# WATER-QUALITY ANALYSIS

## PHASE -4

→ MODEL BUILDING

→ MODEL EVALUATION

→ VISUALISATION USING COGNOS

NAME: E.PREETHI RAJALAKSHMI

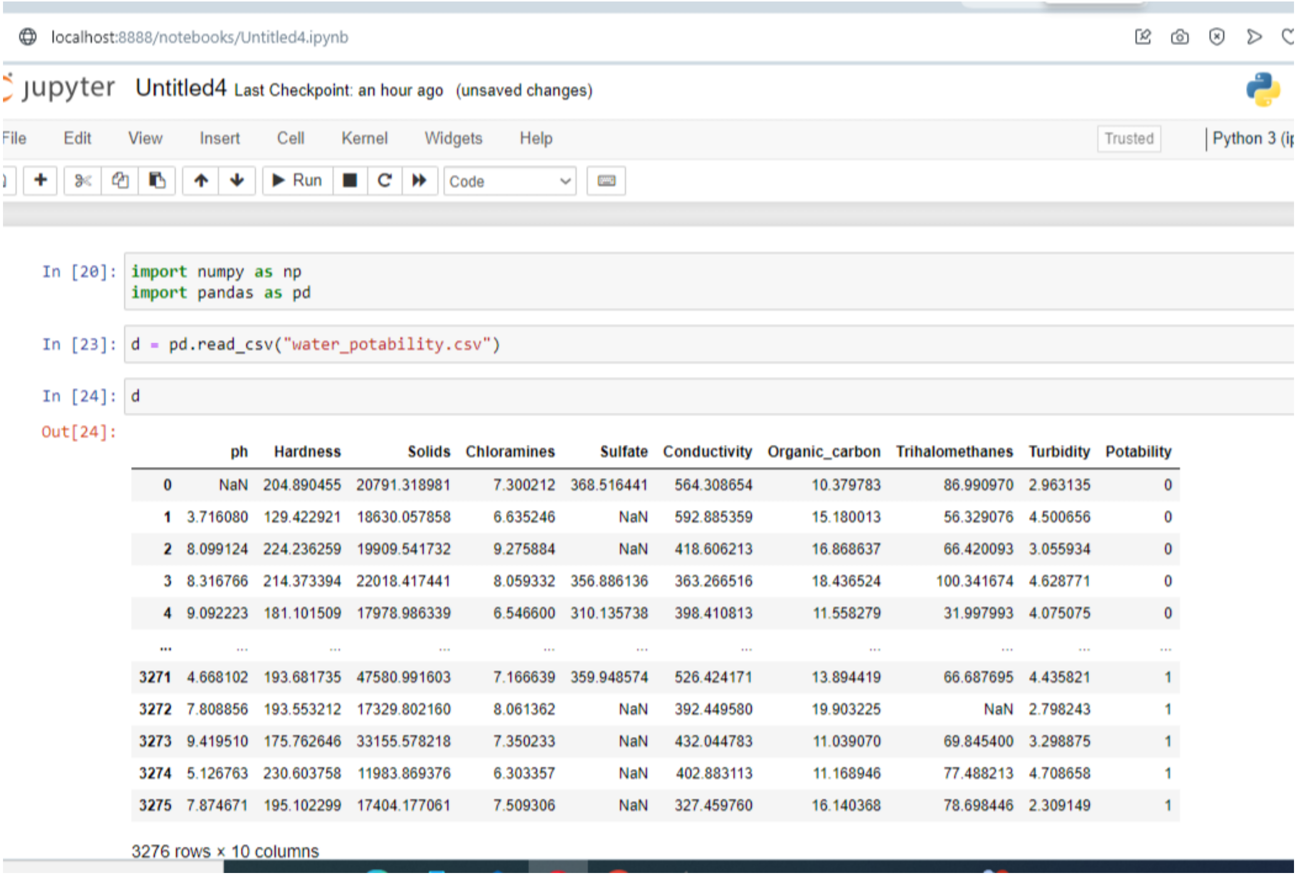
# 1. IMPORT THE LIBRARY

Import numpy as np

Import pandas as pd

Import matplotlib.pyplot as plt

# 2. IMPORT THE DATASET


localhost:8888/notebooks/Untitled4.ipynb
jupyter
Untitled4 Last Checkpoint: an hour ago (unsaved changes)
File
Edit
View
Insert
Cell
Kernel
Widgets
Help
Trusted
Python 3 (ip
Run
Code

```
In [20]: import numpy as np
         import pandas as pd
```

```
In [23]: d = pd.read_csv("water_potability.csv")
```

