Analyses of Approaches to Deal with Missing Data in Water Quality Data Set

Ruoqi Yang

Boston University
**Corresponding author. Email: ruoqi25@bu.edu*

ABSTRACT
The water quality model from the Kaggle Dataset provides useful data for predicting the potability of a specific specimen. Dealing with missing values is imperative for computer and data scientists to obtain accurate results. When using datasets with missing values in statistical analysis or hydrological modeling, the findings can be misguided. This research is going to explore several common imputation methods and techniques to handle a large number of data streams by using neural networks and machine learning. The author evaluates various imputation approaches that can be applied to water quality data using the proposed approach. Eventually, the KNN imputation method performs the best and generates the most accurate testing results among other imputation methods.

Keywords: *Water Quality, Imputation Method, Missing Data, Neuro Network, Data Analysis.*

1. INTRODUCTION

Water, the most fundamental substance that nurtures creatures on earth, is categorically crucial to human beings. Water quality in nature also has an influence on the condition of ecosystems that all living organisms depend on. Therefore, it is essential to set up standards and measurements to ensure water quality is satisfactory for human consumption. Machine learning helps us analyze and predict water quality given a dataset [1]. Missing data in machine learning is very common which might incur data corruption, equipment failure or human error. An incomplete dataset is more frequently presented than a complete one. Data leakage might result in failure and an inaccurate result when training and modeling it using neural networks. Imputation approaches for specific water quality variables are being developed in the majority of current investigations [2]. The experimental results show that the recommended solutions perform better in specific situations. However, determining which imputation approach consistently performs well in a wide range of application circumstances is difficult.

This paper will be organized as follows. Section 2 reviews the water quality dataset from kaggle data website. Section 3 employs histogram and heatmap to visualize the missing portion of the water quality dataset. Section 4 and 5 introduce the application of neural networks in the data science field and the training network in this paper. Section 6 compares several imputation methods and utilizes neural network to analyze the advantages and defects of each. Overall, the author uses Python to model the kaggle water quality dataset with numerous missing inputs and analyze the efficiency of each method. The purpose of this paper is to develop an optimal imputation method when dealing with small scale datasets and analyze the efficiency of neural networks in the data analysis field.

2. KAGGLE WATER QUALITY DATASET

(1) pH value:

pH value measures how acidic or basic a water sample, which ranges from 0-14.

(2) Hardness:

Hardness measures the amount of calcium and magnesium dissolved in water.

(3) Solids (Total dissolved solids -TDS)

Solidity of water measures the suspended and dissolved solid in water.

(4) Chloramines:

Chloramines are disinfectants used to treat drinking water.

(5) Sulfate:

Sulfates level may make the water taste bitter or like medicine.

(6) Conductivity:

Electric conductivity of water measures the saltiness of water samples, which normally ranges from 0 and 1,500 uS/cm.

(7) Organic carbon:

Organic carbon is any carbon-based containment matter that dissolves in untreated water.

(8) Trihalomethanes:

Trihalomethanes is also used to disinfect water normally regarded as TTHM.

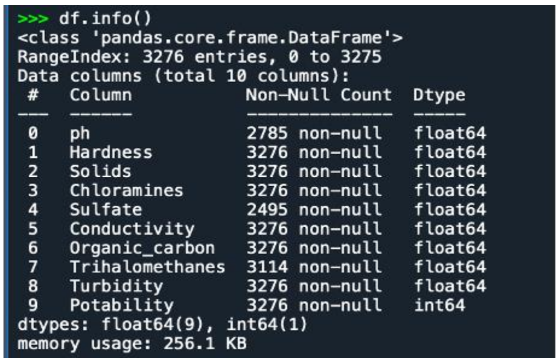
(9) Turbidity:

The turbidity of water depends on the quantity of solid matter present in the suspended state. Low turbidity values indicate high water clarity. High value indicates low water clarity.

(10) Potability:

It indicates if water is safe for human consumption where 1 means potable and 0 means not potable [3].

