MACHINE LEARNING AS A TOOL FOR ANALYSING WATER POTABILITY USING SUPPORT VECTOR MACHINE

By

OGUNGBITE, ALABA BUKOLA

STUDENT NO: B1148714

SCHOOL OF COMPUTING, ENGINEERING AND DIGITAL TECHNOLOGIES

DATA SCIENCE FOUNDATION

IN-COURSE ASSESSMENT

12/01/2021.

Abstract

Water is an essential commodity that requires special attention than it gets. A portable water is the one that is free from contamination and is suitable for drinking. However, many factors are responsible for water contamination these include industrialization, urbanization and human factor including farming have led to water pollution and this had made water unfit for drinking, causing various harmful diseases; thus, adversely affecting the individual's health. This work focuses on machine learning as a tool for analysing water potability using support vector machine by using a dataset downloaded from Kaggle titled "water potability". This contains 10 attributes that are suitable for analysis of drinkable water, these include: pH, potability, hardness, solids, chloramines, sulfate, conductivity, organic_carbon, trihalomethanes, turbidity situated for analysing water potability.

OUTLINE SPECIFICATION OF THE PROBLEM

The objective of this work is to determine the effectiveness of Support Vector Machine (SVM) as a tool in predicting water quality.

The specific objectives were:

- To predict water quality
- Imploring Machine learning as a tool to analyse water quality
- Support vector machine will be used to predict water quality.

TABLE OF CONTENTS AND FIGURES

CHAPTER ONE

INTRODUCTION AND BACKGROUND

The role of water in the health of individual is indispensable and cannot be overemphasized. Somani *et al.*, (2014) reported that more than 80% of the health challenges the citizens of many developing countries of the world are facing is due to lack of portable water. Thus, problems associated with lack of access to good drinkable water is not associated with any particular age group or sex alone as both young and old people, and both males and females are being affected health wise due to consumption of poor water. In fact, Mustafa *et al.*, (2013) stated that over 1.8 million people's death recorded yearly is due to drinking of contaminated water.

United Nation (UN) in one of its reports stated that more than 80% used water returned back into the environment and being used by over 1.8 billion people without any proper treatment and thus, causing high risk of cholera, polio, dysentery and typhoid (UN WWDR, 2017). Over 3 billion people globally are prone to face health challenge due to consumption of water that is not potable (UN-Water, 2021). This is because many countries of the world has no good water quality data that can help in finding solutions to unavailability of potable especially in developing countries where mechanism for recycling water waste is poor or not available (Un-Water, 2021). Furthermore, it is well documented that ever since 1990s, the rate of potable water available to the citizen of many continents of the world including Africa, Asia and Latin America has significantly reduced because of pollution from different sources probably because of high population in these areas (UNEP, 2016). UN-Water (2011) opined that many problems that are caused by unavailability of good quality water are due to intensive agricultural practices, industrialization and untreated urban runoffs.

In order to make potable water available to everyone, governments have laid down different rules to guide water consumption and waste which directly or indirectly affect potable water availability. Despite the efforts of government of the world and different health related bodies

to make good water available to everyone, the major challenge has been the water being contaminated after treatment at different water treatment plants (Somali *et al.*,2014) because there is no well-established algorithm to help when the data collected from analysis of water. Therefore, there is a need to develop an algorithm that could help in analysing water quality parameters and give reasonable information on how potable a water is. Thus, such algorithm could help in predicting how the quality of water irrespective of seasonal change.

Machine learning has been proven to be a vital tool to analyse data of different type. Wilcon *et al.*, (2013) stated that machine learning is such an information technology tool that could provide efficient and effective alternative than the traditional analytical means of analysing data. A support vector machine is a supervised machine learning model using classification algorithms for two-group classification problems. During data splicing (training set and testing set), SVM models are given sets of labelled training data for each category and they are able to categorise new text. Unlike other algorithm, SVM has two advantages: limited number of samples (in the thousand) with better performance and higher speed.

In order, to determine the efficient prediction of quality of water in real life, this current work used support vector machine learning to predict the water quality using water potability dataset obtained from Kaggle.

CHAPTER TWO

2.0 Literature Review

Earth is made of 71% water and this entity has played important role in the existence of human on earth. However, despite high volume of water on earth and its importance to human, larger proportion of this natural endowment is not portable and thus, create a major concern among the individual, researchers and government of the world. Alghamdi *et al.,* (2020) stated that most of the water on earth have unique quality standards which indicate their portability either for application or usage. These authors reported that several human activities including industrialization has contributed to the pollution experienced in many of the water bodies on earth. In addition, unavailability of facilities to recycle many of the used water also contributes tremendously to the amount of hygienic drinkable water (Zeilhofer *et al.*, 2007). Thus, several health challenges have been linked to the consumption of these polluted water. In fact, United Nations (UN) in one of her reports stated that over 1.5 million people die yearly due to consumption of impure water especially in the developing countries where amount of contaminated water is high. More than 80% health problems faced by many people in the developing countries is linked to unavailability of portable water and this has caused more death than accidents, crimes and terrorism (UN, 2010). Hence, there is an urgent need to design and suggest new means of analysing water quality and helping in forecasting the patterns of water quality for proper monitoring irrespective of the seasonal change (Farrell-Poe *et al.*, 2000). Taskaya-Temizel and Casey (2005), Babu and Reddy (2014), Zhang *et al.*, (2014) as well as Alghamdi *et al.*, (2020) opined that there is necessity for creating a special type of model to predict water quality as this create more reliable results.

Recently, a universal tool using for different types of tasks is known as machine learning algorithms, giving room for advanced possibilities for dealing with unsupervised clusterization, data imputation, classification and regression . These tools are commonly used in many

research. Though, they are less common among environmental engineering workers. It provides extremely efficient traditional alternative to an analytical approaches (Wilcox, *et al.*, 2013).

2.1 Different tools used for analysing water portability

Water quality can be estimated using machine learning algorithms (Shafi *et al.*, 2018). Classical machine learning algorithms involve in estimating water quality include; K Nearest Neighbour(KNN), Gaussian Naïve Bayes, Random Forests, Artificial Neural Networks and Support Vector Machines.

2.1.1 K Nearest Neighbour (KNN)

This is one of the basic and naturally understandable algorithms used for different tasks such as classification. It is one of the simplest tools in Machine Learning algorithms that based on supervised learning technique. KNN compares the new data with already existing data and place the new data to the most similar category with the existing data. This machine learning tool can be used for imputation of missing data, that is, replacing missing data with the nearest possible data. Batista and Monard, (2002) opined that it is generally possible to use any model for imputation but KNN becomes most suitable when considering computational costs while keeping sufficient results. Thus, this model has advantage of assigning missing value to the most common and related neighbours among the given set of data (Lu *et al.*, 2012).

2.1.2 Naïve Bayes

This is one of the machine learning most popular algorithms that helps in classification of data based on the conditional probability computation of values. It is one of the simple and fast algorithms that work on based on the principle of Bayes theorem assuming that probability of the presence of a feature is not related to the probability of the feature presence (Zhang,

H.,2004). Gaussian Naïve Bayes which is an extension of Naïve Bayes is a machine learning algorithm used in many classification functions.

2.1.3 Random forests

It is a new model, developed in 1990s and it produces many decision trees that emerge together to ensure more accurate prediction. Thus, each tree is built from a bootstrap sample gotten from the observed data. The idea behind the Random Forest model is the multiple uncorrelated models that perform better when working together rather than individual.

2.1.4 Artificial neural networks (ANN)

This was introduced in 1943 by Warren McCulloch and Walter Pitts and become one of the most popular machine learning models that has numerous applications such as regression, classification, 15 image recognition and so on. Building several layers that are made up of interconnected nodes containing the activation function is what forms the bases of this model.