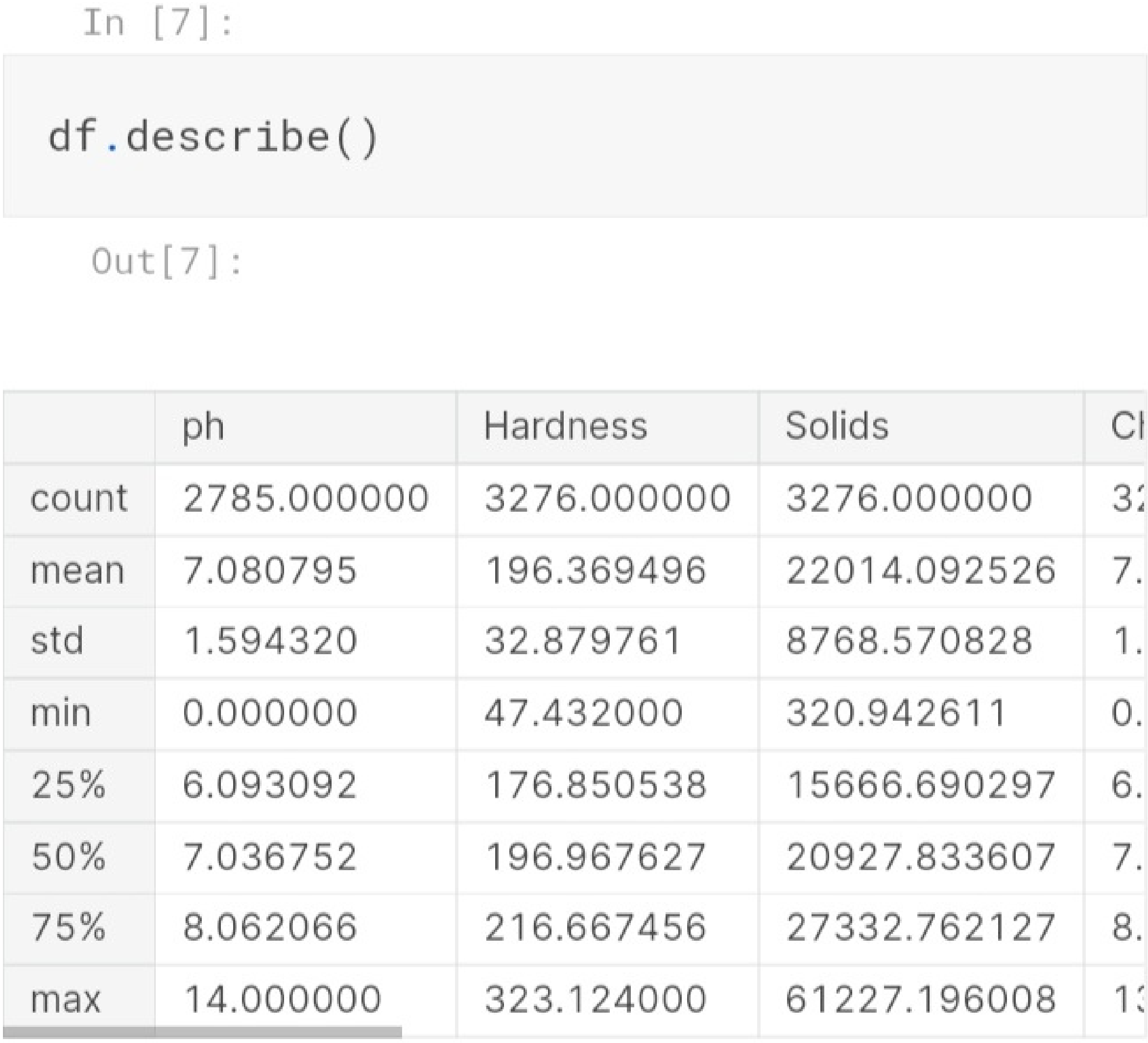
```
In [24]: d
```

Out[24]:

| | 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 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **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 |

