FINAL REPORT
Water Quality Analysis

Team Members: -

- Nikhil Goutham Budarayavalasa (UCID: ngb4)
- Tetapally Tarun Deshmukh (UCID: tt362)
- Kakkera Sai Deepthi (UCID: sk3452)

Abstract: -

Water is the most precious natural resource. A quarter of the Earth is covered by water and a third by land, but only a small portion of it is usable by living things. Water is required for various purposes such as irrigation and drinking. The water should be analyzed for purity before use. Water contains many elements such as "PH", “calcium”, “potassium", "copper", and "iron". The amount of each element must be within a range, and increased amounts of elements are harmful to drinking.

Purified water should be checked regularly. Poor water quality does not only represent environmental degradation and threats to ecosystems. The industry releases so many harmful gasses into the air and harmful chemicals into water sources such as rivers, ponds, and seas. must be analyzed before use. After years of research, water analysis now consists of several standard protocols. In this study, Gibbs diagrams are used to assess the mechanisms responsible for chemical changes in groundwater.

Relationships between major ions were used to draw attention to the special properties of groundwater. The hydro chemical properties of groundwater-related physicochemical parameters were evaluated using the 1:1 Aquiline diagram. Inverse ion exchange and chloro-alkali indices were performed to assess the presence of ion exchange/reverse ion exchange in the groundwater of this region. The results clearly show that the groundwater in the study area is suitable for irrigation. Studies on trace metal concentrations in water show high concentrations of heavy metals.

The Heavy Metal Contamination Index results support this finding, as they indicate that many groundwater samples were higher than the permissible limit for heavy metal contamination.

Introduction: -

Description of hydrogeochemical properties and assessment of groundwater quality. Multivariate statistical techniques such as factor analysis, hierarchical cluster analysis and correlation analysis using correlation coefficient matrix were used. In the current study, 44 groundwater samples were taken from Arani Taluk in Tamil Nadu, southern India. The results of correlation studies indicate that there is a rock-water interaction. Using cluster analysis, the region was divided into four distinct clusters representing groundwater impacts from activities in agricultural, residential, and industrial areas.

The results of factor analysis differentiated regional highly polluted, moderately polluted and uncontaminated groundwater in terms of major ions and heavy metals, using integrated factor and spatial analysis to indicate the location/ area of distribution of the contaminant.

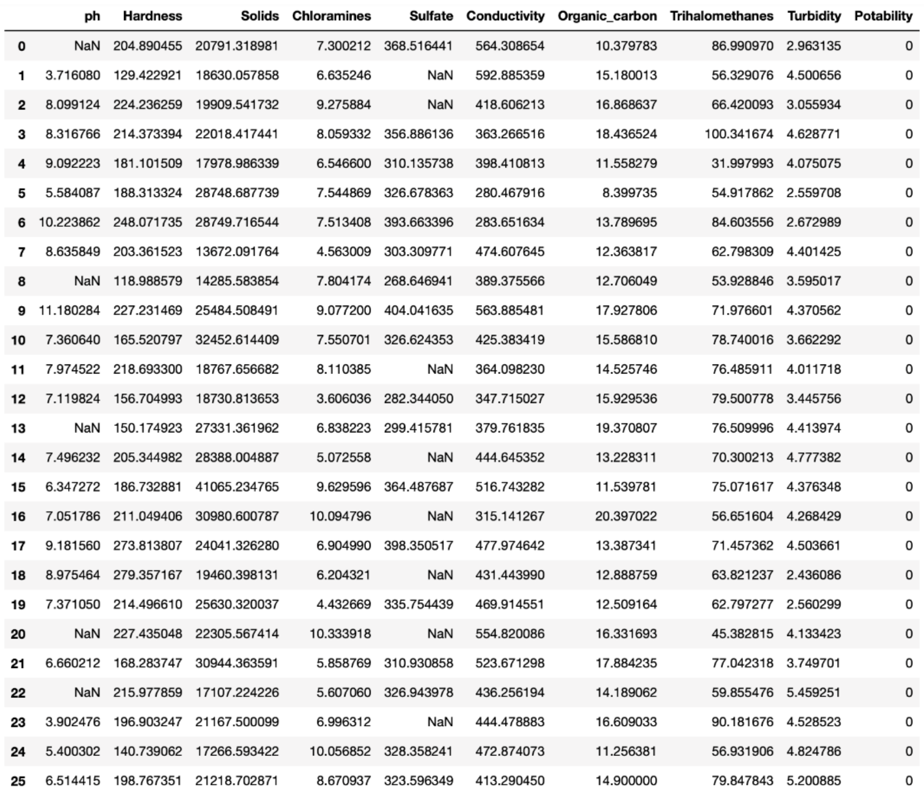
The geochemistry and properties of groundwater are influenced by various geological and anthropogenic processes, and the chemical composition of groundwater, and thus its geochemistry, varies greatly. Alongside major Indian cities and other cities, the region is experiencing significant growth and area expansion as new settlements spawn new urbanized areas. These places don't have the most basic/basic amenities: a sewage system. People are moving into these cities that have only remediation systems, straining the filtration properties of the soil and contaminating the groundwater. There are so many parameters to consider when analyzing water, such as "PH", hardness, solids, chloramines, sulfates, conductivity, organic carbon, trihalomethanes, turbidity. By using some machine learning algorithm, we can know the accuracy rate of the given data. In general, a lot of information from multiple sampling sites are required to assess the chemistry of local groundwater.

The management of large hydro chemical datasets with large variations in physicochemical parameters due to the action of various contaminants/pollutants adds complexity. Many hydrologists and water managers have embraced the use of multivariate statistical analysis in hydro geochemistry as it provides a powerful tool for discovering similarities between physicochemical properties present in water. increase.

