PROJECT DEVELOPMENT PHASE - SPRINT II

Assignment Date	19-11-2022
Team ID	PNT2022TMID30408
Project Name	Efficient Water Quality Analysis and Prediction using Machine Learning
Maximum Marks	8 Mark

DATA PRE-PROCESSING

Click here to view the project:

Importing Required Package:

```
import pandas as pd import seaborn as
sns import numpy as np from
matplotlib import pyplot as plt
%matplotlib inline
```

Loading the Dataset Solution :

```
df = pd.read_csv("water_potability.csv")
df
```

Output:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	NaN	204.890456	20791.31898	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0
1	3.716080	129.422921	18630.05786	6.635246	NaN	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.54173	9.275884	NaN	418.606213	16.868637	66.420093	3.055934	0
3	8.316766	214.373394	22018.41744	8.059332	356.886136	363.266516	18.436525	100.341674	4.628771	0
4	9.092223	181.101509	17978.98634	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0
...	...	...	...	...	...	...	...	...	...	...
3271	4.668102	193.681736	47580.99160	7.166639	359.948574	526.424171	13.894419	66.687695	4.435821	1
3272	7.808856	193.553212	17329.80216	8.061362	NaN	392.449580	19.903225	NaN	2.798243	1
3273	9.419510	175.762646	33155.57822	7.350233	NaN	432.044783	11.039070	69.845400	3.298875	1
3274	5.126763	230.603758	11983.86938	6.303357	NaN	402.883113	11.168946	77.488213	4.708658	1
3275	7.874671	195.102299	17404.17706	7.509306	NaN	327.459761	16.140368	78.698446	2.309149	1

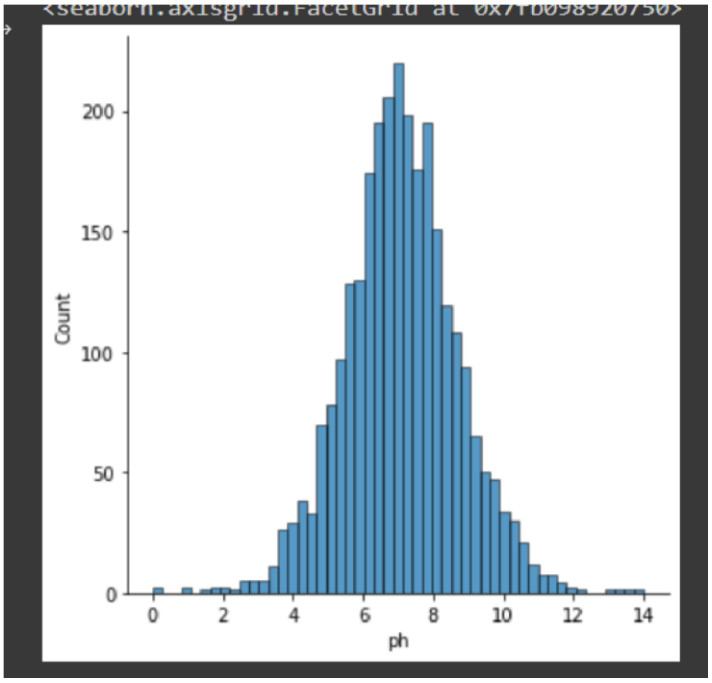
3276 rows × 10 columns

Visualizations

Univariate Analysis Solution:

```
sns.displot(df.ph)
```

Output:

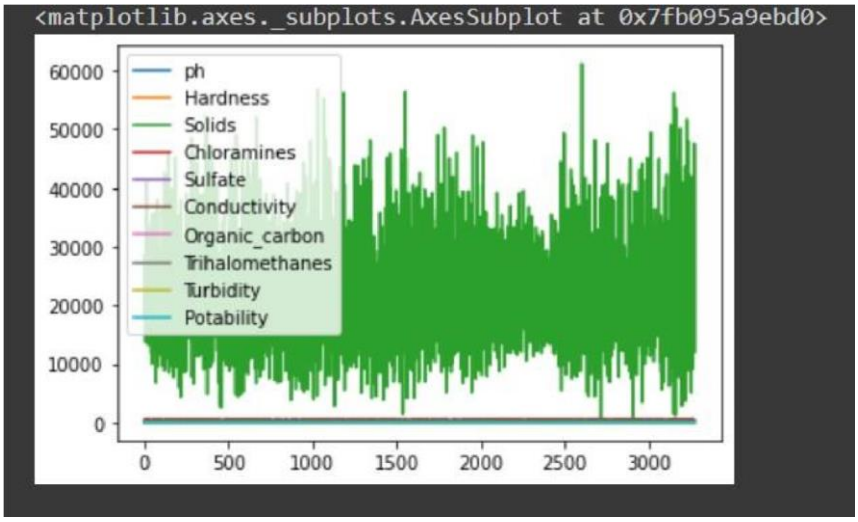


Bi-Variate Analysis

Solution:

```
df.plot.line()
```

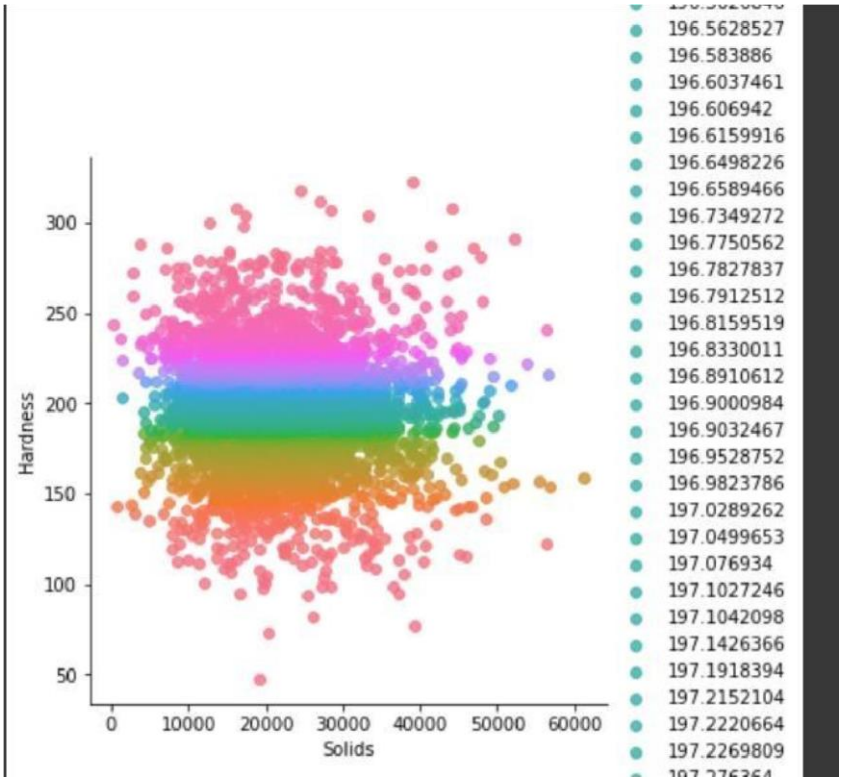
Output:



Multi - Variate Analysis Solution:

```
sns.lmplot("Solids","Hardness",df,hue="Hardness", fit_reg=False);
```

Output:



. Perform descriptive statistics on the dataset.

Solution:

```
df.describe()
```

Output:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
count	2785.000000	3276.000000	3276.000000	3276.000000	2495.000000	3276.000000	3276.000000	3114.000000	3276.000000	3276.000000
mean	7.080795	196.369496	22014.092526	7.122277	333.775777	426.205111	14.284970	66.396293	3.966786	0.390110
std	1.594320	32.879761	8768.570828	1.583085	41.416840	80.824064	3.308162	16.175008	0.780382	0.487849
min	0.000000	47.432000	320.942611	0.352000	129.000000	181.483754	2.200000	0.738000	1.450000	0.000000
25%	6.093092	176.850538	15666.690300	6.127421	307.699498	365.734414	12.065801	55.844536	3.439711	0.000000
50%	7.036752	196.967627	20927.833605	7.130299	333.073546	421.884968	14.218338	66.622485	3.955028	0.000000
75%	8.062066	216.667456	27332.762125	8.114887	359.950170	481.792305	16.557652	77.337473	4.500320	1.000000
max	14.000000	323.124000	61227.196010	13.127000	481.030642	753.342620	28.300000	124.000000	6.739000	1.000000

Handle the Missing values.

Solution:

```
data = pd.read_csv("water_potability.csv") pd.isnull(data["ph"])
```

Output:

```
0        True
1       False
2       False
3       False
4       False
        ...
3271    False
3272    False
3273    False
3274    False
3275    False
Name: ph, Length: 3276, dtype: bool
```

Handling Missing Values -2

Solution:

```
data = pd.read_csv("water_potability.csv") pd.isnull(data["conductivity"])
```

Output:

```
0       False
1       False
2       False
3       False
4       False
        ...
3271    False
3272    False
3273    False
3274    False
3275    False
Name: Conductivity, Length: 3276, dtype: bool
```

Split the data into dependent and independent variables Split the data into Independent variables.