2.1.5 K-means clustering

Unsupervised Clusterization is one of the most useful analytical tools in which data are classified by algorithm into specific amount of classes depending on the internal patterns. This could be used to search for the subtypes and subclasses that could be used for research process, value/compound (Likas *et al.*, 2003)

2.1.6 Support Vector Machines

Support Vector Machine (SVM) is one of the fundamental algorithms and the root of this algorithm is linked to the family of linear models. This model is designed by transferring the original vector in the higher dimension and with maximum gap, searches for dividing hyperplane. On both sides of the hyperplane separating classes is built a two parallel hyperplanes. This separating hyperplane is the one that maximizes the distance to two parallel hyperplanes.

Cortes and Vapnik (1995) reported that this algorithm works on the assumption that the more the difference and distance between the two parallel hyperplanes the smaller the mean error of the classifier. Nawar *et al.,* (2016) as well as Luca *et al.,* (2017) opined that the method of achieving effective non-linear relationships is by using SVM alone. Therefore, this current work focused on the usage of SVM for predicting portability of water.

CHAPTER THREE

3.0 Technical Implementation (Developed System or Model)

3.1 Data Collection

The selected dataset was downloaded from Kaggle website titled water potability. It has 10 attributes that it is suitable for analysis of drinkable water. The columns are Ph, Hardness, Solids, Chloramines, Sulfate, Conductivity, Organic carbon, Trihalomethanes, Turbidity and Potability. The description of each columns was explained in the Table 1 while Table 2 and 3 showed the mineral composition and microbial composition of drinkable water respectively as approved by World Health Organisation (WHO).

Table 1. The parameters for quantifying water portability

ATTRIBUTES	DESCRIPTION
PH	PH is the measure of the acidity/basic of water. It ranges from 0 to 14. PH below 7.0 is believed to be acidic while PH above 7.0 is believed to be alkaline or basic.
HARDNESS	Hardness of water occur when the dissolved calcium, magnesium and minerals are high in water.
SOLIDS	Solids are the inorganic salts and small amounts of organic matter that are present in water
Chloramines	Disinfectants utilized in treating drinking water are chloramines. When ammonia is added to chlorine to treat drinking water chloramines are formed.
SULFATE	The source of most sulfate compounds is the oxidation of sulfite ores, the presence of shales, or the industrial wastes. It can be found in almost all natural water.
CONDUCTIVITY	Conductivity is the quantity of water's ability to pass electrical flow. It's ability is related to the ions concentration in water.
ORGANIC CARBON	Organic carbon can be used as non-specific gauge of quality of water or sanitation of pharmaceutical equipment
TRIHALOMETHANES	When source of waters are dependent on marine influences, regular precipitation, levels of elevated of organic matters is called Trihalomethanes.
TURBIDITY	The amount of relative clearness of water is called turbidity. Turbidity does make water opaque or cloudy.
POTABILITY	Water is said to be potable when it is safe to drink.

Table 2. WHO drinking water standard

PARAMETER	UNIT	LIMIT
Aluminium	mg Al/l	0.2
Arsenic	mg As/l	0.05
Barium	mg Ba/l	0.05
Berylium	ug Be/l	0.2
Cadmium	ug Cd/l	5.0
Calcium	mg Ca/l	200.0
Chromium	mg Cr/l	0.05
Copper	mg Cu/l	1.0
Iron Total	mg Fe/l	0.3
Lead	mg Pb/l	0.01
Magnesium	mg Mg/l	150.0
Manganese	mg Mn/l	0.1
Mercury	ug Hg/l	1.0
Selenium	mg Se/l	0.01
Sodium	mg Na/l	200.0
Zinc	mg Zn/l	5.0

Chlorides	mg Cl/l	250.0
Cyanide	mg Cn/l	0.1
Fluorides	mg F/l	1.5
Nitrates	mg NO_3/l	10.0
Nitrites	mg NO_2/l	-
Sulphates	mg SO_4/l	400.0
Suphides	mg H_2S/l	0
TOTAL "drins"	ug/l	0.03
TOTAL "ddt"	ug/l	1.0
Hydrocarbons	mg/l	0.1
Anionic Detergents	mg/l	0
pH		9.2
Total dissolved solids	mg/l	1500
Total hardness	mg/l	500
Alkalinity	mg/l	500

Table 3. Microbiological parameters

Bacteria	Count/ml	100
Coliform	Count/100ml	0
E. Coli	Count/100ml	0
Salmonella	Count/100ml	0

3.2 Data Analysis

After data processing, support vector machine algorithms were employed to predict water quality. There are some steps to take before applying machine learning algorithm, preliminary steps like, data splitting, correlation analysis to prepare the data for support vector machine algorithm.

3.3 Data Pre-Processing

The data used for this research was downloaded from Kaggle and it was cleaned thoroughly before passing through training and testing stage.

Necessary libraries were installed into R before loading the data.

Libraries package used are:

```
Source on Save
install.packages("caret")
install.packages("dplyr")
install.packages("ggplot2")
install.packages("e1071")
install.packages('tidyverse')
install.packages("corrplot")
install.packages("DataExplorer")
```

The libraries selected contain tools for data pre-processing, feature selection, data splitting and unsupervised machine learning algorithms, etc. The next step is to install library packages

```
library(caret)
library(dplyr)
library(ggplot2)
library(e1071)
library(tidyverse)
library(corrplot)
library(DataExplorer)
```

Our next step is to load the selected dataset into R using the code below:

Load the dataset: water potability <-read.csv("C:/Users/Famubukky/OneDrive - Teesside University/Desktop/MY DATASET/water_potability.csv", header = TRUE). We are reading this dataset with read.csv because the selected dataset is in CSV format. The dataset has required attributes like Ph, Chloramines, Solids, Potability e.t.c that is suitable for analysing water quality. Potability column is used as dependent variable to analyse portable water. The diagram below shows how our dataset looks like after loading it into R.

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
1	NA	204.8905	20791.319	7.300212	368.5164	564.3087	10.379783	86.99097	2.963135	0
2	3.716080	129.4229	18630.058	6.635246	NA	592.8854	15.180013	56.32908	4.500656	0
3	8.099124	224.2363	19909.542	9.275884	NA	418.6062	16.868637	66.42009	3.055934	0
4	8.316766	214.3734	22018.417	8.059332	356.8861	363.2665	18.436524	100.34167	4.628771	0
5	9.092223	181.1015	17978.986	6.546600	310.1357	398.4108	11.558279	31.99799	4.075075	0
6	5.584087	188.3133	28748.688	7.544869	326.6784	280.4679	8.399735	54.91786	2.559708	0

Showing 1 to 6 of 3,276 entries, 10 total columns

This is the diagram of our dataset after loading it into R. This dataset has 10 attributes and the last attribute will be use as a dependent variable.