Dataset: -

Kaggle data set is used for this analysis. It can be accessed here:

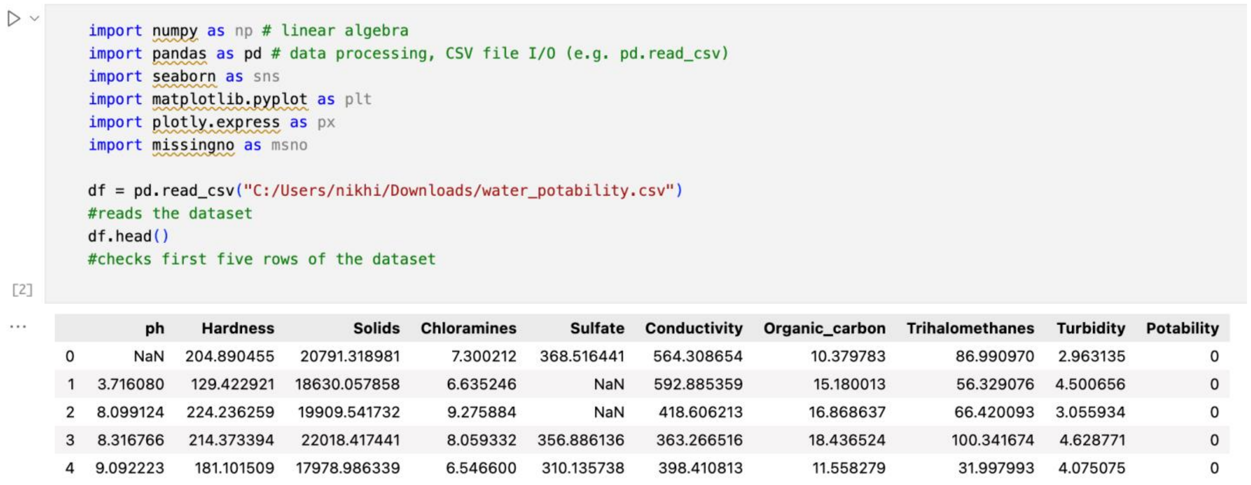
https://www.kaggle.com/code/jaykumar1607/water-quality-analysis-plotly-and-modelling



	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637	66.420093	3.055934	0
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	100.341674	4.628771	0
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0
5	5.584087	188.313324	28748.687739	7.544869	326.678363	280.467916	8.399735	54.917862	2.559708	0
6	10.223862	248.071735	28749.716544	7.513408	393.663396	283.651634	13.789695	84.603556	2.672989	0
7	8.635849	203.361523	13672.091764	4.563009	303.309771	474.607645	12.363817	62.798309	4.401425	0
8	NaN	118.988579	14285.583854	7.804174	268.646941	389.375566	12.706049	53.928846	3.595017	0
9	11.180284	227.231469	25484.508491	9.077200	404.041635	563.885481	17.927806	71.976601	4.370562	0
10	7.360640	165.520797	32452.614409	7.550701	326.624353	425.383419	15.586810	78.740016	3.662292	0
11	7.974522	218.693300	18767.656682	8.110385	NaN	364.098230	14.525746	76.485911	4.011718	0
12	7.119824	156.704993	18730.813653	3.606036	282.344050	347.715027	15.929536	79.500778	3.445756	0
13	NaN	150.174923	27331.361962	6.838223	299.415781	379.761835	19.370807	76.509996	4.413974	0
14	7.496232	205.344982	28388.004887	5.072558	NaN	444.645352	13.228311	70.300213	4.777382	0
15	6.347272	186.732881	41065.234765	9.629596	364.487687	516.743282	11.539781	75.071617	4.376348	0
16	7.051786	211.049406	30980.600787	10.094796	NaN	315.141267	20.397022	56.651604	4.268429	0
17	9.181560	273.813807	24041.326280	6.904990	398.350517	477.974642	13.387341	71.457362	4.503661	0
18	8.975464	279.357167	19460.398131	6.204321	NaN	431.443990	12.888759	63.821237	2.436086	0
19	7.371050	214.496610	25630.320037	4.432669	335.754439	469.914551	12.509164	62.797277	2.560299	0
20	NaN	227.435048	22305.567414	10.333918	NaN	554.820086	16.331693	45.382815	4.133423	0
21	6.660212	168.283747	30944.363591	5.858769	310.930858	523.671298	17.884235	77.042318	3.749701	0
22	NaN	215.977859	17107.224226	5.607060	326.943978	436.256194	14.189062	59.855476	5.459251	0
23	3.902476	196.903247	21167.500099	6.996312	NaN	444.478883	16.609033	90.181676	4.528523	0
24	5.400302	140.739062	17266.593422	10.056852	328.358241	472.874073	11.256381	56.931906	4.824786	0
25	6.514415	198.767351	21218.702871	8.670937	323.596349	413.290450	14.900000	79.847843	5.200885	0

Exploring the data: -

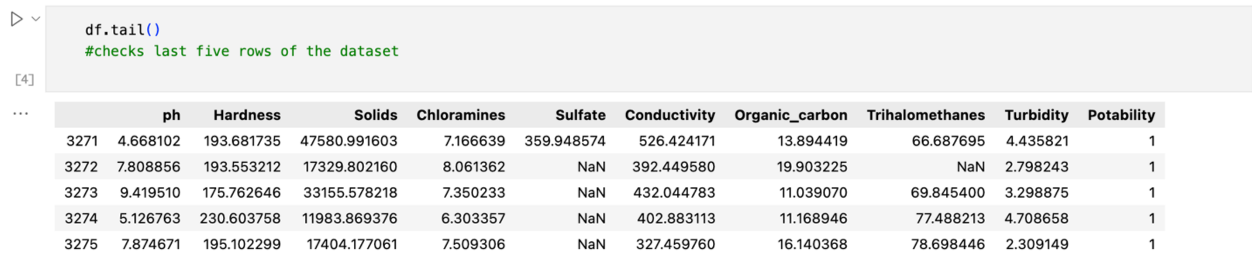
We are exploring the data by using necessary libraries. The most important library is pandas, by using pandas we can read or write the data file. The data file may be a CSV type or EXCEL type. The given below code will explore the data file.



```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import missingno as msno

df = pd.read_csv("C:/Users/nikhi/Downloads/water_potability.csv")
#reads the dataset
df.head()
#checks first five rows of the dataset
```