3276 rows × 10 columns

# 3. DESCRIBE THE DATA



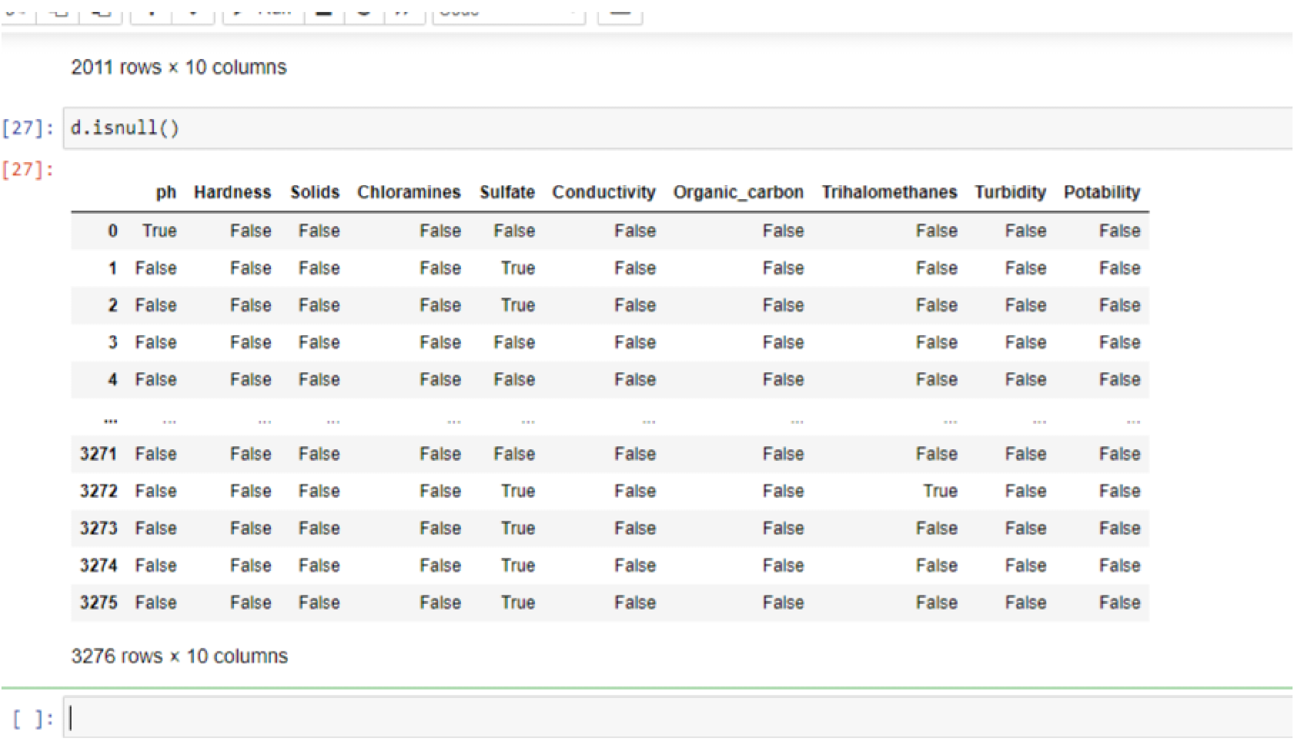
In [7]:

```
df.describe()
```

Out[7]:

| | ph | Hardness | Solids | Cl |
|---|---|---|---|---|
| count | 2785.000000 | 3276.000000 | 3276.000000 | 32 |
| mean | 7.080795 | 196.369496 | 22014.092526 | 7. |
| std | 1.594320 | 32.879761 | 8768.570828 | 1. |
| min | 0.000000 | 47.432000 | 320.942611 | 0. |
| 25% | 6.093092 | 176.850538 | 15666.690297 | 6. |
| 50% | 7.036752 | 196.967627 | 20927.833607 | 7. |
| 75% | 8.062066 | 216.667456 | 27332.762127 | 8. |
| max | 14.000000 | 323.124000 | 61227.196008 | 1 |