Solution:

```
X    = df.iloc[:, :-2].values
print(X)
```

Output:

```
[[           nan 2.04890456e+02 2.07913190e+04 ... 5.64308654e+02
  1.03797831e+01 8.69909705e+01]
 [3.71608007e+00 1.29422921e+02 1.86300579e+04 ... 5.92885359e+02
  1.51800131e+01 5.63290763e+01]
 [8.09912419e+00 2.24236259e+02 1.99095417e+04 ... 4.18606213e+02
  1.68686369e+01 6.64200925e+01]
 ...
 [9.41951032e+00 1.75762646e+02 3.31555782e+04 ... 4.32044783e+02
  1.10390697e+01 6.98454003e+01]
 [5.12676292e+00 2.30603758e+02 1.19838694e+04 ... 4.02883113e+02
  1.11689462e+01 7.74882131e+01]
 [7.87467136e+00 1.95102299e+02 1.74041771e+04 ... 3.27459761e+02
  1.61403676e+01 7.86984463e+01]]
```

Split the data into Dependent variables.

Solution:

```
Y    = df.iloc[:, -1].values
print(Y)
```

Output:

```
[0 0 0 ... 1 1 1]
```

Scale the independent variables Solution:

```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler scaler
= MinMaxScaler()
df[["Hardness"]] = scaler.fit_transform(df[["Hardness"]])
print(df)
```

Output:

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	\
0	NaN	0.571139	0	7.300212	368.516441	564.308654	
1	3.716080	0.297400	0	6.635246	NaN	592.885359	
2	8.099124	0.641311	0	9.275884	NaN	418.606213	
3	8.316766	0.605536	0	8.059332	356.886136	363.266516	
4	9.092223	0.484851	0	6.546600	310.135738	398.410813	
...	...	...	...	...	...	...	
3271	4.668102	0.530482	0	7.166639	359.948574	526.424171	
3272	7.808856	0.530016	0	8.061362	NaN	392.449580	
3273	9.419510	0.465486	0	7.350233	NaN	432.044783	
3274	5.126763	0.664407	0	6.303357	NaN	402.883113	
3275	7.874671	0.535635	0	7.509306	NaN	327.459761	

	Organic_carbon	Trihalomethanes	Turbidity	Potability	nph	nHardness	\
0	10.379783	86.990970	2.963135	0	0	0	
1	15.180013	56.329076	4.500656	0	0	0	
2	16.868637	66.420093	3.055934	0	100	0	
3	18.436525	100.341674	4.628771	0	100	0	
4	11.558279	31.997993	4.075075	0	0	0	
...	...	...	...	...	...	...	
3271	13.894419	66.687695	4.435821	1	0	0	
3272	19.903225	NaN	2.798243	1	100	0	
3273	11.039070	69.845400	3.298875	1	0	0	
3274	11.168946	77.488213	4.708658	1	0	0	
3275	16.140368	78.698446	2.309149	1	100	0	

	wph	wHardness	wSolids	wqi
0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0
2	16.5	0.0	0.0	16.5
3	16.5	0.0	0.0	16.5
4	0.0	0.0	0.0	0.0
...	...	...	...	...
3271	0.0	0.0	0.0	0.0
3272	16.5	0.0	0.0	16.5
3273	0.0	0.0	0.0	0.0
3274	0.0	0.0	0.0	0.0
3275	16.5	0.0	0.0	16.5

[3276 rows x 16 columns]

Split the data into training and testing Solution:

```
from sklearn.model_selection import train_test_split train_size=0.8
X = df.drop(columns = ['ph']).copy() y
= df['ph']
X_train, X_rem, y_train, y_rem = train_test_split(X,y, train_size=0.8)
test_size = 0.5
X_valid, X_test, y_valid, y_test = train_test_split(X_rem,y_rem, test_size=0.
5) print(X_train.shape),
print(y_train.shape) print(X_valid.shape),
```

```
print(y_valid.shape) print(X_test.shape),
print(y_test.shape)
```

Output:

```
(2620, 9)
(2620,)
(328, 9)
(328,)
(328, 9)
(328,)
(None, None)
```

Water Quality Index Calculation :