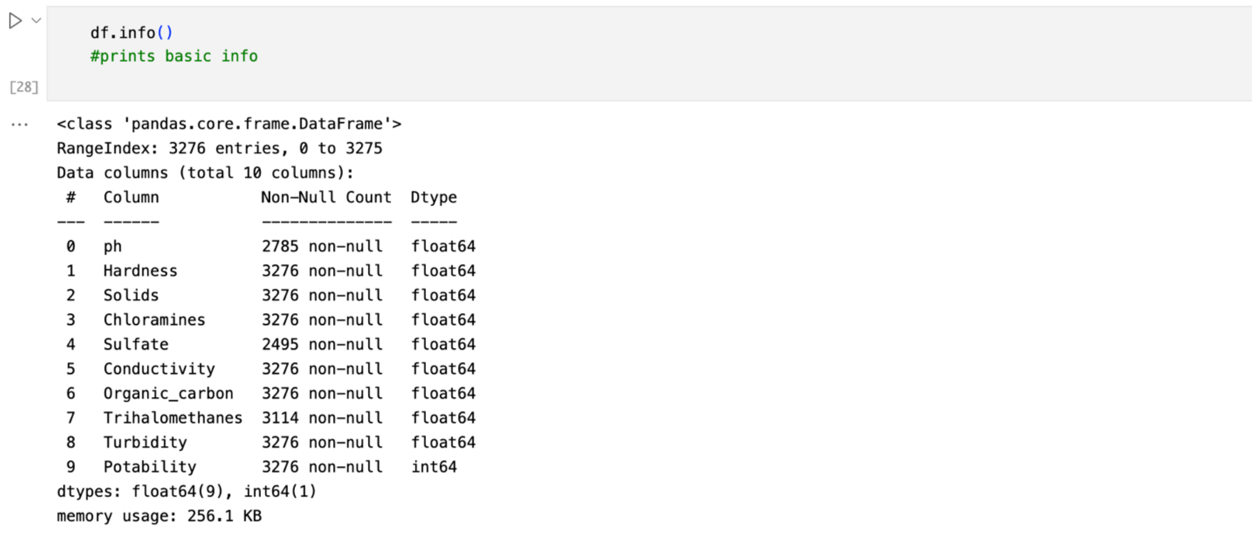
[2]

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637	66.420093	3.055934	0
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	100.341674	4.628771	0
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0



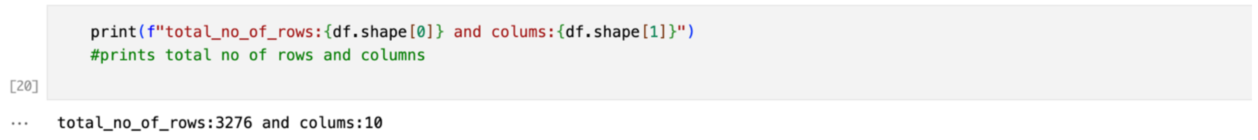
```
df.tail()
#checks last five rows of the dataset
```
[4]

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
3271	4.668102	193.681735	47580.991603	7.166639	359.948574	526.424171	13.894419	66.687695	4.435821	1
3272	7.808856	193.553212	17329.802160	8.061362	NaN	392.449580	19.903225	NaN	2.798243	1
3273	9.419510	175.762646	33155.578218	7.350233	NaN	432.044783	11.039070	69.845400	3.298875	1
3274	5.126763	230.603758	11983.869376	6.303357	NaN	402.883113	11.168946	77.488213	4.708658	1
3275	7.874671	195.102299	17404.177061	7.509306	NaN	327.459760	16.140368	78.698446	2.309149	1



```
df.info()
#prints basic info
```
[28]

```
<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
```



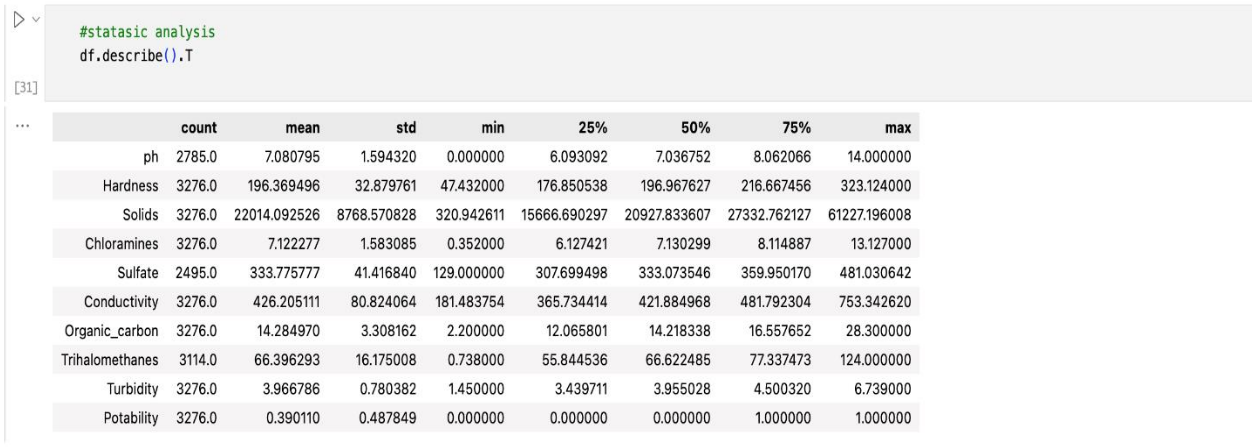
```
print(f"total_no_of_rows:{df.shape[0]} and colums:{df.shape[1]}")
#prints total no of rows and columns
```
[20]

```
total_no_of_rows:3276 and colums:10
```

<u>Data Implementation: -</u>

By using the “df.isna().sum()” command we can get the missing values. Missing values lead to incorrect efficiency of water analysis. So, running the df.describe() command will give you the number of rows of data, as well as the average, minimum, maximum and median values. The df.isfillna() command will help you take the mean instead of the missing values. With this implementation, the output accuracy is good enough. You can see below that there are no missing values after implementation.