Let's check the structure of our dataset

```
> str(water_potability)
'data.frame':	3276 obs. of  10 variables:
 $ ph             : num  NA 3.72 8.1 8.32 9.09 ...
 $ Hardness       : num  205 129 224 214 181 ...
 $ Solids         : num  20791 18630 19910 22018 17979 ...
 $ Chloramines    : num  7.3 6.64 9.28 8.06 6.55 ...
 $ Sulfate        : num  369 NA NA 357 310 ...
 $ Conductivity   : num  564 593 419 363 398 ...
 $ Organic_carbon : num  10.4 15.2 16.9 18.4 11.6 ...
 $ Trihalomethanes: num  87 56.3 66.4 100.3 32 ...
 $ Turbidity      : num  2.96 4.5 3.06 4.63 4.08 ...
 $ Potability     : int  0 0 0 0 0 0 0 0 0 0 ...
```

The next step is to factorise our dataset:

```
> water_potability$Potability <- as.factor(water_potability$Potability)
```

After factorisation, we need to clean our data before we build the training model for the dataset. We must check if there are missing values and clean it.

```
> #Remove missing values
> complete.cases(water_potability)
   [1] FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE
  [14] FALSE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE  TRUE FALSE FALSE  TRUE  TRUE
  [27]  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE
  [40]  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE
  [53]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE
  [66] FALSE  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE
  [79]  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE
  [92] FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE FALSE
 [105] FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE
 [118]  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE
 [131]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE
 [144]  TRUE  TRUE  TRUE FALSE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE
 [157] FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE FALSE
 [170]  TRUE  TRUE FALSE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE
 [183]  TRUE  TRUE  TRUE FALSE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
 [196] FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE  TRUE
 [209]  TRUE  TRUE  TRUE FALSE  TRUE FALSE FALSE FALSE  TRUE FALSE  TRUE  TRUE FALSE
 [222]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE
```

```
> na_values <- which(!complete.cases(water_potability))
> water <- water_potability[-na_values, ]
```

We use anyNA() method to check if there is any missing value after cleaning the dataset. On running the code, it brings out false which indicate there are no missing value after cleaning.

```
> anyNA(water)
[1] FALSE
```

Let's check the summary of our dataset after cleaning by using summary function()

```
> summary(water)
       ph             Hardness          Solids         Chloramines        Sulfate
 Min.   : 0.2275   Min.   : 73.49   Min.   :  320.9   Min.   : 1.391   Min.   :129.0
 1st Qu.: 6.0897   1st Qu.:176.74   1st Qu.:15615.7   1st Qu.: 6.139   1st Qu.:307.6
 Median : 7.0273   Median :197.19   Median :20933.5   Median : 7.144   Median :332.2
 Mean   : 7.0860   Mean   :195.97   Mean   :21917.4   Mean   : 7.134   Mean   :333.2
 3rd Qu.: 8.0530   3rd Qu.:216.44   3rd Qu.:27182.6   3rd Qu.: 8.110   3rd Qu.:359.3
 Max.   :14.0000   Max.   :317.34   Max.   :56488.7   Max.   :13.127   Max.   :481.0
  Conductivity   Organic_carbon  Trihalomethanes     Turbidity     Potability
 Min.   :201.6   Min.   : 2.20   Min.   :  8.577   Min.   :1.450   0:1200
 1st Qu.:366.7   1st Qu.:12.12   1st Qu.: 55.953   1st Qu.:3.443   1: 811
 Median :423.5   Median :14.32   Median : 66.542   Median :3.968
 Mean   :426.5   Mean   :14.36   Mean   : 66.401   Mean   :3.970
 3rd Qu.:482.4   3rd Qu.:16.68   3rd Qu.: 77.292   3rd Qu.:4.514
 Max.   :753.3   Max.   :27.01   Max.   :124.000   Max.   :6.495
```

There is need to normalise our dataset to change the numeric columns values to a common scale

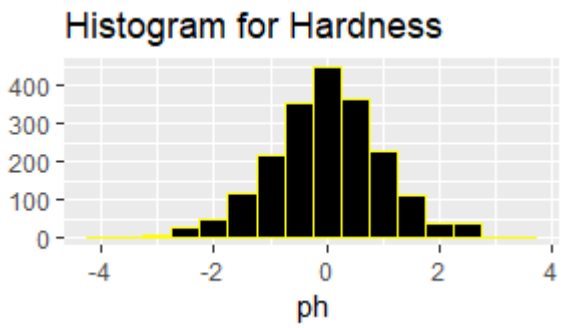
```
> #normalize data
> p1<-preProcess(water[,c(1:10)], method=c("center", "scale"))
> water1 <- predict(p1, water[,c(1:10)])
```

Let's us check the summary of our dataset after normalisation

```
> summary(water1)
       ph             Hardness           Solids         Chloramines
 Min.   :-4.3592   Min.   :-3.7529   Min.   :-2.4989   Min.   :-3.624051
 1st Qu.:-0.6332   1st Qu.:-0.5890   1st Qu.:-0.7292   1st Qu.:-0.628111
 Median :-0.0373   Median : 0.0375   Median :-0.1139   Median : 0.006037
 Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.000000
 3rd Qu.: 0.6146   3rd Qu.: 0.6273   3rd Qu.: 0.6092   3rd Qu.: 0.615456
 Max.   : 4.3945   Max.   : 3.7190   Max.   : 4.0003   Max.   : 3.781289
    Sulfate          Conductivity      Organic_carbon     Trihalomethanes
 Min.   :-4.95629   Min.   :-2.78651   Min.   :-3.65650   Min.   :-3.596657
 1st Qu.:-0.62109   1st Qu.:-0.74147   1st Qu.:-0.67177   1st Qu.:-0.649880
 Median :-0.02409   Median :-0.03804   Median :-0.01073   Median : 0.008791
 Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.00000   Mean   : 0.000000
 3rd Qu.: 0.63356   3rd Qu.: 0.69192   3rd Qu.: 0.69936   3rd Qu.: 0.677427
 Max.   : 3.58707   Max.   : 4.04914   Max.   : 3.80426   Max.   : 3.582680
   Turbidity         Potability
 Min.   :-3.228989   0:1200
 1st Qu.:-0.675102   1: 811
 Median :-0.001988
 Mean   : 0.000000
 3rd Qu.: 0.697699
 Max.   : 3.235769
```

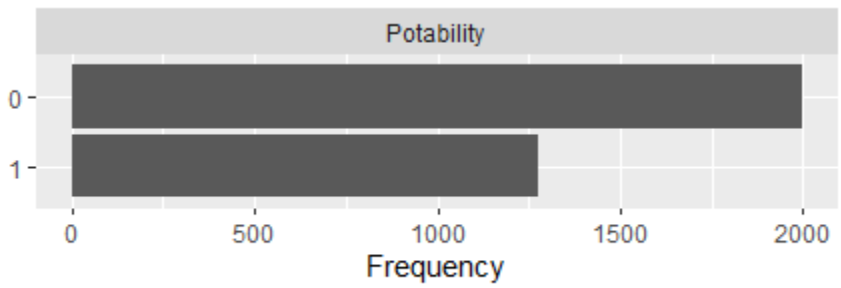
Performing Eploratory Data Analysis on one of the attribute(Hardness) in our dataset.

```
> #Perform Eploratory Data Analysis
> qplot(water1$Hardness,
+       geom="histogram",
+       binwidth = 0.5,
+       main = "Histogram for Hardness",
+       xlab = "ph",
+       fill=I("black"),
+       col=I("yellow"),
+       )
>
```



