College code: 4212
Register num: 421221243005

WATER QUALITY ANALYSIS

DATA ANALYTICS WITH COGNOS:GROUP2

PHASE:4

In the previous phases we have discussed about the step-by-step process, Design thinking and at the phase3 we have discussed about the data preprocessing techniques and many more in the last steps and in this step we have given some problem statements to solve in the WATER QUALITY ANALYSIS

In this part we will continue building our project, Building the analysis by creating visualizations

Problem: Continue building the analysis by creating visualizations and developing a predictive model.

IBM NAAN MUDHALVAN

dac-phase-4

October 31, 2023

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import plotly.express as px
```

[2]:
```
df = pd.read_csv("water_potability.csv")
```

[3]:
```
df.head
```

[3]:
```
<bound method NDFrame.head of             ph    Hardness       Solids
Chloramines     Sulfate  \
0          NaN  204.890456  20791.31898     7.300212  368.516441
1     3.716080  129.422921  18630.05786     6.635246         NaN
2     8.099124  224.236259  19909.54173     9.275884         NaN
3     8.316766  214.373394  22018.41744     8.059332  356.886136
4     9.092223  181.101509  17978.98634     6.546600  310.135738
…          …         …          …        …          …
3271  4.668102  193.681736  47580.99160     7.166639  359.948574
3272  7.808856  193.553212  17329.80216     8.061362         NaN
3273  9.419510  175.762646  33155.57822     7.350233         NaN
3274  5.126763  230.603758  11983.86938     6.303357         NaN
3275  7.874671  195.102299  17404.17706     7.509306         NaN

      Conductivity  Organic_carbon  Trihalomethanes  Turbidity  Potability
0       564.308654       10.379783        86.990970   2.963135           0
1       592.885359       15.180013        56.329076   4.500656           0
2       418.606213       16.868637        66.420093   3.055934           0
3       363.266516       18.436525       100.341674   4.628771           0
4       398.410813       11.558279        31.997993   4.075075           0
…            …             …              …          …         …
3271    526.424171       13.894419        66.687695   4.435821           1
3272    392.449580       19.903225              NaN   2.798243           1
3273    432.044783       11.039070        69.845400   3.298875           1
3274    402.883113       11.168946        77.488213   4.708658           1
3275    327.459761       16.140368        78.698446   2.309149           1

[3276 rows x 10 columns]>
```

[4]:
```python
df.info(memory_usage="deep")
```

```
<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
```

[5]:
```python
print(df.shape)
print(len(df))
print(f'Number of rows: {df.shape[0]} \nNumber of columns: {df.shape[1]}')
```

```
(3276, 10)
3276
Number of rows: 3276
Number of columns: 10
```

[6]:
```python
df.describe()
```

[6]:
```
                ph     Hardness        Solids  Chloramines      Sulfate  \
count  2785.000000  3276.000000   3276.000000  3276.000000  2495.000000
mean      7.080795   196.369496  22014.092526     7.122277   333.775777
std       1.594320    32.879761   8768.570828     1.583085    41.416840
min       0.000000    47.432000    320.942611     0.352000   129.000000
25%       6.093092   176.850538  15666.690300     6.127421   307.699498
50%       7.036752   196.967627  20927.833605     7.130299   333.073546
75%       8.062066   216.667456  27332.762125     8.114887   359.950170
max      14.000000   323.124000  61227.196010    13.127000   481.030642

       Conductivity  Organic_carbon  Trihalomethanes    Turbidity  Potability
count   3276.000000     3276.000000      3114.000000  3276.000000 3276.000000
mean     426.205111       14.284970        66.396293     3.966786    0.390110
std       80.824064        3.308162        16.175008     0.780382    0.487849
min      181.483754        2.200000         0.738000     1.450000    0.000000
25%      365.734414       12.065801        55.844536     3.439711    0.000000
```

```
50%     421.884968   14.218338    66.622485   3.955028   0.000000
75%     481.792305   16.557652    77.337473   4.500320   1.000000
max     753.342620   28.300000   124.000000   6.739000   1.000000
```

[7]:
```
df.describe?
```

[8]:
```
df.isnull().sum()
```

[8]:
```
ph                 491
Hardness             0
Solids               0
Chloramines          0
Sulfate            781
Conductivity         0
Organic_carbon       0
Trihalomethanes    162
Turbidity            0
Potability           0
dtype: int64
```

[9]:
```
def isnull_prop(df):
    total_rows = df.shape[0]
    missing_val_dict = {}
    for col in df.columns:
        missing_val_dict[col] = [df[col].isnull().sum(), (df[col].isnull().
 ↪sum() / total_rows)]
    return missing_val_dict
null_dict = isnull_prop(df)
print(null_dict.items())
```

```
dict_items([('ph', [491, 0.14987789987789987]), ('Hardness', [0, 0.0]),
('Solids', [0, 0.0]), ('Chloramines', [0, 0.0]), ('Sulfate', [781,
0.23840048840048841]), ('Conductivity', [0, 0.0]), ('Organic_carbon', [0, 0.0]),
('Trihalomethanes', [162, 0.04945054945054945]), ('Turbidity', [0, 0.0]),
('Potability', [0, 0.0])])
```

[10]:
```
import pandas as pd

# Check for NaN values in your DataFrame (assuming df is your DataFrame)
print(df.isnull().sum())
```

```
ph                 491
Hardness             0
Solids               0
Chloramines          0
Sulfate            781
Conductivity         0
```