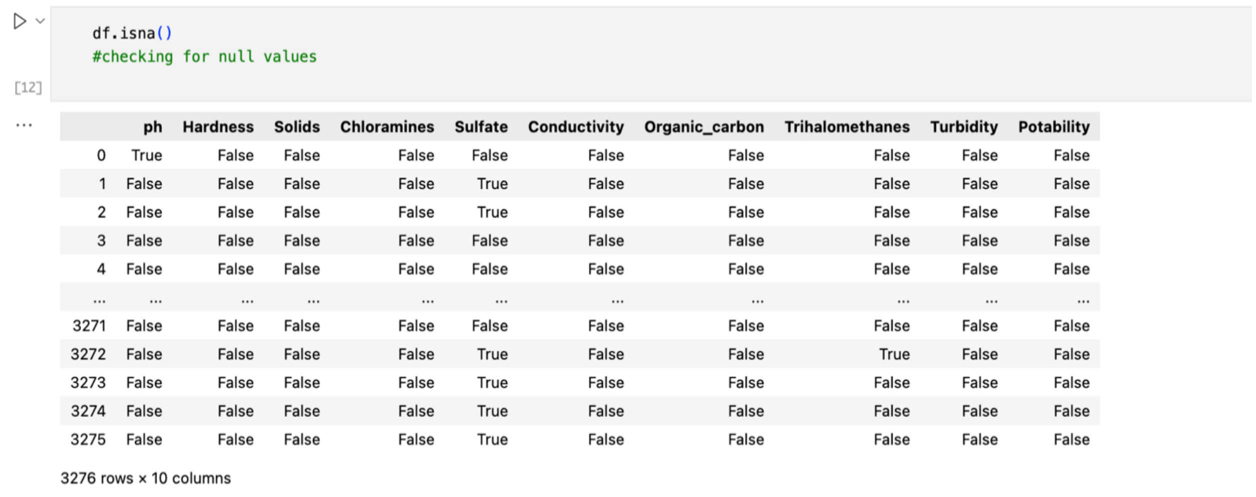
The df.describe() computes and displays summary statistics for a python data frame.



```
[31]
#statasic analysis
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
ph	2785.0	7.080795	1.594320	0.000000	6.093092	7.036752	8.062066	14.000000
Hardness	3276.0	196.369496	32.879761	47.432000	176.850538	196.967627	216.667456	323.124000
Solids	3276.0	22014.092526	8768.570828	320.942611	15666.690297	20927.833607	27332.762127	61227.196008
Chloramines	3276.0	7.122277	1.583085	0.352000	6.127421	7.130299	8.114887	13.127000
Sulfate	2495.0	333.775777	41.416840	129.000000	307.699498	333.073546	359.950170	481.030642
Conductivity	3276.0	426.205111	80.824064	181.483754	365.734414	421.884968	481.792304	753.342620
Organic_carbon	3276.0	14.284970	3.308162	2.200000	12.065801	14.218338	16.557652	28.300000
Trihalomethanes	3114.0	66.396293	16.175008	0.738000	55.844536	66.622485	77.337473	124.000000
Turbidity	3276.0	3.966786	0.780382	1.450000	3.439711	3.955028	4.500320	6.739000
Potability	3276.0	0.390110	0.487849	0.000000	0.000000	0.000000	1.000000	1.000000

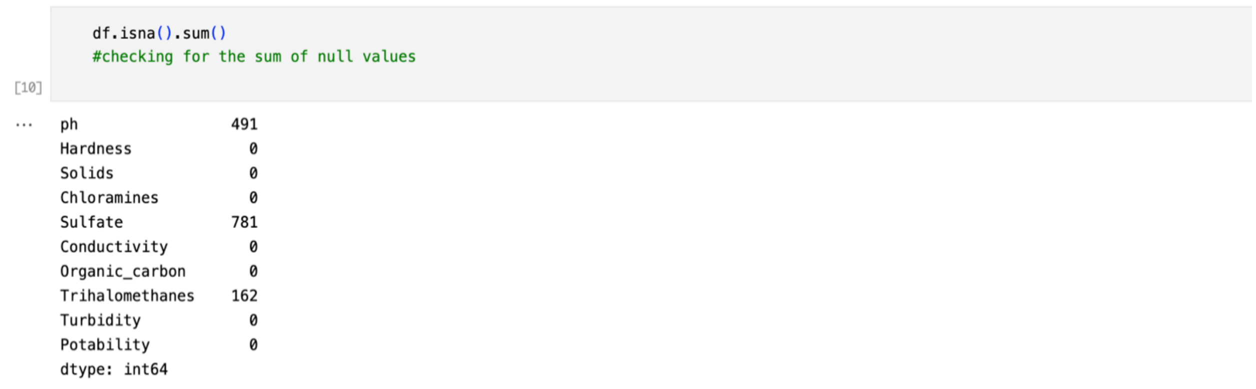
The df.isna() returns a data frame object where the values are of Boolean values True for NA values. The df.isna().sum() returns the number of NA values in the column.



```
[12]
df.isna()
#checking for null values
```

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	True	False	False	False	False	False	False	False	False	False
1	False	False	False	False	True	False	False	False	False	False
2	False	False	False	False	True	False	False	False	False	False
3	False	False	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...
3271	False	False	False	False	False	False	False	False	False	False
3272	False	False	False	False	True	False	False	True	False	False
3273	False	False	False	False	True	False	False	False	False	False
3274	False	False	False	False	True	False	False	False	False	False
3275	False	False	False	False	True	False	False	False	False	False

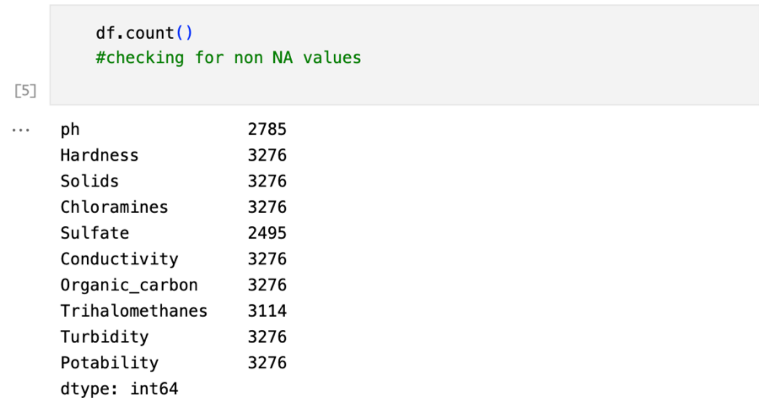
3276 rows × 10 columns



```
[10]
df.isna().sum()
#checking for the sum of null values
```

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

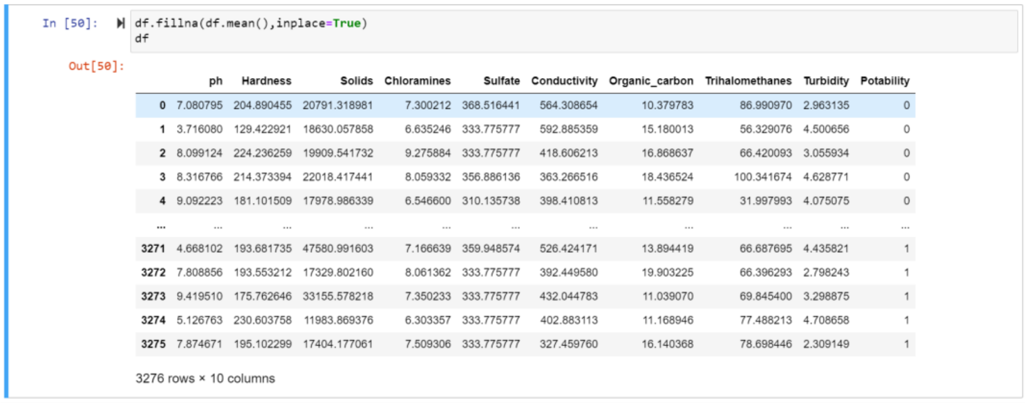
The df.count() returns the number of non NA values.