# 4. HANDLING MISSING VALUES



2011 rows × 10 columns

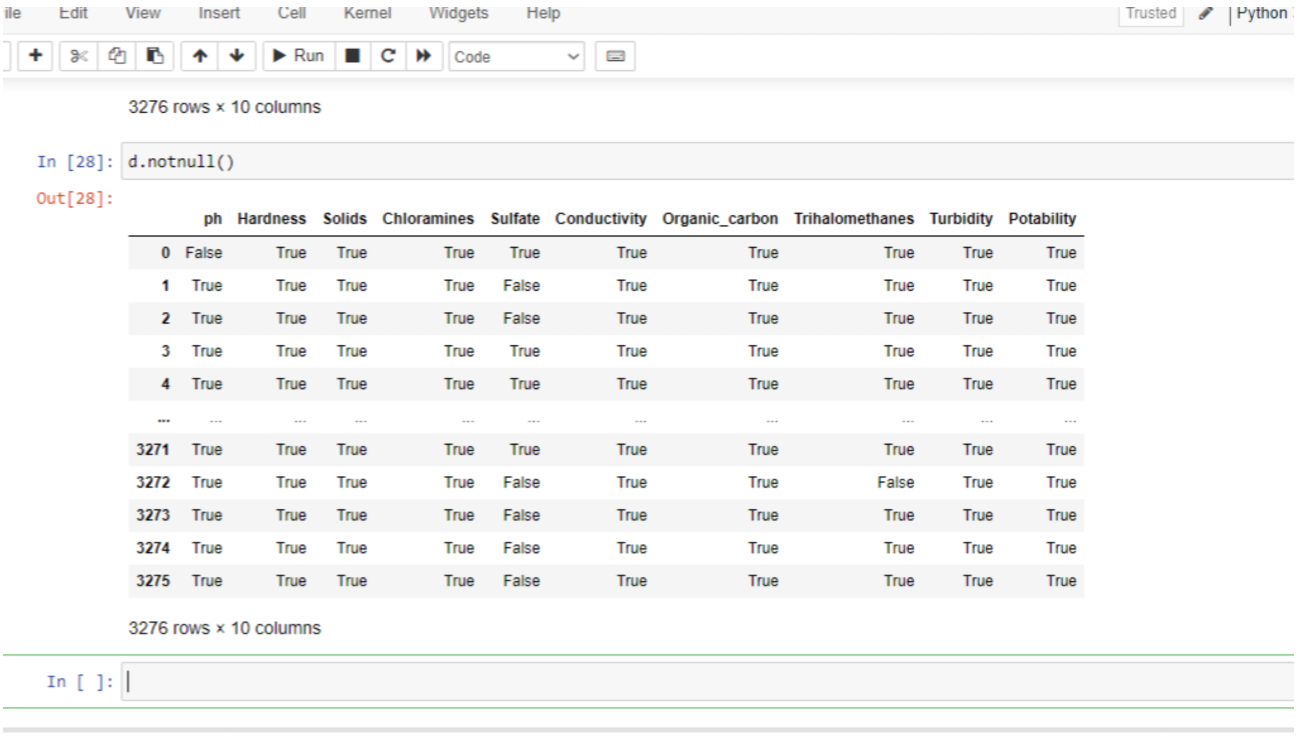
```
[27]: d.isnull()
```

[27]:

| | 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

```
[ ]: 
```



3276 rows × 10 columns

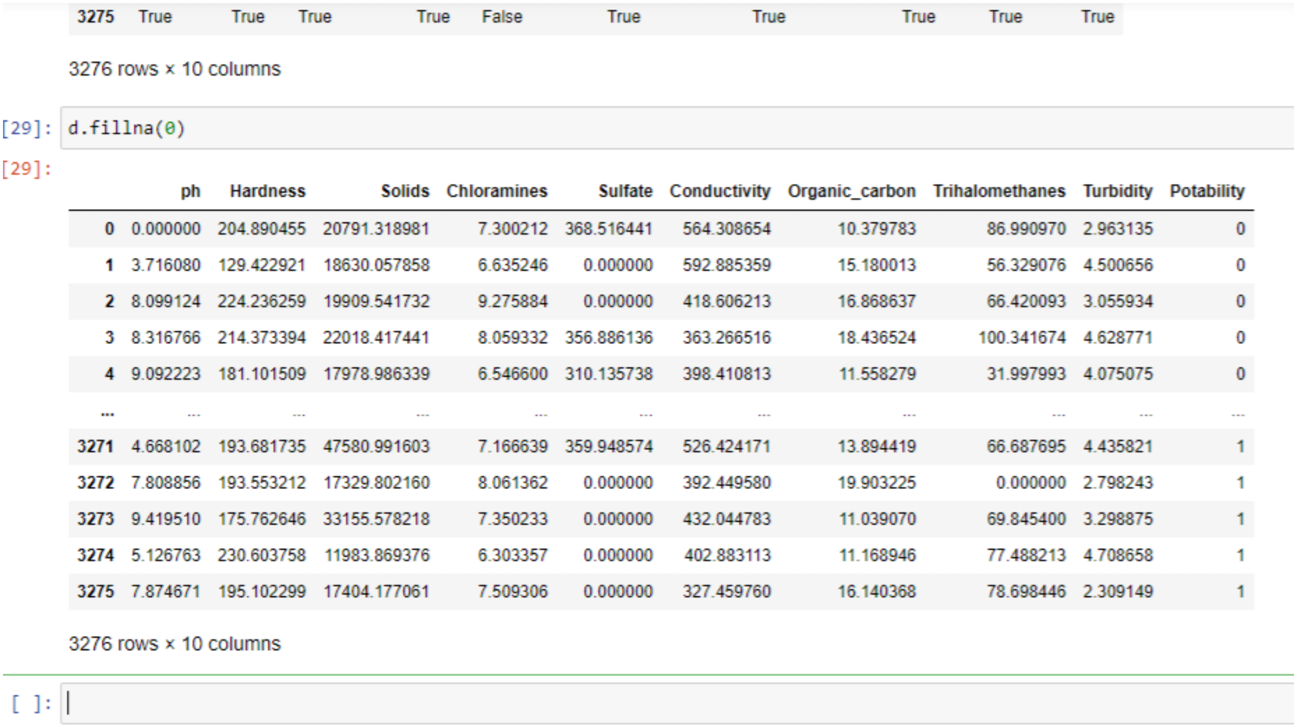
```
In [28]: d.notnull()
```

Out[28]:

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

3276 rows × 10 columns

```
In [ ]: 
```



| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **3275** | True | True | True | True | False | True | True | True | True | True |

