It focuses on improving the model where the previous model has made errors and is often used for classification.

Logistic Regression (LR): Despite its name 'regression,' Logistic Regression is a classification model used to estimate the probability of a binary outcome. It uses the logistic function to predict the probability of an outcome being in one of two classes. It is suitable for classification problems with binary target classes.

Each model has its strengths and weaknesses, and choosing the appropriate model for a given problem depends on the complexity of the data, the type of problem, and practical requirements.

## ***3. Results and Discussion***

Time line history version

| Version | 01 | 02 | 03 | 04 | 05 |
|---|---|---|---|---|---|
| Best accuracy | 0.660 | 0.720 | 0.807 | 0.814 | 0.825 |
| Model | RF | SVC | LR | XGB | XGB |
| Replace missing value with 0 | | | | | |
| Delete all data in a row when missing values are found. | | | | | |
| Replace Missing Value with condition | | | | | |
| Replace Outlier with condition | | | | | |
| Add Feature Engineering | | | | | |

In Version 1 , missing values were replaced with 0, and the Random Forest (RF) model provided the highest accuracy at 0.660.

Version 2 involved deleting all data in rows where missing values were found, and the Support Vector Classifier (SVC) yielded the best accuracy at 0.720

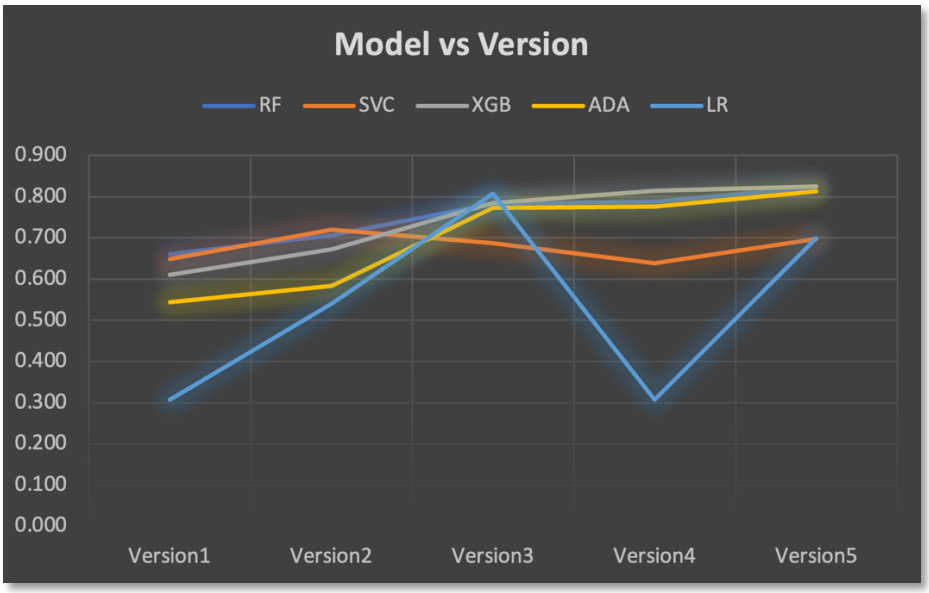
Version 3 presented more significant improvements in accuracy, with missing values replaced under certain conditions, and the Logistic Regression (LR) model achieved the best accuracy at 0.807.

In Version 4 outliers were replaced with conditions, and the XGBoost (XGB) model demonstrated the best accuracy at 0.814.

Version 5 showed the best accuracy in the series at 0.825, with the addition of Feature Engineering in data processing, and the XGB model continued to provide the highest accuracy.

From this table, we can conclude that different data management approaches affect the performance of models in each version, and incorporating feature engineering may help improve model accuracy. Additionally, no single technique is suitable for all models, and the techniques used in data processing must be tailored to the specific characteristics of the dataset and the model used.

A line graph showing the accuracy of various machine learning models across multiple versions of testing or processing the dataset.




There were five versions of the experiment, each with different approaches to handling missing data or outliers and feature engineering.

The models RF (Random Forest), SVC (Support Vector Classifier), XGB (eXtreme Gradient Boosting), ADA (AdaBoost), and LR (Logistic Regression) were tested in each version.

The most effective version was Version 5, using the XGB model, which achieved the highest accuracy score of 0.825.

For data management, Version 5 incorporated new feature engineering, which was evident from the improved accuracy compared to earlier versions.

Replacing missing values with conditions and replacing outliers with conditions were applied in Versions 3 to 5, resulting in increased accuracy scores.

Deleting all data in a row when missing values were found was used in Versions 2 and 3, but it seemed that adding feature engineering in subsequent versions was more effective.

This analysis indicates that intelligent feature engineering and managing missing values and outliers can enhance the accuracy of machine learning models.

## ***4. Conclusion***

Effective Data Management Enhances Model Performance: The experiments show that efficient data management can significantly improve machine learning model performance. This is clearly seen in the improved accuracy across different versions of the experiment.

Importance of Specific Data Management Techniques: Not every data management technique is suitable for all models. For instance, while replacing missing values with zeros might work well for some models, it may not be appropriate for others.

Impact of Removing Rows with Missing Values: While removing entire rows where missing values are found can simplify the dataset and enhance the performance of certain models, it could also lead to the loss of valuable information for other models.

Model-Specific Responses to Data Management: Different models respond differently to changes in data management. For example, the XGBoost model performed best in the version with added features, whereas Logistic Regression benefited more from specific conditions for replacing missing values.

Testing and Evaluation Are Crucial: The importance of testing and evaluating various models with different data management techniques cannot be overstated. This process helps in understanding which models are best suited to specific changes in data management.

Selecting Appropriate Techniques for Dataset and Model: It's vital to carefully choose data management techniques that are appropriate for both the dataset and the specific model being considered, as this significantly impacts the efficiency of the model.