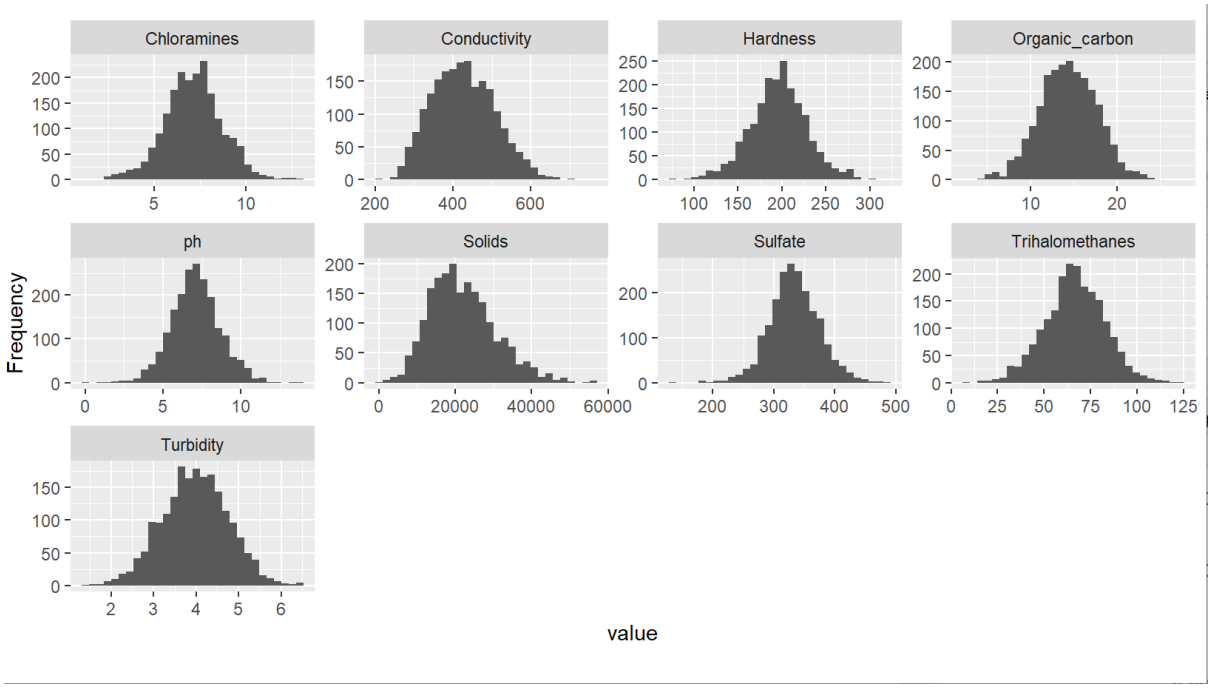
Taking the plot_bar of our dataset before Sampling

```
> plot_bar(water_potability)
```



Before Sampling

```
> plot_histogram(water)
```

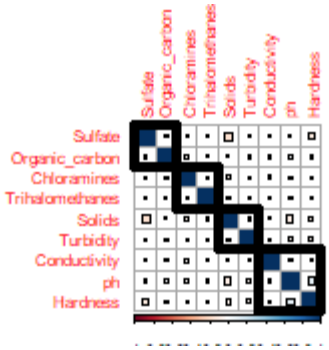


Plotting the histogram of water

Correlation Analysis

To extract the possible relationship between the parameters correction analysis is been used. Dependent variables which is potability will be use to predict estimate variable easily through attainable parameters

```
> #Corerelation plot
> correlations <- cor(water1[,1:9])
> corrplot(correlations,
+             method = "square",
+                   outline = T,
+                   addgrid.col = "darkgray",
+                   order="hclust",
+                   mar = c(0,0,0,2),
+                   addrect = 4,
+                   rect.col = "black",
+                   rect.lwd = 5,
+                   cl.pos = "b",
+                   tl.col = "red",
+                   tl.cex = 0.5,
+                   cl.cex = 0.5)# find attributes that are highly correct
ed
>
```

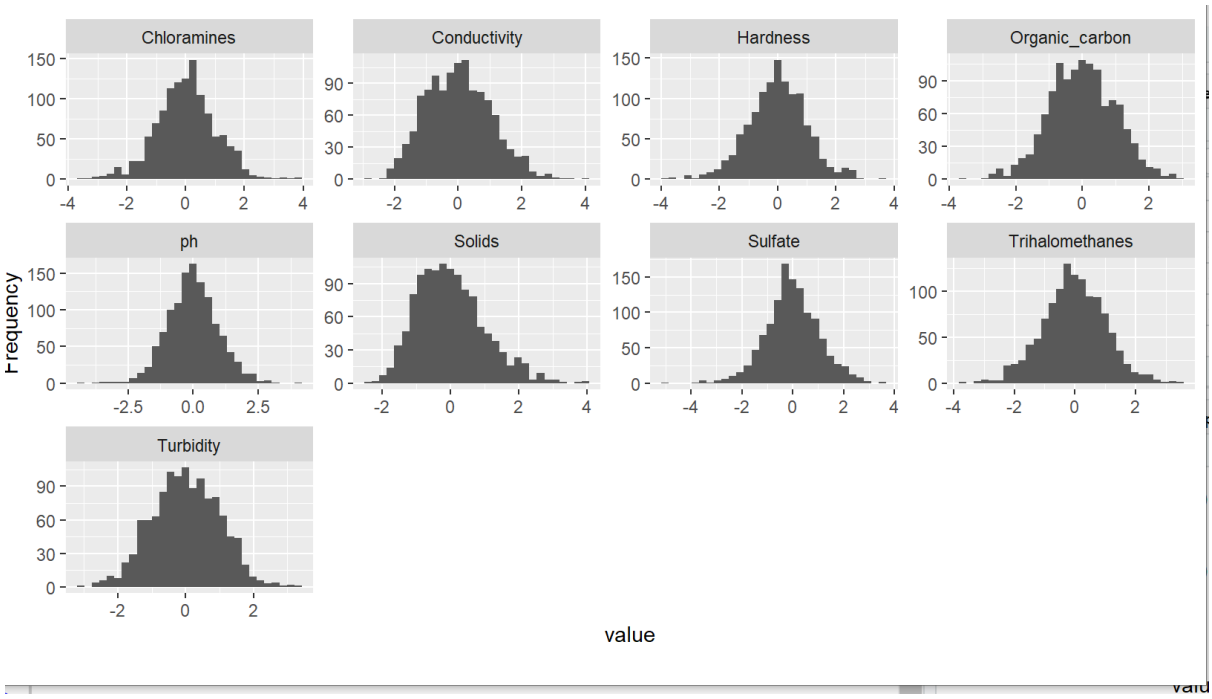



```
> highlyCorrelated <- findCorrelation(correlations, cutoff = 0.6, verbose
 = TRUE, names = TRUE)
All correlations <= 0.6
> highlyCorrelated
character(0)
```

Taking water sampling for our dataset. At set. seed(777) means that it should take the same homogeinous sample everytime we run our sample. That means it is not going to take the sample randomly. 600 datasets were selected for sampling

```
> #sampling
> set.seed(777)
> water3 <- water1 %>%
+   group_by(Potability) %>%
+   sample_n(600)
>
```

```
> plot_histogram(water3)
```



Taking histogram of the dataset after sampling

```
> plot_bar(water3)
>
```

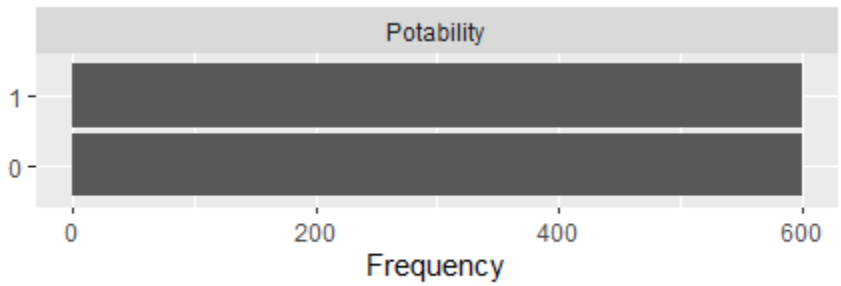


Diagram: After Sampling

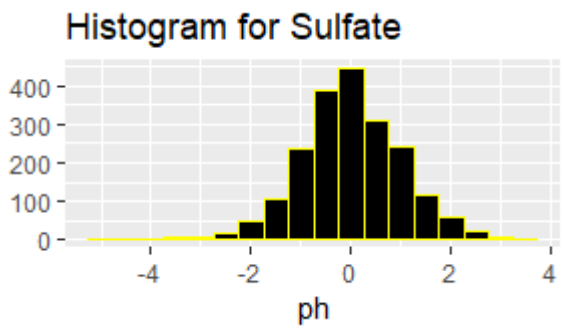
CHAPTER FOUR

4.0 Performance Evaluation

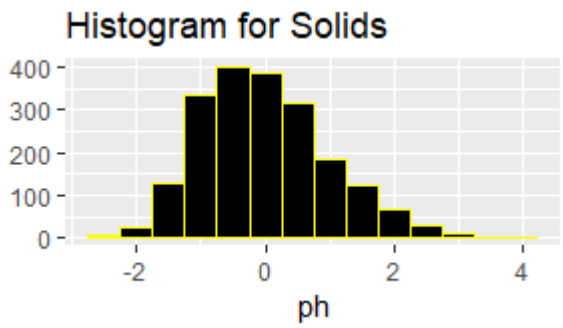
4.1 SHOWING GPLOTS FOR SOME OF THE ATTRIBUTES

Each diagram shows the nature of the dataset.