3. DATA ANALYSIS



```
>>> df.info()
<class 'pandas.core.frame.DataFrame'>
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
```

Figure1 Data frame of water quality sample set

Data from the Python console reveals that there are several columns and rows where many lines of data are missing and stored with the value of NaN. (Fig. 1) The built- in function df.info() shows that the pH value, Sulfate and Trihalomethanes display a significant leakage of data where the column of pH value has 491 missing items, the column of Sulfate has 781 missing items, and the column of Trihalomethanes has 162 items. The blank area in the missingno matrix also helps visualize the data. (Fig. 2)

The dataset shows that approximately sixty percent of sample water is potable, and forty percent of sample water is not potable. The significant difference in sample diversity might incur training error in machine learning [4]. On the other hand, the heat map also shows there is no direct correlation between all variables since all values on the heatmap cluster around zero, and smaller numbers represent small deviations. (Fig 3)

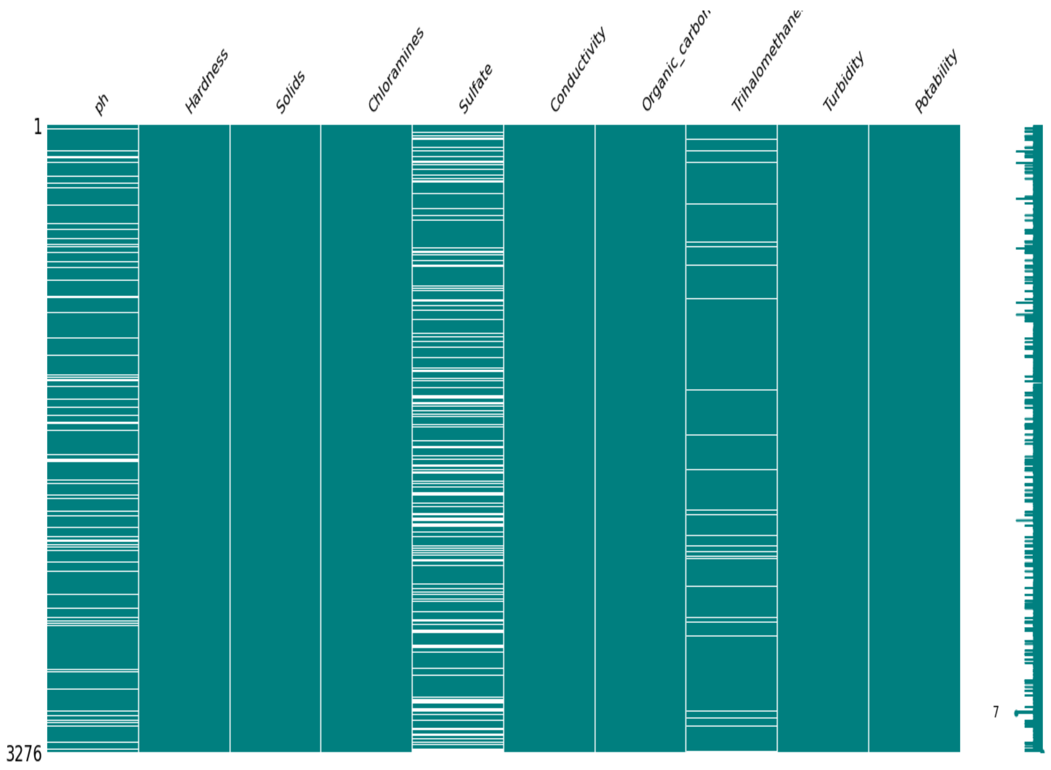


Figure2 Missingno matrix of water quality data set

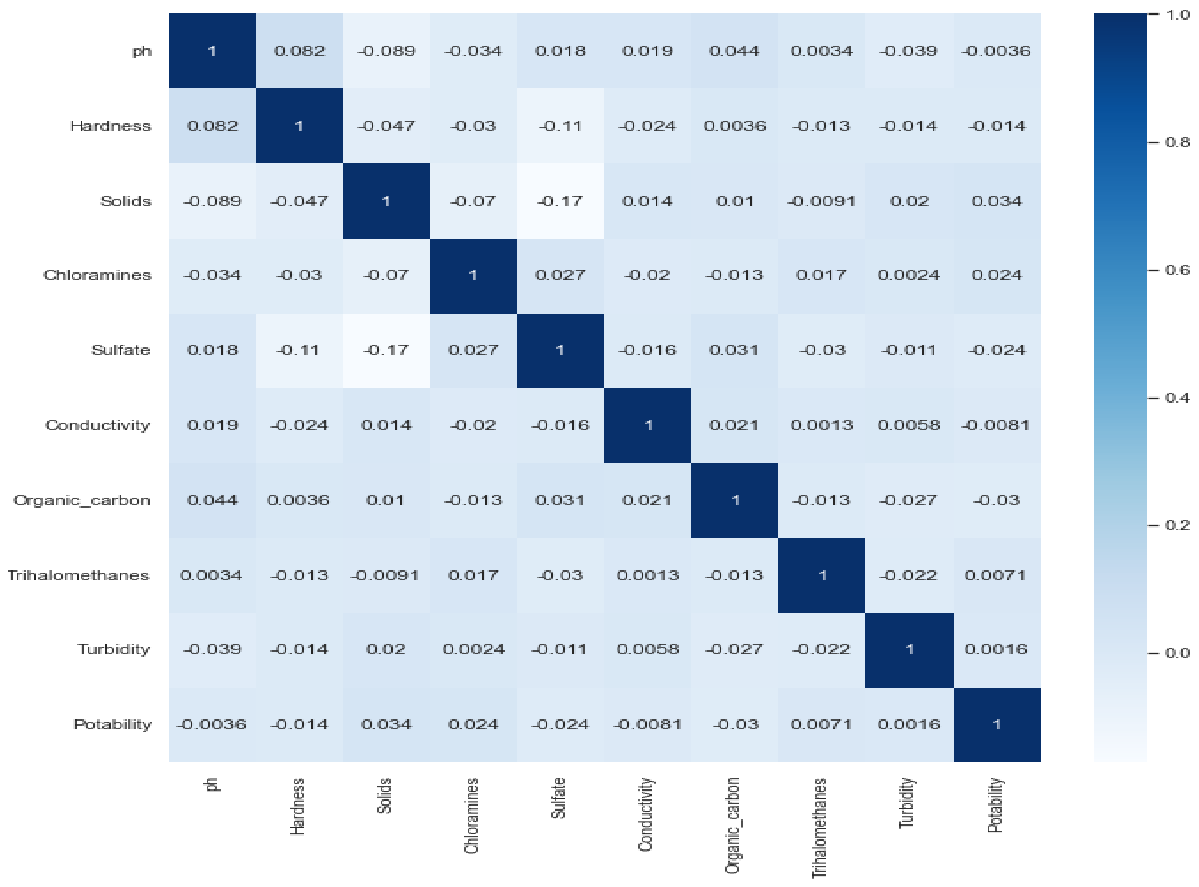


Figure3 Heat map of water quality dataset

4. NEURAL NETWORK

Artificial neural networks is a subset of machine learning that can be used to analyze and predict the trend or behavior of a specific dataset. The concept of neural network was inspired by the human brain structure which contains myriads of neurons and edges connected to each other. They transfer information using neurons, edges and the rest of the nervous system which send out messages through electrical impulses and chemical signals. Analogy to neurons in the human brain, the node layer includes one input layer and multiple hidden layers and one output layer in an artificial neural network. Each node, or artificial neuron, is linked to the others and has its own weight and threshold. If the output of a node exceeds a certain threshold value, the node is activated, and data is sent to the next tier of the network. If this is not the case, no data is transferred to the next network tier. Neural Networks are universal function approximators which are mathematical functions that map input to output.

A dense layer in a neural network is deeply connected to the layer before it, meaning that the layer's neurons are connected to every neuron in the layer before it. In artificial neural network networks, this layer is the most widely utilized since the sequential nature allows to stack linearly layers on top of each other. In a model, the dense layer's neuron receives input from every neuron in the preceding layer, and the dense layer's neurons conduct matrix-vector multiplication. The row vector of the output from the preceding layers is identical to the column vector of the dense layer in matrix vector multiplication. In matrix-vector multiplication, the row vector must have the same number of columns as the column vector. The process of finding the weights and biases is called training the network and it is usually regarded as Feedforward and Backpropagation. (Fig 4) [5]