Solution:

```
df['nph']=df.ph.apply(lambda x: (100 if (8.5>=x>=7)
else(80 if (8.6>=x>=8.5) or (6.9>=x>=6.8) else(60 if
(8.8>=x>=8.6) or (6.8>=x>=6.7) else(40 if
(9>=x>=8.8) or (6.7>=x>=6.5) else 0)))))
```

For second column:

```
df['nHardness']=df.Hardness.apply(lambda x: (100 if (x>=6)

else(80 if (6>=x>=5.1) else(60 if (5>=x>=4.1) else(40 if

(4>=x>=3) else 0)))))
```

For Third Column:

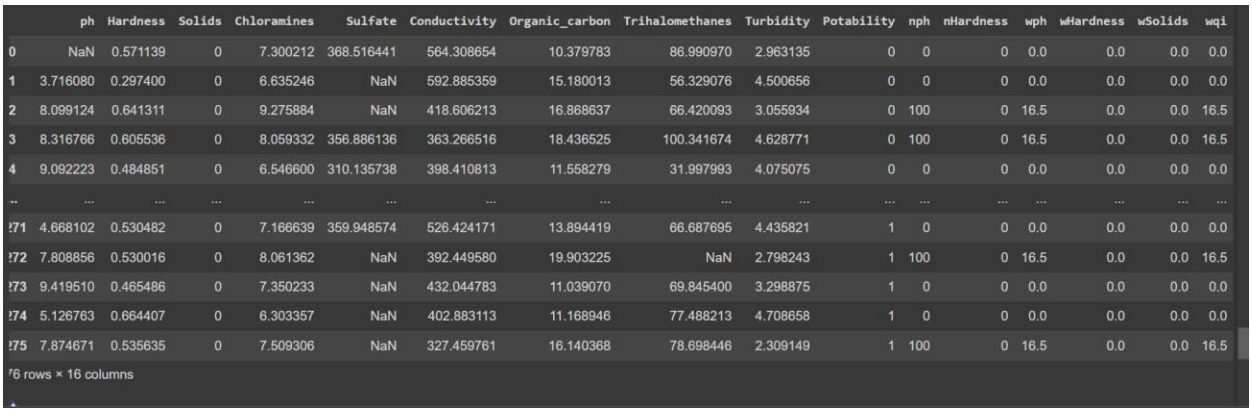
```
df['Solids']=df.Solids.apply(lambda x:(100 if (5>=x>=0)
else(80 if (50>=x>=5) else(60
if (500>=x>=50) else(40 if
(10000>=x>=500) else 0)))))
```

Calculation water Quality Index:

```
#calculation of water quality index WQI
df['wph']=df.nph*0.165
```

```
df['wHardness']=df.nHardness*0.281
df['wSolids']=df.Solids*0.281
df['wqi']=df.wph+df.wHardness+df.wSolids
df
```

Output:



	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability	nph	nHardness	wph	wHardness	wSolids	wqi
0	NaN	0.571139	0	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0	0	0	0.0	0.0	0.0	0.0
1	3.716080	0.297400	0	6.635246	NaN	592.885359	15.180013	56.329076	4.500656	0	0	0	0.0	0.0	0.0	0.0
2	8.099124	0.641311	0	9.275884	NaN	418.606213	16.868637	66.420093	3.055934	0	100	0	16.5	0.0	0.0	16.5
3	8.316766	0.605536	0	8.059332	356.886136	363.266516	18.436525	100.341674	4.628771	0	100	0	16.5	0.0	0.0	16.5
4	9.092223	0.484851	0	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0	0	0	0.0	0.0	0.0	0.0
..	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
271	4.668102	0.530482	0	7.166639	359.948574	526.424171	13.894419	66.687695	4.435821	1	0	0	0.0	0.0	0.0	0.0
272	7.808856	0.530016	0	8.061362	NaN	392.449580	19.903225	NaN	2.798243	1	100	0	16.5	0.0	0.0	16.5
273	9.419510	0.465486	0	7.350233	NaN	432.044783	11.039070	69.845400	3.298875	1	0	0	0.0	0.0	0.0	0.0
274	5.126763	0.664407	0	6.303357	NaN	402.883113	11.168946	77.488213	4.708658	1	0	0	0.0	0.0	0.0	0.0
275	7.874671	0.535635	0	7.509306	NaN	327.459761	16.140368	78.698446	2.309149	1	100	0	16.5	0.0	0.0	16.5

276 rows × 16 columns

Calculate the Average of WQI:

Solution:

```
average=df.groupby('Potability')['wqi'].mean()
```

Output:

```
Potability
0    6.372472
1    7.315462
Name: wqi, dtype: float64
```