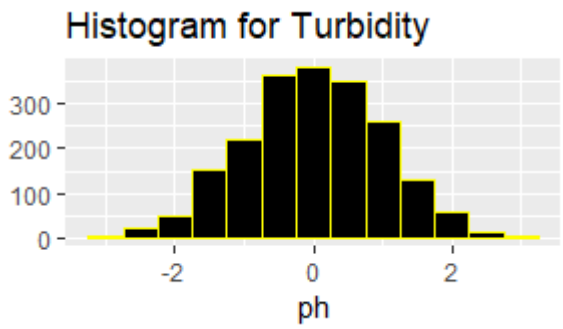
```
> qplot(water1$Sulfate,
+       geom="histogram",
+       binwidth = 0.5,
+       main = "Histogram for Sulfate",
+       xlab = "ph",
+       fill=I("black"),
+       col=I("yellow"),
+ )
>
```




```
> qplot(water1$Solids,
+       geom="histogram",
+       binwidth = 0.5,
+       main = "Histogram for Solids",
+       xlab = "ph",
+       fill=I("black"),
+       col=I("yellow"),
+ )
>
```




```
> qplot(water1$Turbidity,
+       geom="histogram",
+       binwidth = 0.5,
+       main = "Histogram for Turbidity",
+       xlab = "ph",
+       fill=I("black"),
+       col=I("yellow"),
+ )
>
```



4.2 DATA SPLICING

```
> #train dataset
> intrain <- createDataPartition(y = water3$Potability, p = 0.7, list = FA
LSE)
> View(intrain)
> training <- water3[intrain, ]
> testing <- water3[-intrain, ]
```

Our next set is to split the data into two namely training and testing. Training set is used in model building while training set is for evaluation of the model.

Caret package will help in createDataPartition() which is used in partitioning of our dataset into testing and training set.

Three parameters will be used for createDataPartition() function:

The “y” parameter is used for the dependent variable

The “P” parameter will be used for the percentage of the split. We are going to split our data into two 70:30. 70% for training set and 30% for testing set.

The list parameter is used to return whether to return list or matrix. We use false for not returning a list.

```
> trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
```

The train control method

We have using 3 parameters for the train control method:

Method: This parameter makes use of resampling method. In this demo we are going to use repeated cv method.

Number: This is used to hold number of resampling iterations

Repeat: This parameter can used to compute sets to for our repeated cross-validation. We are going to put our setting number =10 and repeat = 3.

```
> svm_Linear <- train(Potability ~., data = training, method = "svmLinea
r",trControl=trctrl,preProcess = c("center", "scale"),tuneLength = 10)
```

The “e1071” package installed will help us in clustering of our dataset. The train control method used earlier returns a list. This will be pass to our train method().

We are going to pass the train method() parameter as “Svmlinear”. The dependent variable(potability) will use all columns as a classifier. The result of our train control method will be passed with “trcontrol” parameter. Pre-processing of our training data is called preprocess.

“center” and “scale” values will be passed into our preprocess parameter. The two values will help for scaling and centering the data.

Our training data will be converted to mean 0 and standard deviation 1 after preprocessing. The parameter to be used for turning our algorithm is called “tunelength” and it holds an integer value.

The result of our train method will be saved in Svm_Linear Variable.

```
> svm_Linear
Support Vector Machines with Linear Kernel

840 samples
  9 predictor
  2 classes: '0', '1'

Pre-processing: centered (9), scaled (9)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 756, 756, 756, 756, 756, 756, ...
Resampling results:

  Accuracy   Kappa
  0.5031746  0.006349206

Tuning parameter 'C' was held constant at a value of 1
```

This is a linear method it tested at value “C” = 1. Since our model is trained at value c =1.

We can use predict method to predict classes for our test set.

The caret package installed earlier gives predict method() for predicting results. Two arguments will be passed “trained model” and “newdata”. Newdata will be used to hold our data frame testing. Predict method will return a list, we are going to save it in test_pred variable.

```
> test_pred <- predict(svm_Linear, newdata = testing)
> test_pred
  [1] 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 0 1 1 0 0 0 0 1 1 0 0 1 1 1 0 1 1
 [34] 1 1 1 0 1 1 1 0 0 0 0 1 0 1 1 1 0 1 1 1 0 0 0 1 1 1 0 0 0 0 1 0 1
 [67] 1 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 0 1 0 0 0 1 0
[100] 1 1 0 0 1 0 0 1 0 1 1 1 0 1 1 1 0 0 1 1 1 0 0 1 1 0 1 1 1 1 0 0 1
[133] 1 0 0 1 0 1 0 0 1 1 0 0 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0 0 1 0 0 0 1
[166] 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 1 1 1 0 1 1 0 1 1 0 0 0 1 0 1 0 0 0
[199] 0 0 0 1 0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 1 0 1 1 0 1 1 1 0 0 0 1 1 1
[232] 0 0 0 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 1 0 0 0 1
[265] 0 0 0 1 1 1 1 0 1 1 0 0 1 0 1 0 1 0 1 1 0 1 0 0 0 0 0 1 1 1 0 0 0
[298] 1 0 1 0 0 0 1 0 1 0 1 1 0 0 1 1 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 0
[331] 0 1 0 0 1 0 0 0 0 1 1 0 1 0 0 1 1 1 0 0 1 0 0 1 1 0 1 0 1 1
Levels: 0 1
```

We are going to check the accuracy of our model. Confusion matrix will be used to predict the accuracy.

```
> confusionMatrix(table(test_pred, testing$Potability))
Confusion Matrix and Statistics
```

```
test_pred  0  1
        0 89 98
        1 91 82

               Accuracy : 0.475
                 95% CI : (0.4224, 0.528)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 0.8417

                  Kappa : -0.05

 Mcnemar's Test P-Value : 0.6625

            Sensitivity : 0.4944
            Specificity : 0.4556
         Pos Pred Value : 0.4759
         Neg Pred Value : 0.4740
             Prevalence : 0.5000
         Detection Rate : 0.2472
   Detection Prevalence : 0.5194
      Balanced Accuracy : 0.4750

       'Positive' Class : 0
```