```
[5] df.count()
    #checking for non NA values
```

```
... ph                 2785
    Hardness           3276
    Solids             3276
    Chloramines        3276
    Sulfate            2495
    Conductivity       3276
    Organic_carbon     3276
    Trihalomethanes    3114
    Turbidity          3276
    Potability         3276
    dtype: int64
```

The df.fillna(df.mean(),inplace=True) will replace the missing values with mean to uniform the data.



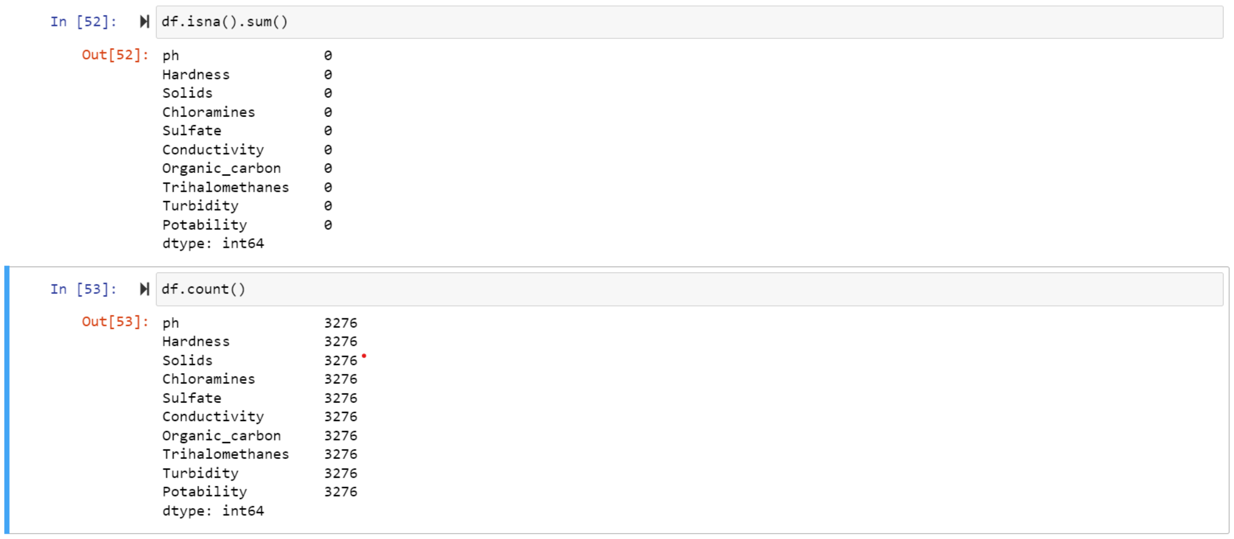
```
In [50]:  df.fillna(df.mean(),inplace=True)
          df
```

Out[50]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	7.080795	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0
1	3.716080	129.422921	18630.057858	6.635246	333.775777	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.541732	9.275884	333.775777	418.606213	16.868637	66.420093	3.055934	0
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	100.341674	4.628771	0
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0
...	...	...	...	...	...	...	...	...	...	...
3271	4.668102	193.681735	47580.991603	7.166639	359.948574	526.424171	13.894419	66.687695	4.435821	1
3272	7.808856	193.553212	17329.802160	8.061362	333.775777	392.449580	19.903225	66.396293	2.798243	1
3273	9.419510	175.762646	33155.578218	7.350233	333.775777	432.044783	11.039070	69.845400	3.298875	1
3274	5.126763	230.603758	11983.869376	6.303357	333.775777	402.883113	11.168946	77.488213	4.708658	1
3275	7.874671	195.102299	17404.177061	7.509306	333.775777	327.459760	16.140368	78.698446	2.309149	1

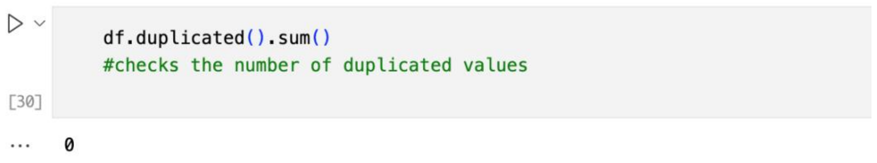
3276 rows × 10 columns

Now as the NA values are replaced with mean, we can run the df.isna().sum() to verify that.



```
In [52]:  df.isna().sum()

Out[52]: ph                 0
          Hardness           0
          Solids             0
          Chloramines        0
          Sulfate            0
          Conductivity       0
          Organic_carbon     0
          Trihalomethanes    0
          Turbidity          0
          Potability         0
          dtype: int64
```

```
In [53]:  df.count()

Out[53]: ph                 3276
          Hardness           3276
          Solids             3276
          Chloramines        3276
          Sulfate            3276
          Conductivity       3276
          Organic_carbon     3276
          Trihalomethanes    3276
          Turbidity          3276
          Potability         3276
          dtype: int64
```