3276 rows × 10 columns

```
[29]: d.fillna(0)
```

[29]:

| | ph | Hardness | Solids | Chloramines | Sulfate | Conductivity | Organic_carbon | Trihalomethanes | Turbidity | Potability |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.000000 | 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 | 0.000000 | 592.885359 | 15.180013 | 56.329076 | 4.500656 | 0 |
| **2** | 8.099124 | 224.236259 | 19909.541732 | 9.275884 | 0.000000 | 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 | 0.000000 | 392.449580 | 19.903225 | 0.000000 | 2.798243 | 1 |
| **3273** | 9.419510 | 175.762646 | 33155.578218 | 7.350233 | 0.000000 | 432.044783 | 11.039070 | 69.845400 | 3.298875 | 1 |
| **3274** | 5.126763 | 230.603758 | 11983.869376 | 6.303357 | 0.000000 | 402.883113 | 11.168946 | 77.488213 | 4.708658 | 1 |
| **3275** | 7.874671 | 195.102299 | 17404.177061 | 7.509306 | 0.000000 | 327.459760 | 16.140368 | 78.698446 | 2.309149 | 1 |

3276 rows × 10 columns

```
[ ]:
```

# 5. MODEL BUILDING

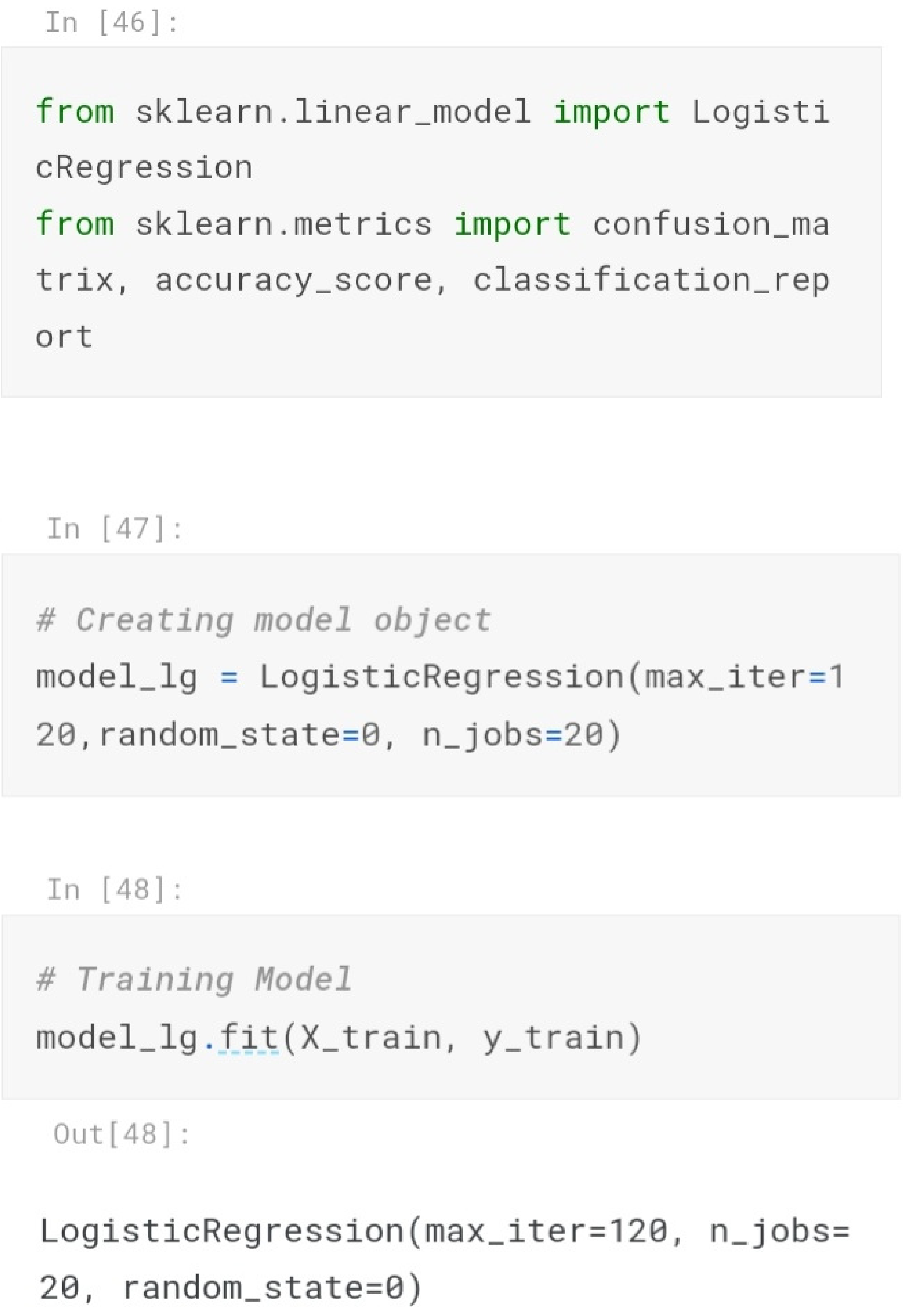
In [44]:

```
# import train-test split
from sklearn.model_selection import trai
n_test_split
```