The output display of our model accuracy for test set is 47.50%

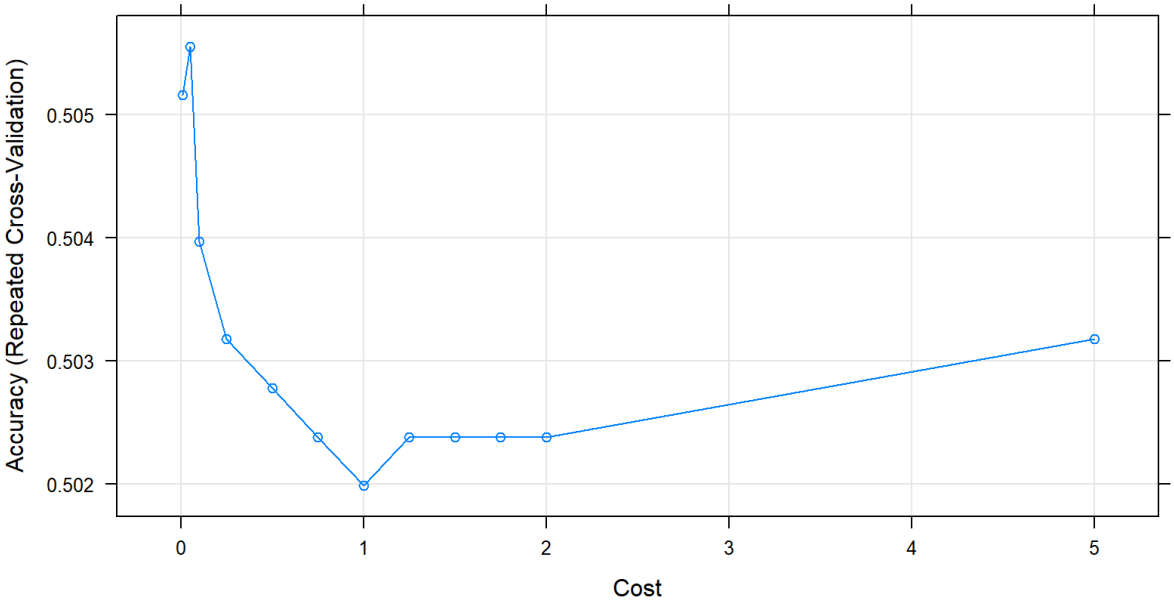
The above example shows that we can build svm linear classifier. We can enter some values of C into grid dataframe by using expand.grid(). We can now use our classifier for testing dataframe at a specific values of C. We need to put it in tuneGrid parameter with train method

```
> grid <- expand.grid(C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25,
 1.5, 1.75, 2,5))
```

```
> svm_Linear_Grid <- train(Potability ~., data = training, method = "svmLi
near",trControl=trctrl,preProcess = c("center", "scale"),tuneGrid = grid,t
uneLength= 10)
There were 32 warnings (use warnings() to see them)
> svm_Linear_Grid
Support Vector Machines with Linear Kernel

840 samples
  9 predictor
  2 classes: '0', '1'

Pre-processing: centered (9), scaled (9)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 756, 756, 756, 756, 756, 756, ...
Resampling results across tuning parameters:
```

```
  C     Accuracy   Kappa
  0.00        NaN         NaN
  0.01  0.5051587  0.010317460
  0.05  0.5055556  0.011111111
  0.10  0.5039683  0.007936508
  0.25  0.5031746  0.006349206
  0.50  0.5027778  0.005555556
  0.75  0.5023810  0.004761905
  1.00  0.5019841  0.003968254
  1.25  0.5023810  0.004761905
  1.50  0.5023810  0.004761905
  1.75  0.5023810  0.004761905
  2.00  0.5023810  0.004761905
  5.00  0.5031746  0.006349206

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was C = 0.05.
```

```
> plot(svm_Linear_Grid)
```



The diagram above shows that our classifier is has best accuracy on C = 0.05. We can now make prediction for our test set using this model.

```
> test_pred_grid <- predict(svm_Linear_Grid, newdata = testing)
> test_pred_grid
  [1] 1 1 1 1 1 1 0 0 0 1 1 0 0 0 1 1 0 1 1 0 0 0 0 1 1 0 0 1 1 1 0 1 1
 [34] 1 1 1 0 1 1 1 0 0 0 0 1 1 1 1 1 0 1 1 1 0 0 0 1 1 1 0 0 0 0 1 0 1
 [67] 1 0 1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 0 1 0 0 0 1 1 1 0 0 1 0 0 0 1 0
[100] 1 1 0 0 1 0 0 1 0 1 1 1 0 1 1 1 0 0 1 1 1 0 0 1 1 0 1 1 1 1 0 0 1
[133] 1 0 0 1 0 1 0 0 1 1 0 1 1 1 0 0 0 1 1 0 0 0 1 0 0 0 0 0 1 0 0 0 1
[166] 0 0 1 0 1 0 1 0 1 0 0 0 0 0 0 1 1 1 0 1 1 0 1 1 0 0 0 1 0 1 0 0 0
[199] 0 0 0 1 0 1 1 0 1 0 1 1 1 1 1 0 0 0 0 1 0 1 1 0 1 1 1 0 0 0 1 1 1
[232] 0 0 0 1 1 1 1 0 0 1 0 0 1 0 0 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 1
[265] 0 0 0 1 1 1 1 0 1 1 0 0 1 0 1 0 1 0 1 1 0 1 0 0 0 0 0 1 1 1 0 0 0
[298] 1 0 1 0 0 0 1 0 1 0 1 1 0 0 1 1 0 0 1 0 0 0 1 0 1 0 0 1 0 0 0 0 0
[331] 0 1 0 0 1 0 0 0 0 1 1 0 0 0 0 1 1 1 0 0 1 0 0 1 1 0 1 0 1 1
Levels: 0 1
```

We can check the accuracy by using confusion-matrix

```
> confusionMatrix(table(test_pred_grid, testing$Potability))
Confusion Matrix and Statistics


test_pred_grid   0   1
             0  87 100
             1  93  80

               Accuracy : 0.4639
                 95% CI : (0.4115, 0.5169)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 0.9227

                  Kappa : -0.0722



 Mcnemar's Test P-Value : 0.6658

            Sensitivity : 0.4833
            Specificity : 0.4444
         Pos Pred Value : 0.4652
         Neg Pred Value : 0.4624
             Prevalence : 0.5000
         Detection Rate : 0.2417
   Detection Prevalence : 0.5194
      Balanced Accuracy : 0.4639

       'Positive' Class : 0
```

The results of this confusion matrix shows that, the accuracy on the test set is 46.39% which is a little bit reduced compared to the first one. Because of this we are going to remove the Set.Seed(777) for sampling. We are going to show sampling randomly.

Taking Sampling randomly

```
> #sampling
> water3 <- water1 %>%
+   group_by(Potability) %>%
+   sample_n(600)
> plot_bar(water3)
> plot_bar(water_potability)
> plot_histogram(water3)
> qplot(water1$Turbidity,
+       geom="histogram",
+       binwidth = 0.5,
+       main = "Histogram for Turbidity",
+       xlab = "ph",
+       fill=I("black"),
+       col=I("yellow"),
+ )
```

```
> #train dataset
> intrain <- createDataPartition(y = water3$Potability, p = 0.7, list = FA
LSE)
> training <- water3[intrain, ]
> testing <- water3[-intrain, ]
> trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)
> svm_Linear <- train(Potability ~., data = training, method = "svmLinea
r",trControl=trctrl,preProcess = c("center", "scale"),tuneLength = 10)
```

```
> svm_Linear
Support Vector Machines with Linear Kernel

840 samples
  9 predictor
  2 classes: '0', '1'

Pre-processing: centered (9), scaled (9)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 756, 756, 756, 756, 756, 756, ...
Resampling results:

  Accuracy  Kappa
  0.502381  0.004761905

Tuning parameter 'C' was held constant at a value of 1
```

```
> test_pred <- predict(svm_Linear, newdata = testing)
> test_pred
  [1] 1 0 0 0 1 1 0 1 0 1 0 0 1 1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 1 1
 [34] 0 0 1 0 0 0 1 1 0 1 0 1 0 1 1 1 1 0 1 0 0 0 0 0 0 0 0 1 1 1 0 1 0
 [67] 0 0 0 1 0 1 0 0 0 1 1 1 0 0 1 1 1 1 1 0 1 1 0 1 0 0 1 0 1 1 0 0 0
[100] 0 1 0 1 0 1 1 0 1 0 0 1 0 1 0 0 1 1 1 1 0 0 0 1 0 1 1 0 1 0 0 0 0
[133] 1 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 1 1 1 0
[166] 0 0 1 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 1 1 0 0
[199] 1 1 1 0 1 1 0 1 0 0 0 1 1 0 1 1 1 0 1 1 1 0 0 0 0 0 0 0 1 0 0 1 0
[232] 1 0 1 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 1 0 0 1 1 1
[265] 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 0 0 1 0 0 1 0 0 1 0 1 0 0 0 1 0 0
[298] 0 0 0 0 1 1 1 1 0 1 1 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 1 0 0
[331] 0 0 0 0 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 0 0 1 1 0 1 1 0 1 0
Levels: 0 1
```