```
df.duplicated().sum()
#checks the number of duplicated values
[30]
...  0
```

Split the data:-

In order to execute the analysis of the water quality using various algorithms, we need to get Training sets and Test sets. We may divide the data into training and test datasets using the Sklearn packages.

In [11]:
```
#get the training data and test data
y=data["Potability"]
y
```

Out[11]:
```
0       0
1       0
2       0
3       0
4       0
       ..
3271    1
3272    1
3273    1
3274    1
3275    1
Name: Potability, Length: 3276, dtype: int64
```

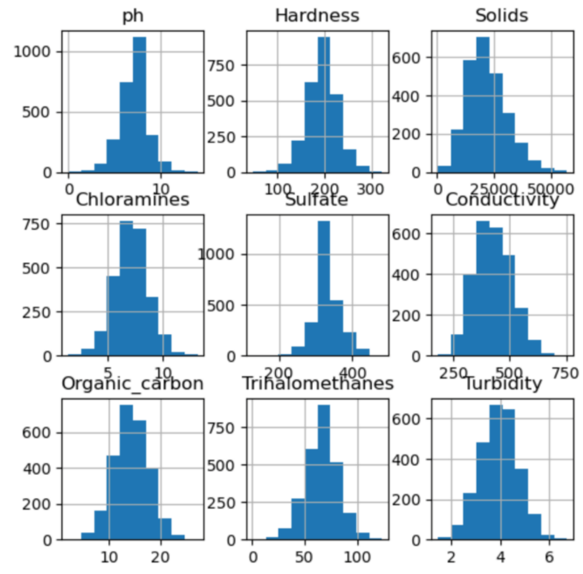
In [12]:
```
del data["Potability"]
x=data
x
```

Out[12]:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity
0	7.080795	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135
1	3.716080	129.422921	18630.057858	6.635246	333.775777	592.885359	15.180013	56.329076	4.500656
2	8.099124	224.236259	19909.541732	9.275884	333.775777	418.606213	16.868637	66.420093	3.055934
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	100.341674	4.628771
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075
...	...	...	...	...	...	...	...	...	...
3271	4.668102	193.681735	47580.991603	7.166639	359.948574	526.424171	13.894419	66.687695	4.435821
3272	7.808856	193.553212	17329.802160	8.061362	333.775777	392.449580	19.903225	66.396293	2.798243
3273	9.419510	175.762646	33155.578218	7.350233	333.775777	432.044783	11.039070	69.845400	3.298875
3274	5.126763	230.603758	11983.869376	6.303357	333.775777	402.883113	11.168946	77.488213	4.708658
3275	7.874671	195.102299	17404.177061	7.509306	333.775777	327.459760	16.140368	78.698446	2.309149

In [13]:
```
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.2,shuffle=True,random_state=None)

```

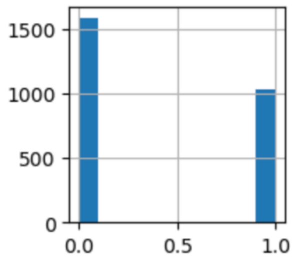
In [32]:
```
X_train.hist(figsize=(6,6))
plt.show()
X_train
```



	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity
1342	8.248703	201.338857	24912.879705	7.877666	389.088844	415.683448	12.842048	51.859151	3.540532
2107	3.906568	152.818846	17857.716894	7.363060	324.382843	342.408791	7.637198	75.201346	3.204796
2337	6.676137	192.685849	27468.980170	5.809436	281.406871	299.886997	12.237608	46.069213	4.336648
2797	7.210774	163.047283	14230.419131	7.352941	324.095726	441.524088	9.793010	89.919916	4.462408
507	7.121458	204.164139	20574.364258	7.089146	333.775777	353.927593	16.488156	57.022783	3.774601
...	...	...	...	...	...	...	...	...	...
774	7.080795	231.159824	12856.728623	6.977019	344.447737	392.516902	13.217223	80.956458	3.308294
1223	7.582942	185.045993	23072.780698	5.892273	297.505772	518.048524	12.038457	80.312558	4.164029
270	7.291888	127.544297	27784.048484	9.754476	247.335412	439.649329	17.285042	59.556330	5.328713
2665	7.114387	196.533960	27022.708505	6.176786	333.775777	497.523605	18.704279	59.730438	3.762682
598	8.214100	192.177127	12819.875436	7.536023	343.477262	418.678765	13.352604	86.978101	3.381864

```
Y_train.hist(figsize=(2,2))
plt.show()
Y_train
```

[41]

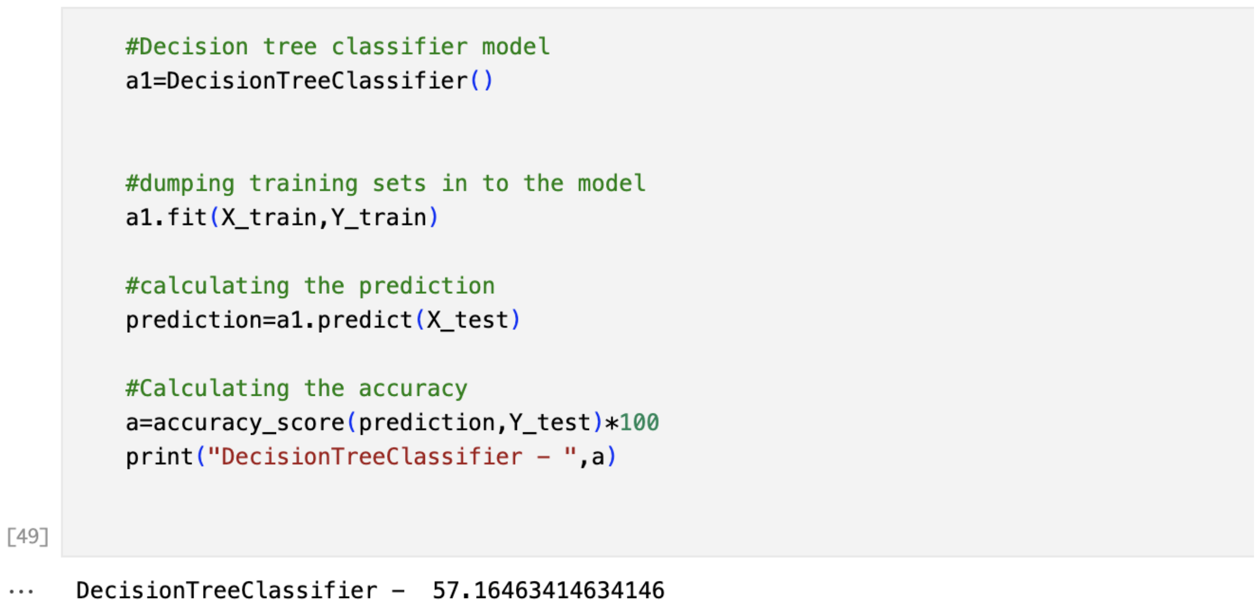