In [45]:

```
X_train, X_test, y_train, y_test = train
_test_split(X, y, test_size=0.33, random
_state=42)
```

Using  logistic regression model



In [46]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
```

In [47]:

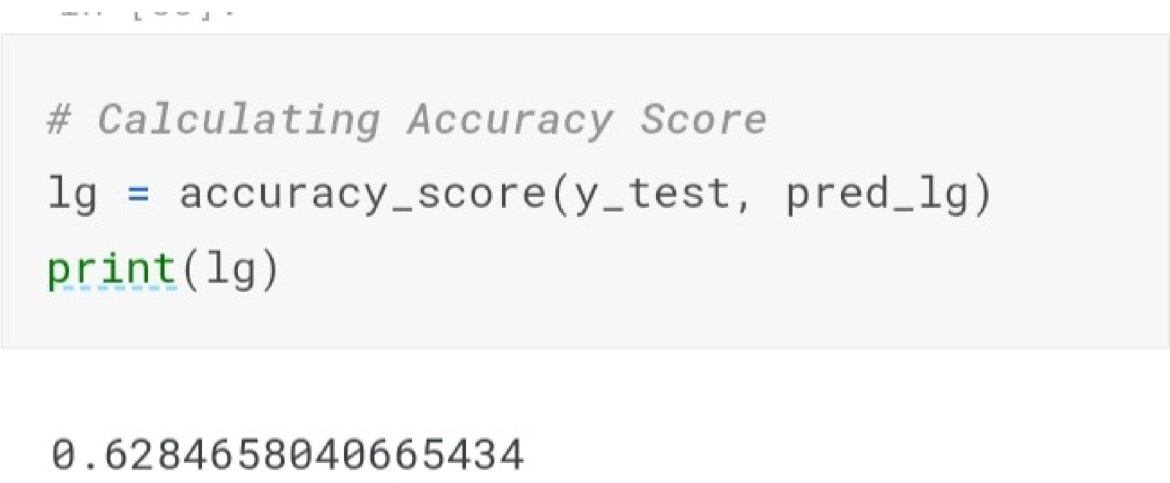
```
# Creating model object
model_lg = LogisticRegression(max_iter=120,random_state=0, n_jobs=20)
```

In [48]:

```
# Training Model
model_lg.fit(X_train, y_train)
```

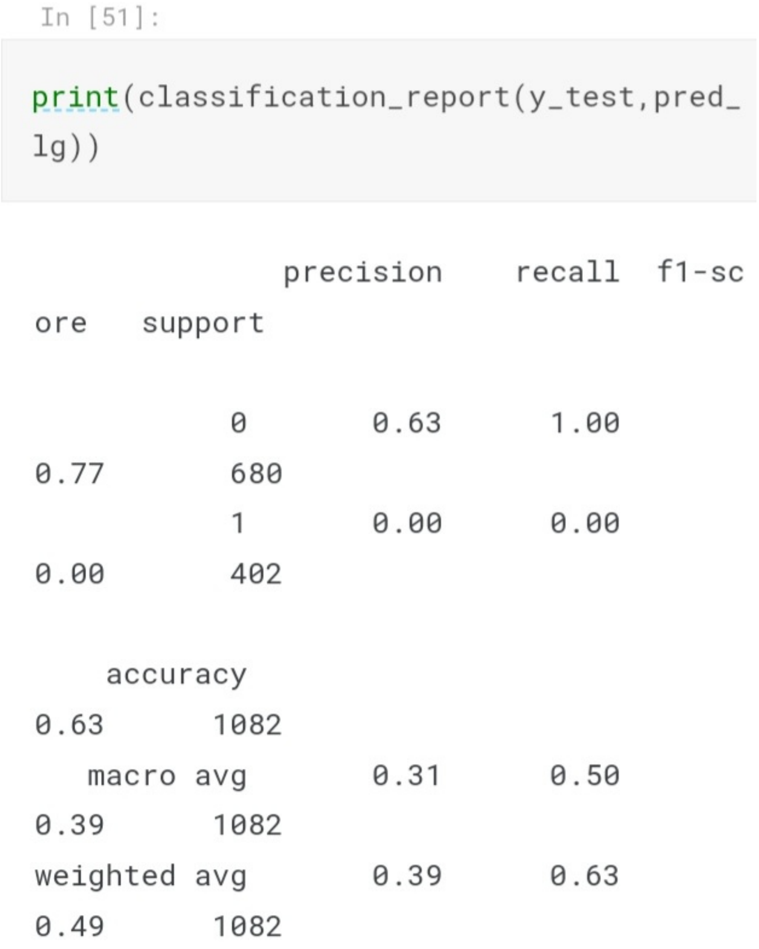
Out[48]:

```
LogisticRegression(max_iter=120, n_jobs=20, random_state=0)
```



```
# Calculating Accuracy Score
lg = accuracy_score(y_test, pred_lg)
print(lg)
```

```
0.6284658040665434
```



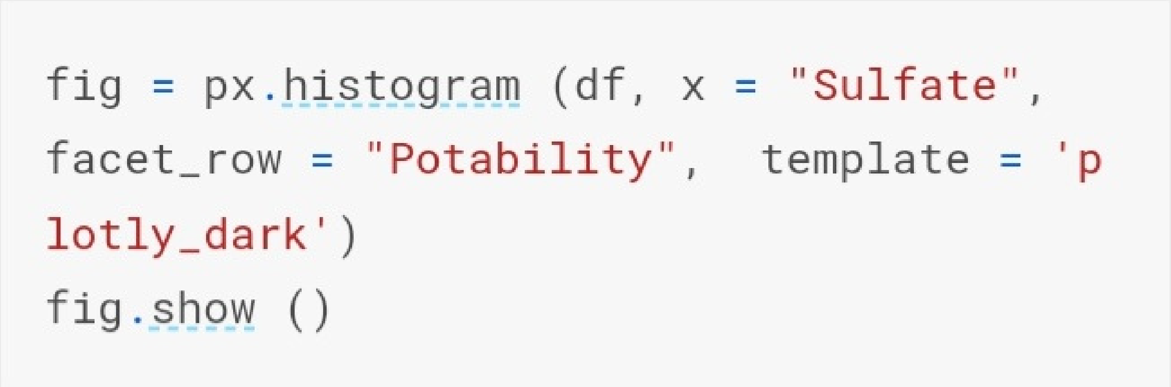
In [51]:

```
print(classification_report(y_test,pred_
lg))
```

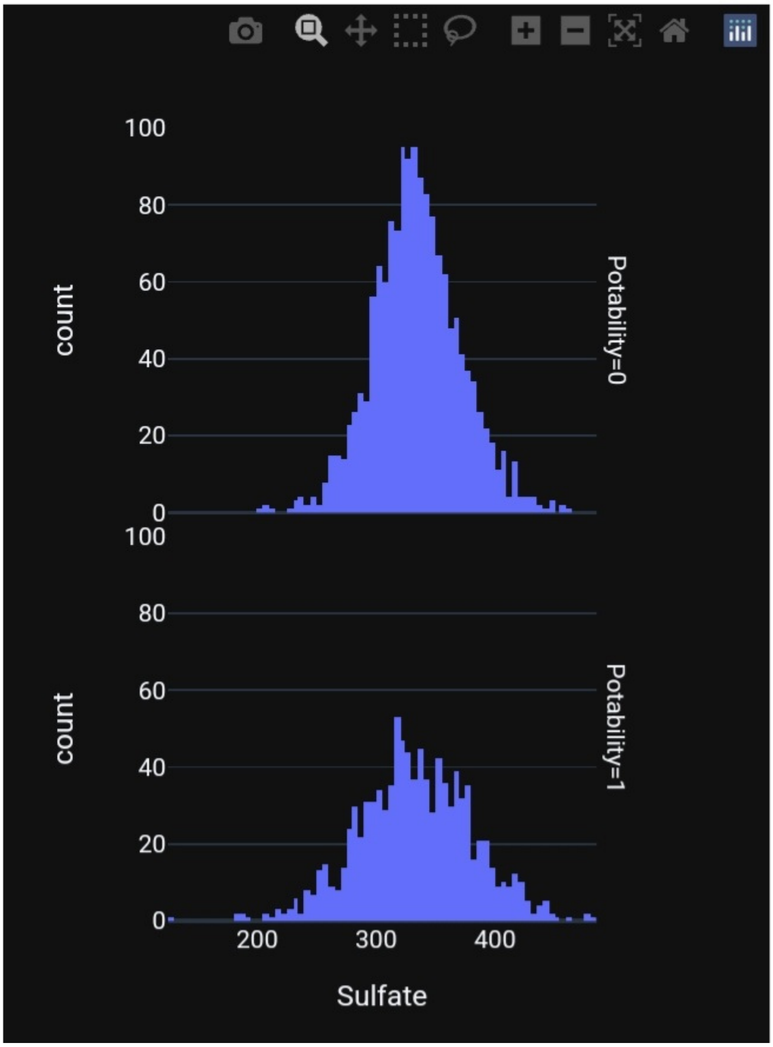
```
              precision    recall  f1-sc
ore   support