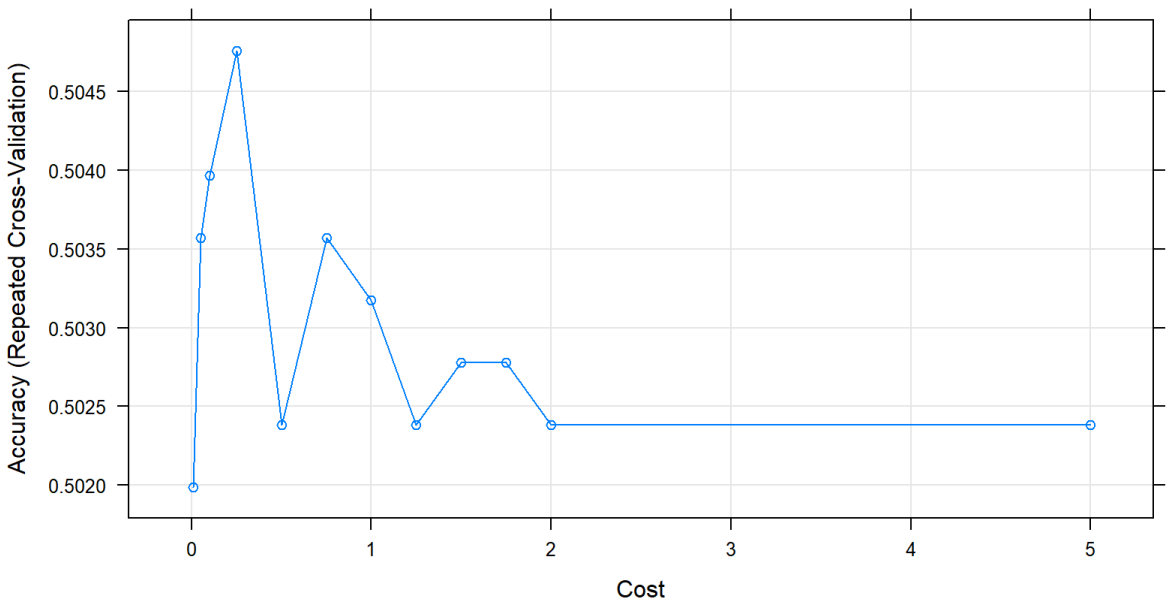
```
> confusionMatrix(table(test_pred, testing$Potability))
Confusion Matrix and Statistics


test_pred  0  1
        0 99 95
        1 81 85

               Accuracy : 0.5111
                 95% CI : (0.4582, 0.5639)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 0.3561

                  Kappa : 0.0222

                   . .

 Mcnemar's Test P-Value : 0.3271

            Sensitivity : 0.5500
            Specificity : 0.4722
         Pos Pred Value : 0.5103
         Neg Pred Value : 0.5120
             Prevalence : 0.5000
         Detection Rate : 0.2750
   Detection Prevalence : 0.5389
      Balanced Accuracy : 0.5111

       'Positive' Class : 0
```

The output display of our model accuracy for test set is 51.11%

```
> grid <- expand.grid(C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25,
 1.5, 1.75, 2,5))
> svm_Linear_Grid <- train(Potability ~., data = training, method = "svmLi
near",trControl=trctrl,preProcess = c("center", "scale"),tuneGrid = grid,t
uneLength= 10)
There were 32 warnings (use warnings() to see them)
> svm_Linear_Grid
Support Vector Machines with Linear Kernel

840 samples
  9 predictor
  2 classes: '0', '1'

Pre-processing: centered (9), scaled (9)
Resampling: Cross-Validated (10 fold, repeated 3 times)
Summary of sample sizes: 756, 756, 756, 756, 756, 756, ...
Resampling results across tuning parameters:
```

```
  C     Accuracy   Kappa
  0.00       NaN          NaN
  0.01  0.5019841  0.003968254
  0.05  0.5035714  0.007142857
  0.10  0.5039683  0.007936508
  0.25  0.5047619  0.009523810
  0.50  0.5023810  0.004761905
  0.75  0.5035714  0.007142857
  1.00  0.5031746  0.006349206
  1.25  0.5023810  0.004761905
  1.50  0.5027778  0.005555556
  1.75  0.5027778  0.005555556
  2.00  0.5023810  0.004761905
  5.00  0.5023810  0.004761905

Accuracy was used to select the optimal model using the largest value.
The final value used for the model was C = 0.25.
```

```
> plot(svm_Linear_Grid)
```




```
> test_pred_grid <- predict(svm_Linear_Grid, newdata = testing)
> test_pred_grid
  [1] 1 0 0 0 1 1 0 1 0 1 0 0 1 1 0 1 0 0 0 0 0 0 1 0 1 0 0 0 0 1 0 1 1
 [34] 0 0 1 0 0 0 1 1 0 1 0 1 0 1 1 1 1 0 1 0 0 0 0 0 0 0 0 1 1 1 0 1 0
 [67] 0 0 0 1 0 1 0 0 0 1 1 1 0 0 1 1 1 1 1 0 1 1 0 1 0 0 1 0 1 1 0 0 0
[100] 0 1 0 1 0 1 1 0 1 0 0 1 0 1 0 0 1 1 1 1 0 0 0 1 0 1 1 0 1 0 0 0 0
[133] 1 0 0 0 1 0 1 0 0 1 0 1 1 0 1 1 0 0 1 0 0 1 0 1 0 1 0 0 0 1 1 1 0
[166] 0 0 1 1 1 1 0 1 1 0 0 1 1 0 0 1 1 1 0 1 1 0 0 1 0 0 0 0 1 1 1 0 0
[199] 1 1 1 0 1 1 0 1 0 0 0 1 1 0 1 1 1 0 1 1 1 0 0 0 0 0 0 0 1 0 0 1 0
[232] 1 0 1 0 0 1 0 0 0 0 1 1 1 1 0 1 0 0 0 1 0 0 1 0 1 0 0 1 0 0 1 1 1
[265] 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 0 0 0 1 0 0 1 0 0 1 0 1 0 0 0 1 0 0
[298] 0 0 0 0 1 1 1 1 0 1 1 0 0 1 0 0 0 1 0 1 0 1 0 0 0 0 0 1 1 0 1 0 0
[331] 0 0 0 0 1 1 0 1 1 1 1 1 1 1 0 0 0 1 1 1 0 0 1 1 0 1 1 0 1 0
Levels: 0 1
```

```
> confusionMatrix(table(test_pred_grid, testing$Potability))
Confusion Matrix and Statistics


test_pred_grid  0  1
             0 99 95
             1 81 85

               Accuracy : 0.5111
                 95% CI : (0.4582, 0.5639)
    No Information Rate : 0.5
    P-Value [Acc > NIR] : 0.3561

                  Kappa : 0.0222


 Mcnemar's Test P-Value : 0.3271

            Sensitivity : 0.5500
            Specificity : 0.4722
         Pos Pred Value : 0.5103
         Neg Pred Value : 0.5120
             Prevalence : 0.5000
         Detection Rate : 0.2750
   Detection Prevalence : 0.5389
      Balanced Accuracy : 0.5111

       'Positive' Class : 0
```

The results of this confusion matrix is the same thing with the accuracy on the test set is 51.11%.

Technical discussions and conclusions based on your findings.

It is very important to model and predict for water quality for the protection of the environment. Developing a model such as Support Vector Machine algorithms can be used to measure the future of water quality. However, the SVM algorithm has proven to be the highest accuracy of the prediction of the water quality. In future work, the developed model is reliable and effective to implement the prediction of water quality for different types of water.