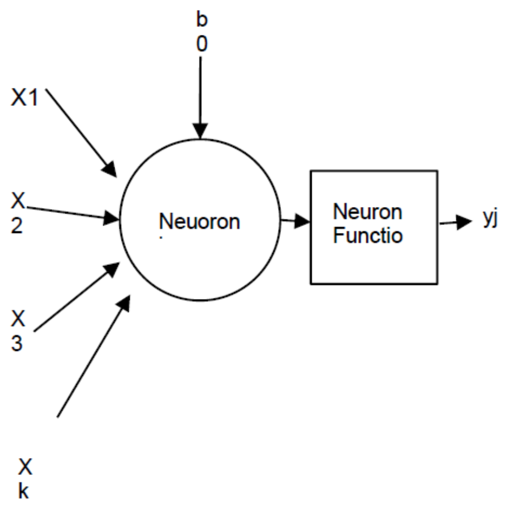


Figure 4 Feed Forward math formula

$$\square_{\square} = \square(\square_{1} \times \square_{1} + \square_{2} \times \square_{2} \cdots\cdots \square_{\square} \times \square_{\square} + \square 0) \quad (1)$$

5. CONSTRUCT TRAINING NETWORK

This paper uses python package keras to construct neural networks and analyze data. To avoid bias or significant distortion of a data set, the data needs to be preprocessed to enhance the performance of machine learning. Normalization is the process of converting raw data into a clean set such that values of numeric columns in a dataset have been transformed into a similar scale without distorting the ranges of values. Z-score is the most commonly used normalization method in machine learning where x represents the raw score, μ is the mean value and σ is the standard deviation [6].

$$\square = (\square - \square)/\square \quad (2)$$

Activation function is the function that applied to each neuron in the network (1) to obtain the output of each node. There are multiple renown activation functions in machine learning such as “sigmoid”, “Rectified Linear Unit”(ReLU) and “hyperbolic tangent” activation functions, but they must be used appropriately.

The next step is to construct training networks. In this model, the training network will be added 5 layers and each layer will be composed of a dense layer with 120 neurons and a dropout layer that truncate 30 percent data to handle overfitting. “Sigmoid” activation function has a binary nature that maps input into some value between 0 and 1. (Fig 5) Therefore, it can be utilized to map potability into yes or no. Due to the differentiable nature (Fig 6), the model will use “ReLU” activation because its performance is much quicker than other activation functions [7]. On the other hand, the dataset is separated into two segments where 70 percent of the data are consumed to train the network and 30 percent of data are used to test the performance of the model. Optimizer and loss function are selected randomly and are listed as follow. (Fig 7)

$$\square(\square) = \frac{1}{1+\square^{-\square}} = \frac{\square^{\square}}{\square^{\square}+1} = 1 - \square(-\square)$$

$$\square(\square) = \square^{+} = \square\square\square\,(0, \square)$$

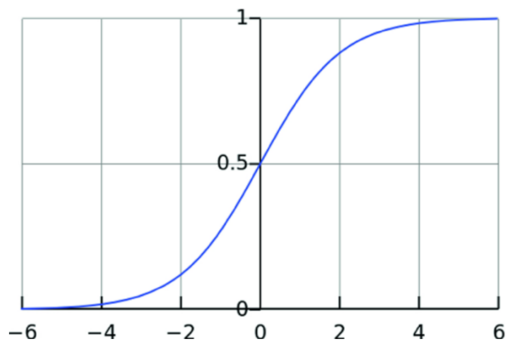


Figure 5 Sigmoid activation function

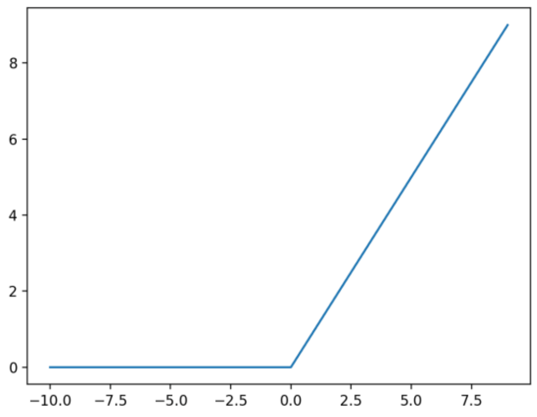
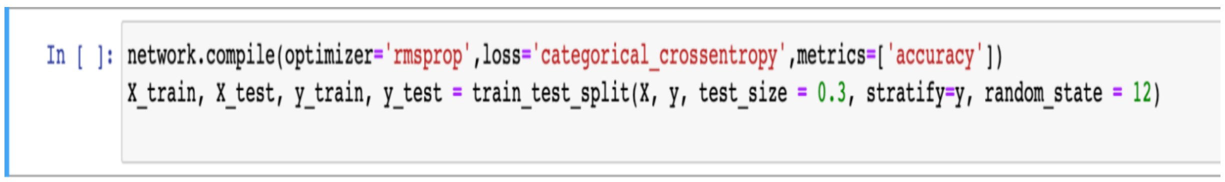