```
2629    0
2988    0
2059    0
185     0
2703    0
       ..
3128    1
706     1
1476    0
2894    0
740     1
Name: Potability, Length: 2620, dtype: int64
```

Evaluating the model performance:-

1. Decision Tree Classifier:-

Decision Tree is a supervised machine learning algorithm that mimics the way humans make decisions by using a set of rules. The Decision Tree Classifier method is used to evaluate the water quality. The efficiency that we will obtain from utilizing this decision tree classifier is measured by the accuracy score. With this model, we first deploy the training datasets that we acquired before obtaining the prediction using test datasets. We can calculate the accuracy score using that forecast.



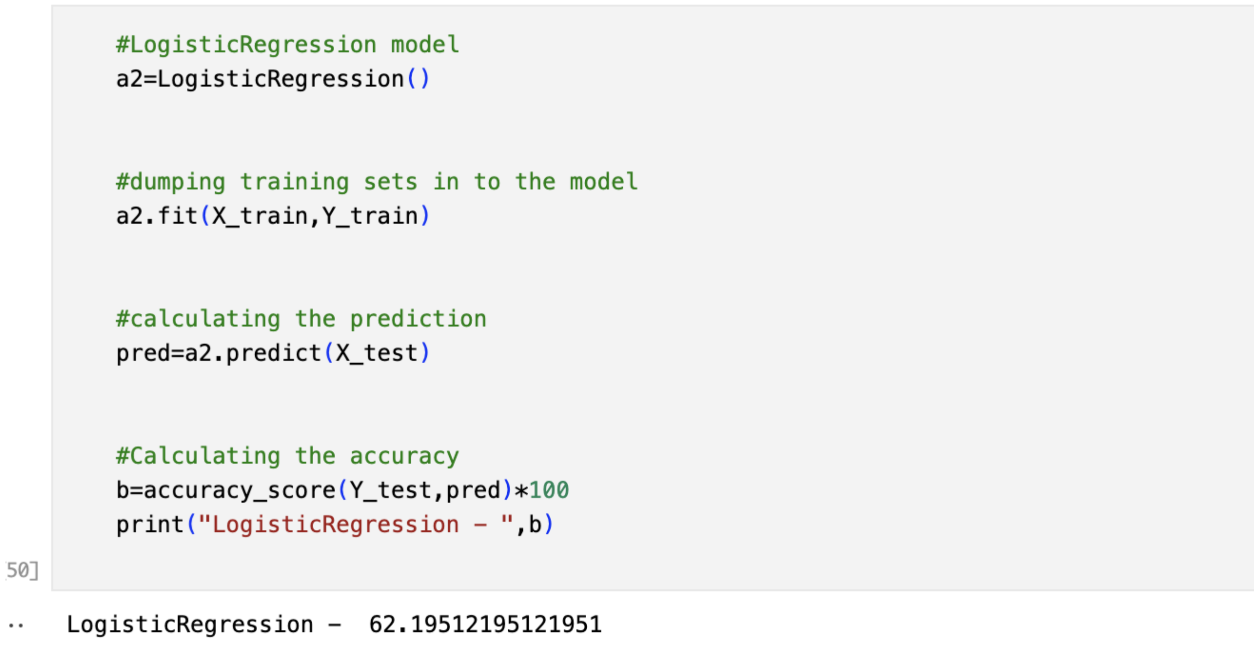
```
#Decision tree classifier model
a1=DecisionTreeClassifier()


#dumping training sets in to the model
a1.fit(X_train,Y_train)

#calculating the prediction
prediction=a1.predict(X_test)

#Calculating the accuracy
a=accuracy_score(prediction,Y_test)*100
print("DecisionTreeClassifier - ",a)
```

[49]

```
...  DecisionTreeClassifier -  57.16463414634146
```

2. Logistic Regression Model:-

Logistic regression is a process of modeling the probability of a discrete outcome given an input variable. To analyze the water quality, the algorithm logistic regression is applied. The efficiency that we will obtain from applying this Logistic Regression is measured by the accuracy score. With this model, we first deploy the training datasets that we acquired before obtaining the prediction using test datasets. We can calculate the accuracy score using that forecast.



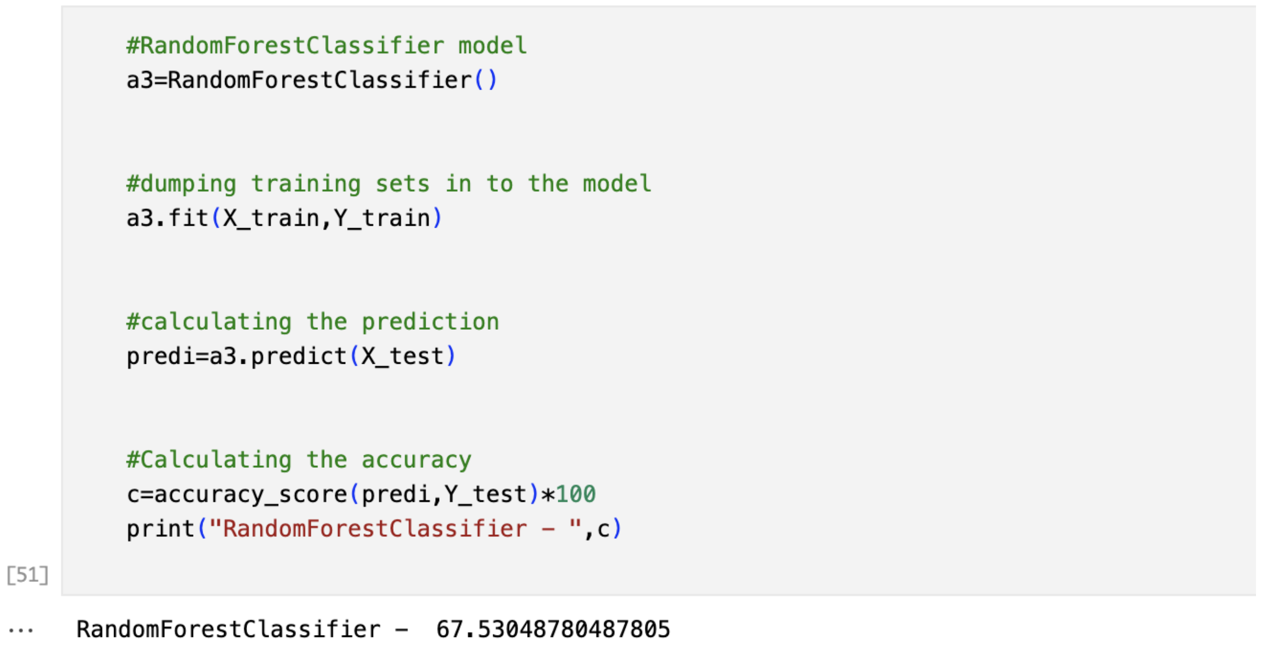
```
#LogisticRegression model
a2=LogisticRegression()


#dumping training sets in to the model
a2.fit(X_train,Y_train)


#calculating the prediction
pred=a2.predict(X_test)