Comparing the sampling analysis of random sampling and Set. Seed(777) of sampling. It is better to take sample randomly which shows better result for the accuracy than setting random limit for sampling.

References

Alghamdi A.G., Aly A.A., Aldhumri S. A and Al-Barakaha F. N. (2020). Hydrochemical quality assessment of groundwater resources in Al-Madinah City, Western Saudi Arabia. Sustainability 12.

Babu C.N and Reddy B.E. (2014). "A moving-average filter based hybrid ARIMA-ANN model for forecasting time series data," *Applied Soft Computing*, 23:27–38.

Batista, G.E.A.P.A. and Monard, M.C. (2002) K-Nearest Neighbour as Imputation Method: Experimental Results. Technical Report, ICMC-USP.

Cortes, C. and Vapnik, V. (1995). Support-Vector Networks. Machine Learning, 20, 273-297. https://doi.org/10.1007/BF00994018

Farrell-Poe K., Payne W., and Emanuel R., (2000). Water Quality &Monitoring, University of Arizona Repository, http://hdl.handle.net/10150/146901.

https://www.edureka.co/

https://www.kaggle.com/

Likas, Aristidis, Nikos Vlassis, and Jakob J. Verbeek., (2003) "The global k-means clustering algorithm." *Pattern Recognition* (Elsevier) 36(2): 451–461.

Liu X, Cheng G, Wang B, Lin S (2012) Optimum design of pile foundation by automatic grouping genetic algorithms. ISRN Civil Engineering.

Lucà, F., Conforti, M., Castrignanò, A., Matteucci, G. and Buttafuoco, G., (2017) Effect of calibration set size on prediction at local scale of soil carbon by Vis-NIR spectroscopy. Geoderma 288, 175–183.

Mustafa, A., Scholz, M., Khan, S. and Ghaffar, A., (2013). Application of solar disinfection for treatment of contaminated public water supply in a developing country: field observations. Journal of water and health, 11(1), pp.135-145.

Nawar, S., Buddenbaum, H., Hill, J., Kozak, J. & Mouazen, A. M.,(2016). Estimating the soil clay content and organic matter bymeans of different calibration methods of vis-NIR diffuse reflectance spectroscopy. Soil and Tillage Research 155,510–522.

Shafi, U., Mumtaz, R., Anwar, H., Qamar, A.M., Khurshid, H., (2018). Surface Water Pollution Detection using Internet of Things. In Proceedings of the 2018 15th International Conference on Smart Cities: Improving Quality of Life Using ICT & IoT (HONET-ICT), Islamabad, Pakistan, 8–10 October pp. 92–96.

Somani, P. D., Ray, S. and Singh, S. (2014). Assessment of water quality. International Journal of Scientific and Engineering Research, 5(12).

Taskaya-Temizel T. and Casey M. C., (2005). "A comparative study of autoregressive neural network hybrids," Neural Networks, vol. 18, no. 5–6, pp. 781–789.

Taskaya-Temizel, T., Casey, M.-C., & Ahmad, K. (2005). Pre-processing inputs for optimally-configured time-delay neural networks. IEE Electronic Letters, 41, 198–200.

UN water, "Clean water for a healthy world," Development, (2010), https://www.undp.org/content/undp/en/home/presscenter/articles/2010/03/22/clean-water-for-a-healthyworld. html.

United Nation, World Water Development Report, (2017).

United Nations Environment Programme (UNEP,2016)

United Nation-Water, (2010)

United Nation-Water, (2011)

United Nation-Water, (2021)

Wilcox, Catherine, Wei Lee Woon, and Zeyar Aung.,(2013) *Applications of Machine Learning in Environmental Engineering.* Technical Report, Abu Dhabi: Masdar Institute of Science and Technology.

Zeilhofer P., Zeilhofer L. V. A. C., Hardoim E. L., Lima Z. M., and Oliveira C. S., (2007). "GIS applications for mapping and spatial modeling of urban-use water quality: a case study in District of Cuiabá, Mato Grosso, Brazil," Cadernos de Saúde Pública,vol. 23, no. 4, pp. 875–884.

Zhang X., Hu N., Cheng Z., and Zhong H., (2014)."Vibration data recovery based on compressed sensing," Acta Physica Sinica, vol. 63, no. 20, pp. 119–128.

Zhang, H (2004). The optimality of naive Bayes. AA, 1, 3.

Appendices with source code

https://www.kaggle.com/artimule/drinking-water-probability

B1148714.R

```
install.packages("caret")

install.packages("dplyr")

install.packages("ggplot2")

install.packages("e1071")

install.packages('tidyverse')

install.packages("corrplot")

install.packages("DataExplorer")

library(caret)

library(dplyr)

library(ggplot2)

library(e1071)

library(tidyverse)

library(corrplot)

library(DataExplorer)
```

```
water_potability<-read.csv("C:/Users/Famubukky/OneDrive-Teesside University/Desktop/MY DATASET/water_potability.csv", header = TRUE)

str(water_potability)

water_potability$Potability <- as.factor(water_potability$Potability)

#Remove missing values

complete.cases(water_potability)

na_values <- which(!complete.cases(water_potability))

water <- water_potability[-na_values, ]

anyNA(water)

summary(water)



#normalize data

p1<-preProcess(water[,c(1:10)], method=c("center", "scale"))

water1 <- predict(p1, water[,c(1:10)])

summary(water1)

#Perform Eploratory Data Analysis

qplot(water1$Hardness,

    geom="histogram",

    binwidth = 0.5,

    main = "Histogram for Hardness",

    xlab = "ph",
```

```
    fill=I("black"),
    col=I("yellow"),
    )
plot_bar(water_potability)
plot_histogram(water)
#Correlation plot
correlations <- cor(water1[,1:9])
corrplot(correlations,
         method = "square",
             outline = T,
             addgrid.col = "darkgray",
             order="hclust",
             mar = c(0,0,0,2),
             addrect = 4,
             rect.col = "black",
             rect.lwd = 5,
             cl.pos = "b",
             tl.col = "red",
             tl.cex = 0.5,
             cl.cex = 0.5)# find attributes that are highly corrected
highlyCorrelated <- findCorrelation(correlations, cutoff = 0.6, verbose = TRUE, names = TRUE)
```

```
highlyCorrelated



#sampling

set.seed(777)

water3 <- water1 %>%

  group_by(Potability) %>%

  sample_n(600)

plot_bar(water3)

plot_bar(water_potability)

plot_histogram(water3)

qplot(water1$Turbidity,

      geom="histogram",

      binwidth = 0.5,

      main = "Histogram for Turbidity",

      xlab = "ph",

      fill=I("black"),

      col=I("yellow"),

)

#train dataset

intrain <- createDataPartition(y = water3$Potability, p = 0.7, list = FALSE)

training <- water3[intrain, ]
```

```
testing <- water3[-intrain, ]

trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3)

svm_Linear <- train(Potability ~., data = training, method = "svmLinear",trControl=trctrl,preProcess =
c("center", "scale"),tuneLength = 10)

svm_Linear

test_pred <- predict(svm_Linear, newdata = testing)

test_pred

confusionMatrix(table(test_pred, testing$Potability))

grid <- expand.grid(C = c(0,0.01, 0.05, 0.1, 0.25, 0.5, 0.75, 1, 1.25, 1.5, 1.75, 2,5))

svm_Linear_Grid <- train(Potability ~., data = training, method =
"svmLinear",trControl=trctrl,preProcess = c("center", "scale"),tuneGrid = grid,tuneLength= 10)

svm_Linear_Grid

plot(svm_Linear_Grid)

test_pred_grid <- predict(svm_Linear_Grid, newdata = testing)

test_pred_grid

confusionMatrix(table(test_pred_grid, testing$Potability))
```