Figure 6 Relu activation function



```
In [ ]: network.compile(optimizer='rmsprop',loss='categorical_crossentropy',metrics=['accuracy'])
        X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, stratify=y, random_state = 12)
```

Figure 7 Optimizer and training proportion

6. METHOD

6.1. Deleting Missing Data

Data omission is the most common imputation method that data scientists and computer scientists use. Data omission is an intentional behavior such that entire columns or rows of a data item are removed from the dataset. It is definitely the most effective way to deal with a large dataset since directly removing the data does not require any calculation. Although it is relatively easy to apply, the sample space might drop significantly when dealing with a larger dataset, leading to output in a bad direction [8].

df_dropped = df.dropna()

Table 1. Deletion imputation method result

Size of X_train	1407
Test Accuracy	0.6655629277

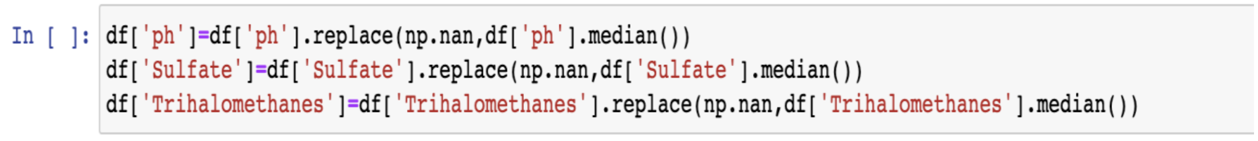
Test Loss	0.8560913205
Train Accuracy	0.99147123
Train Loss	0.1007632166

The result shows a test accuracy of 0.666 and a training accuracy of 0.9915 which is relatively good compared to other imputation methods. However, the size of the training set has been reduced by 30% due to removing all rows that contain missing values. Despite the model trained with the removal of all missing values to create a robust model, myriads of information were lost, and this imputation method works poorly if the percentage of missing values is excessive in comparison to the complete dataset.

6.2. Median or Mean Imputation

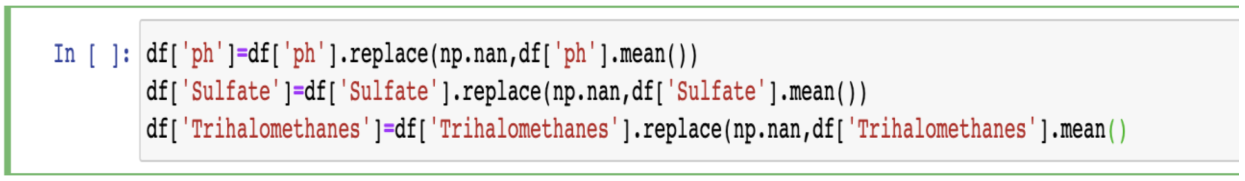
Mean and Median imputation methods are similar to each other in which all NaN values in the dataset will be replaced with the mean, median of the values in the corresponding column. Compared with the previous way, this strategy can prevent data loss.

6.2.1 Code for Median imputation



```
In [ ]: df['ph']=df['ph'].replace(np.nan,df['ph'].median())
        df['Sulfate']=df['Sulfate'].replace(np.nan,df['Sulfate'].median())
        df['Trihalomethanes']=df['Trihalomethanes'].replace(np.nan,df['Trihalomethanes'].median())
```

6.2.2 Code for Mean Imputation



```
In [ ]: df['ph']=df['ph'].replace(np.nan,df['ph'].mean())
        df['Sulfate']=df['Sulfate'].replace(np.nan,df['Sulfate'].mean())
        df['Trihalomethanes']=df['Trihalomethanes'].replace(np.nan,df['Trihalomethanes'].mean()
```

6.2.3 Result

Table 2. Median imputation method

Size of X_train	2293
Test Accuracy	0.64394712448
Test Loss	0.87261056900
Train Accuracy	0.9511556625
Train Loss	0.1930120289325