#Calculating the accuracy
b=accuracy_score(Y_test,pred)*100
print("LogisticRegression - ",b)
```

50]

```
·· LogisticRegression -  62.19512195121951
```

3. Random forest Classifier:-

A random forest is an algorithm that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. A technique called Random Forest Classifier is used to evaluate the quality of water. The accuracy rating represents the effectiveness of this Random Forest classifier. In this model, we first deploy the obtained training sets before obtaining the prediction using the obtained test datasets. We can calculate the accuracy score using that forecast.



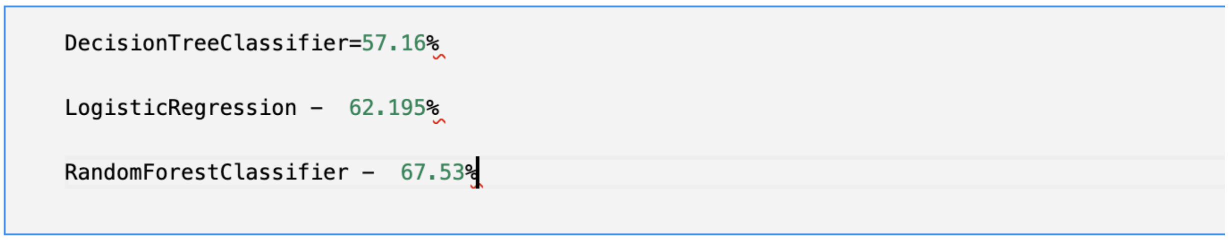
```
#RandomForestClassifier model
a3=RandomForestClassifier()


#dumping training sets in to the model
a3.fit(X_train,Y_train)


#calculating the prediction
predi=a3.predict(X_test)


#Calculating the accuracy
c=accuracy_score(predi,Y_test)*100
print("RandomForestClassifier - ",c)
```

[51]

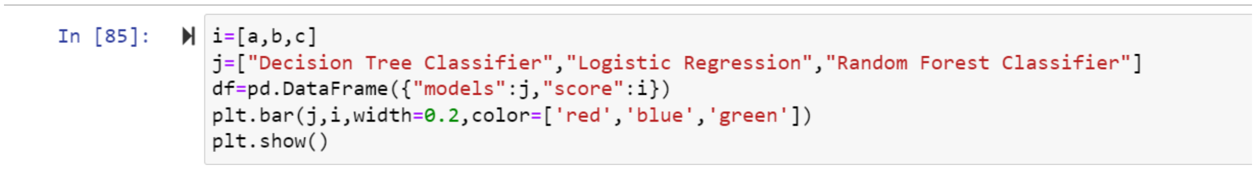
```
··· RandomForestClassifier -  67.53048780487805
```

Final Output:-

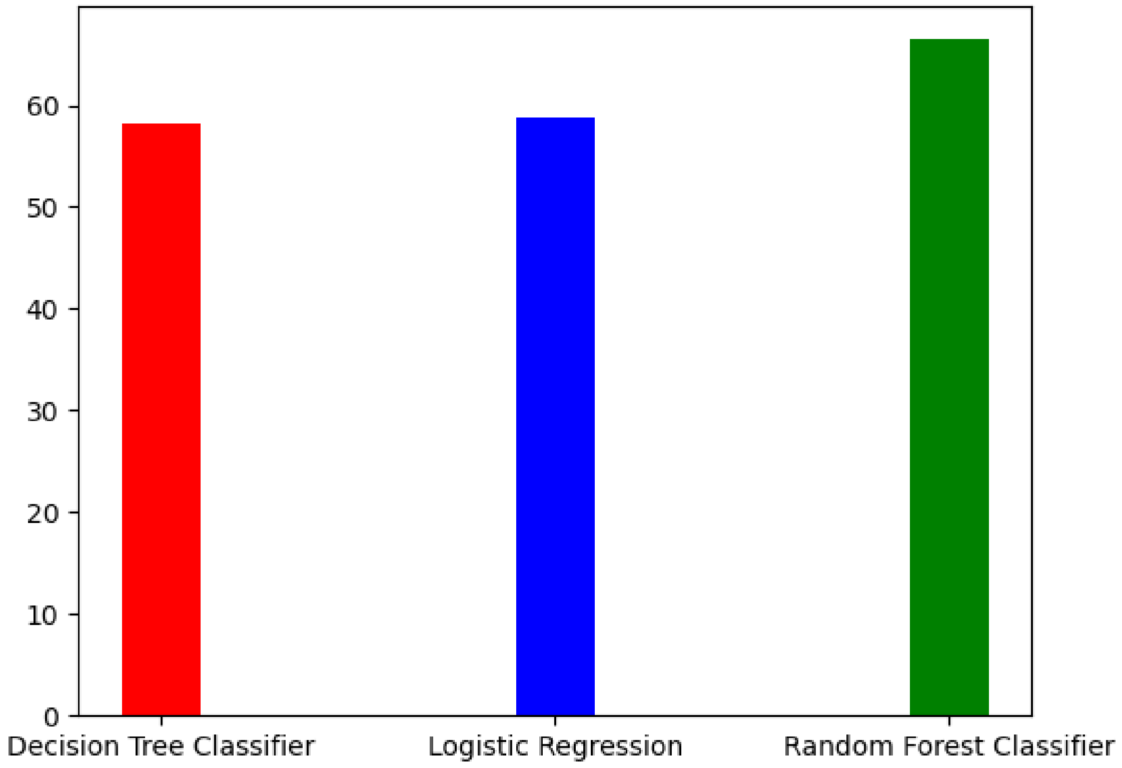
In this study, we conducted an analysis of water quality using three distinct algorithms, each of which was evaluated based on its accuracy score. After comparing the accuracy scores obtained from these algorithms, we found that the Random Forest classifier was the most effective algorithm for assessing water quality, achieving an accuracy score of 70.88%. The second-best algorithm was the Logistic Regression method, which achieved an accuracy score of 63.4%. Lastly, the Decision Tree classifier yielded the least accurate results, with an accuracy score of 58.8%.



```
DecisionTreeClassifier=57.16%

LogisticRegression -  62.195%

RandomForestClassifier -  67.53%
```



```
In [85]:  i=[a,b,c]
          j=["Decision Tree Classifier","Logistic Regression","Random Forest Classifier"]
          df=pd.DataFrame({"models":j,"score":i})
          plt.bar(j,i,width=0.2,color=['red','blue','green'])
          plt.show()
```


Decision Tree Classifier
Logistic Regression
Random Forest Classifier