           0       0.63      1.00      
0.77       680
           1       0.00      0.00      
0.00       402

    accuracy                           
0.63      1082
   macro avg       0.31      0.50      
0.39      1082
weighted avg       0.39      0.63      
0.49      1082
```

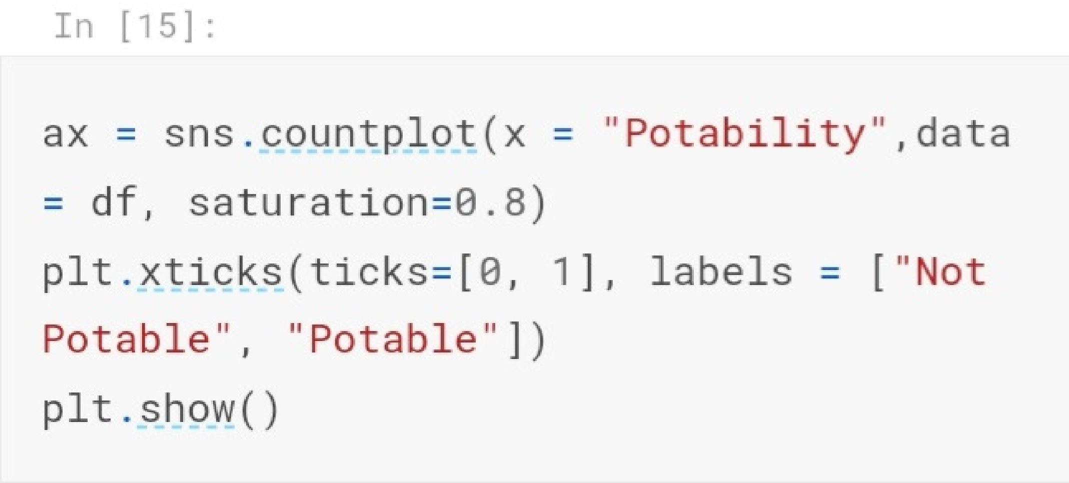
# 6. MODEL VISUALIZATION

Potability: Indicates if water is safe for human consumption where 1 means Potable and 0 means Not potable.



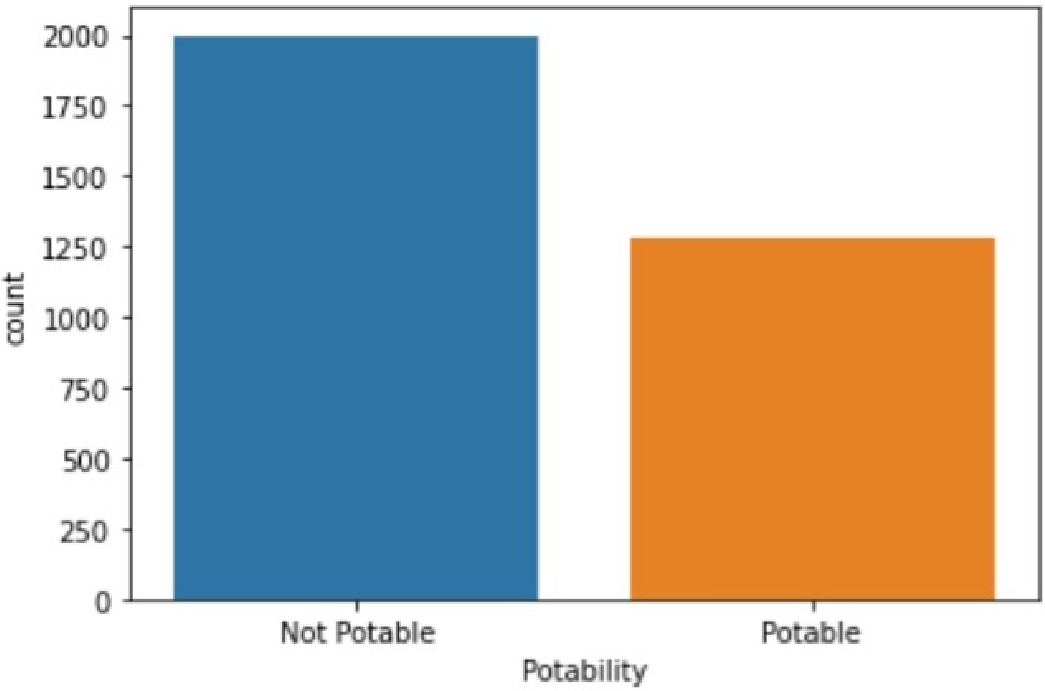
```
fig = px.histogram (df, x = "Sulfate",
facet_row = "Potability",  template = 'plotly_dark')
fig.show ()
```





In [15]:

```
ax = sns.countplot(x = "Potability",data = df, saturation=0.8)
plt.xticks(ticks=[0, 1], labels = ["Not Potable", "Potable"])
plt.show()
```


count
Not Potable
Potable
Potability