Table 3. Mean imputation method

Size of X_train	2293
Test Accuracy	0.6327568888866
Test Loss	0.79316341869
Train Accuracy	0.9476668238639
Train Loss	0.21497708559036

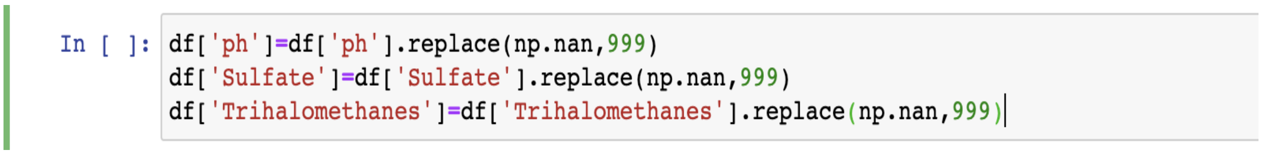
The result for Mean and Median imputation are consistent with each other. Both models return a test accuracy of around 0.63 which is a relatively good prediction. Although the mean/median method is easy to implement and efficient, there are still many defects to improve. If the number of NaN is large in comparison to the entire dataset, the variance of the variable will be significantly skewed, which might cause an underestimation of the variance [9].

In addition, the estimation of covariance and its correlation with other variables in the sample will also be affected. Because the mean / median value that now replaces the missing data does not always retain the relationship with the remaining variables, mean/median imputation may change internal relationships. Finally, focusing all missing data at the mean / median value may result in common occurrences in the distribution being identified as outliers.

6.3. Arbitrary Value Imputation

Arbitrary imputation is an imperative technique used in imputation as it can handle both numerical and categorical variables. For categorical all NaN values will be filled with ‘missing’ and for numerical empty items will be filled with any arbitrary integer. (Integer 999 in this model). This should be used when data is not missing at random. Arbitrary value imputation method works well with tree-based algorithms but not with linear regression or logistic regression [9].

6.3.1 Code



```
In [ ]: df['ph']=df['ph'].replace(np.nan,999)
        df['Sulfate']=df['Sulfate'].replace(np.nan,999)
        df['Trihalomethanes']=df['Trihalomethanes'].replace(np.nan,999)
```

6.3.2 Result

Table 4. Arbitrary Value Imputation Method Result

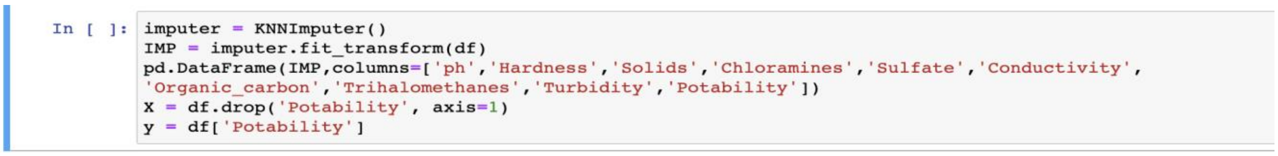
Size of X_train	2293
Test Accuracy	0.55645984411
Test Loss	0.878900647163
Train Accuracy	0.89271694421
Train Loss	0.32720312476

Similar to mean/median imputation technique, the arbitrary imputation method is easy to implement, and it retains the importance of “missing values' ' if it exists. However, it can distort original variable distribution because arbitrary values can create outliers. The result obtained from our model returns a test accuracy of 0.5564 which performs awful compared to the previous two imputation methods. On the other hand, the value that is chosen for imputation cannot be too “arbitrary”. It is prohibited to use a value that is close to the mean or median. Replacing the NA by arbitrary values should be used when there are reasons to believe that the NA is not missing at random.

6.4. Nearest Neighbor Imputation Imputation (KNN Imputation)

When the training set is small or moderate in size, K-nearest neighbors can be a quick and effective method for imputing missing values. Missing value imputation using univariate methods is a simple way of estimating the value that may not always provide an accurate result. Algorithms like k-Nearest Neighbors (kNN) can be used to impute missing data values. According to sociologists and community researchers, people reside in communities because their neighbors provide a sense of comfort and safety, as well as community attachment and relationships that help to define a community's identity. Similarly, k-Nearest Neighbors (kNN) is a data-based imputation technology that finds neighboring points using a distance metric and estimates missing values using completed values of neighboring observations.

6.4.1 Code



```
In [ ]: imputer = KNNImputer()
        IMP = imputer.fit_transform(df)
        pd.DataFrame(IMP,columns=['ph','Hardness','Solids','Chloramines','Sulfate','Conductivity',
        'Organic_carbon','Trihalomethanes','Turbidity','Potability'])
        X = df.drop('Potability', axis=1)
        y = df['Potability']
```

6.4.2 Result

Table 5. KNN Imputation Method Result

Size of X_train	2293
Test Accuracy	0.6429297924
Test Loss	0.761351287364
Train Accuracy	0.959441781044
Train Loss	0.209606617

KNN is also known as Lazy Learner because it has no training period. During the training phase, it does not learn anything. The training data isn't used to derive any discriminative functions. In other words, it does not require any training. It saves the training dataset and uses it only when making real-time predictions to learn from it. [10]This makes the KNN method much faster than other training-based algorithms. Furthermore, Since the KNN algorithm requires no training before making predictions, new data can be added seamlessly which will not impact the accuracy of the algorithm. On the other hand, there remain lots of defects for the KNN algorithm. First and foremost, KNN does not work well with a large dataset. In large datasets, the cost of calculating the distance between the new point and each existing point is huge which degrades the performance of the algorithm. Second, the KNN algorithm doesn't work well with high dimensional data because with a large number of dimensions, it becomes difficult for the algorithm to calculate the distance in each dimension. Lastly, feature scaling (standardization and normalization) is done before applying KNN algorithm to any dataset.

7. CONCLUSION

In conclusion, the amount of the missing data has a considerable impact on imputation accuracy. The majority of imputation approaches do a good job at filling in missing data. KNN and median imputation method perform better compared to the other two, but the latter one might bring significant distortion to the original dataset. Therefore, KNN is the most promising imputation method when dealing with small size dataset such as water quality dataset. However, this model is refrained by the water quality dataset in which all missing data clusters at two or three missing columns. KNN network might not be the optimal imputation method when missing items are randomly selected from all columns. The author will focus on that in the future research.

REFERENCES

[1] Zhang Y., Fitch P., Thorburn P.J. Predicting the trend of dissolved oxygen based on the kPCA-RNN model Water, 12 (2) (2020), p. 62

[2] Soley-Bori M. Dealing with missing data: Key assumptions and methods for applied analysis. Boston University, 4 (2013), pp. 1-19

[3] Aditya Kadiwal. 5/14/2021. Water Quatlity Dataset, version 3. Retrieved 02/16/2022 from https://www.kaggle.com/adityakadiwal/water-potability

[4] S. Azadi, J. Feng, and T. Darrell, ‘‘Learning detection with diverse proposals,’’ in Proc. IEEE Conf. Comput. Vis. Pattern Recognit., Jul. 2017, pp. 7374–7377

[5] S. Albawi, T. A. Mohammed and S. Al-Zawi, "Understanding of a convolutional neural network", 2017 International Conference on Engineering and Technology (ICET), 2017, pp. 1-6, doi: 10.1109/ICEngTechnol.2017.8308186.

[6] Jo, J.-M. Effectiveness of Normalization Pre-Processing of Big Data to the Machine Learning Performance. The Journal of the Korea Institute of Electronic Communication Sciences, 14(3), 2019, 547–552. https://doi.org/10.13067/JKIECS.2019.14.3.547

[7] Li, B., Li, Y. & Rong, X. The extreme learning machine learning algorithm with tunable activation function. Neural Comput & Applic 22, (2013), 531–539 https://doi.org/10.1007/s00521-012-0858-9

[8] Therese D. Pigott. A Review of Methods for Missing Data, Educational Research and Evaluation, 7:4, 2001, 353-383, DOI: 10.1076/edre.7.4.353.8937

[9] Liu, G., Zhu, J., & Liu, X. Imputation Method of Random Arbitrary Missing Data Based on Improved Close Degree of Grey Incidence. The Journal of Grey System, 31(2), (2019) 74+. https://link.gale.com/apps/doc/A672005936/AONE?u=mlin_oweb&sid=googleScholar&xid=90ccc728

[10] Malarvizhi, R., & Thanamani, A. S. K-nearest neighbor in missing data imputation. International Journal of Engineering Research and Development, 5(1) (2012) 5-7.