```
Organic_carbon        0
Trihalomethanes     162
Turbidity             0
Potability            0
dtype: int64
```

[11]:
```python
df_missing = pd.DataFrame.from_dict(null_dict,
                                    orient="index",
                                    columns=['missing', 'miss_percent'])
df_missing
```

[11]:

	missing	miss_percent
ph	491	0.149878
Hardness	0	0.000000
Solids	0	0.000000
Chloramines	0	0.000000
Sulfate	781	0.238400
Conductivity	0	0.000000
Organic_carbon	0	0.000000
Trihalomethanes	162	0.049451
Turbidity	0	0.000000
Potability	0	0.000000

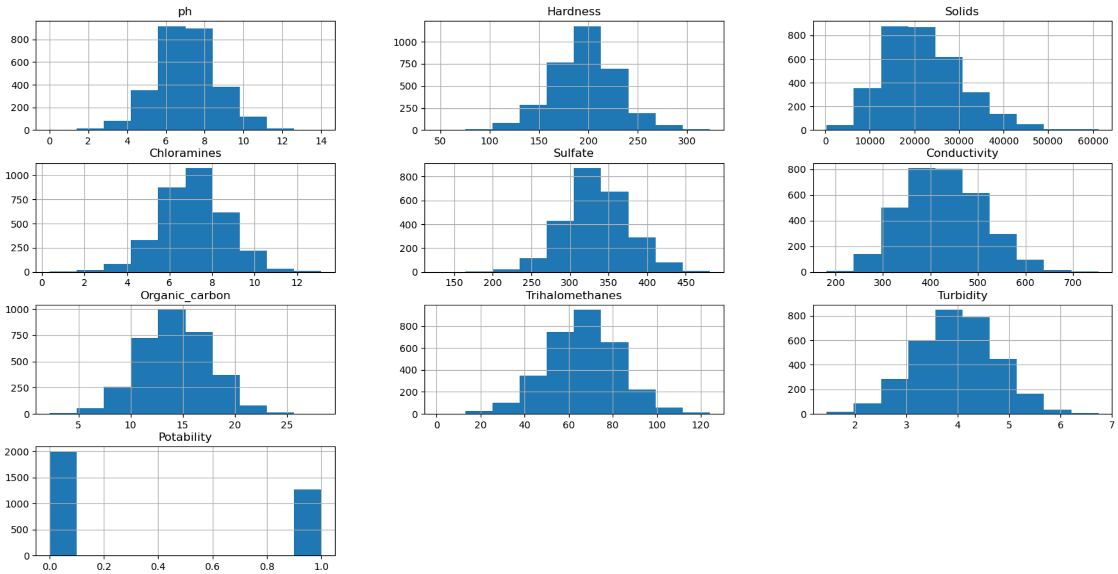
[33]:
```python
import numpy as np

# Check for infinite values
print(np.isinf(X_train).sum())
```

```
ph                 0
Hardness           0
Solids             0
Chloramines        0
Sulfate            0
Conductivity       0
Organic_carbon     0
Trihalomethanes    0
Turbidity          0
dtype: int64
```

1 visualization

[13]:
```python
plt.rcParams['figure.figsize'] = [20,10]
df.hist()
plt.show()
```



[14]:
```python
fig = px.scatter (df, x = "ph", y = "Sulfate", color = "Potability", template = "plotly_dark",  trendline="ols")
fig.show ()
```

[15]:
```python
fig = px.scatter (df, x = "Organic_carbon", y = "Hardness", color = "Potability", template = "plotly_dark",  trendline="lowess")
fig.show ()
```

2 logestic regression

[16]:
```python
import seaborn as sns
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
```

[17]:
```python
# Creating model object
model_lg = LogisticRegression(max_iter=120,random_state=0, n_jobs=20)
```

[18]:
```python
df.fillna(df.mean(), inplace=True)  # Replace NaN with mean of the column
```

[19]:
```python
X_train=df[["ph","Hardness","Solids","Chloramines","Sulfate","Conductivity","Organic_carbon",'
y_train= df["Potability"]
```

[20]:
```python
# Training Model
model_lg.fit(X_train, y_train)
```

[20]: LogisticRegression(max_iter=120, n_jobs=20, random_state=0)

[21]:
```
X_train, X_test, y_train, y_test = train_test_split(
    ␣
  ↪df[["ph","Hardness","Solids","Chloramines","Sulfate","Conductivity","Organic_carbon","Triha
    df["Potability"],
    test_size=0.2,  # You can adjust the test size as needed
    random_state=42  # You can set a random seed for reproducibility
)
pred_lg = model_lg.predict(X_test)
```

[22]:
```
lg = accuracy_score(y_test, pred_lg)
print(lg)
```

0.6280487804878049

[23]:
```
print(classification_report(y_test,pred_lg))
```

```
              precision    recall  f1-score   support

           0       0.63      1.00      0.77       412
           1       0.00      0.00      0.00       244

    accuracy                           0.63       656
   macro avg       0.31      0.50      0.39       656
weighted avg       0.39      0.63      0.48       656
```

C:\Users\lab\anaconda3\lib\site-
packages\sklearn\metrics_classification.py:1318: UndefinedMetricWarning:

Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.

C:\Users\lab\anaconda3\lib\site-
packages\sklearn\metrics_classification.py:1318: UndefinedMetricWarning:

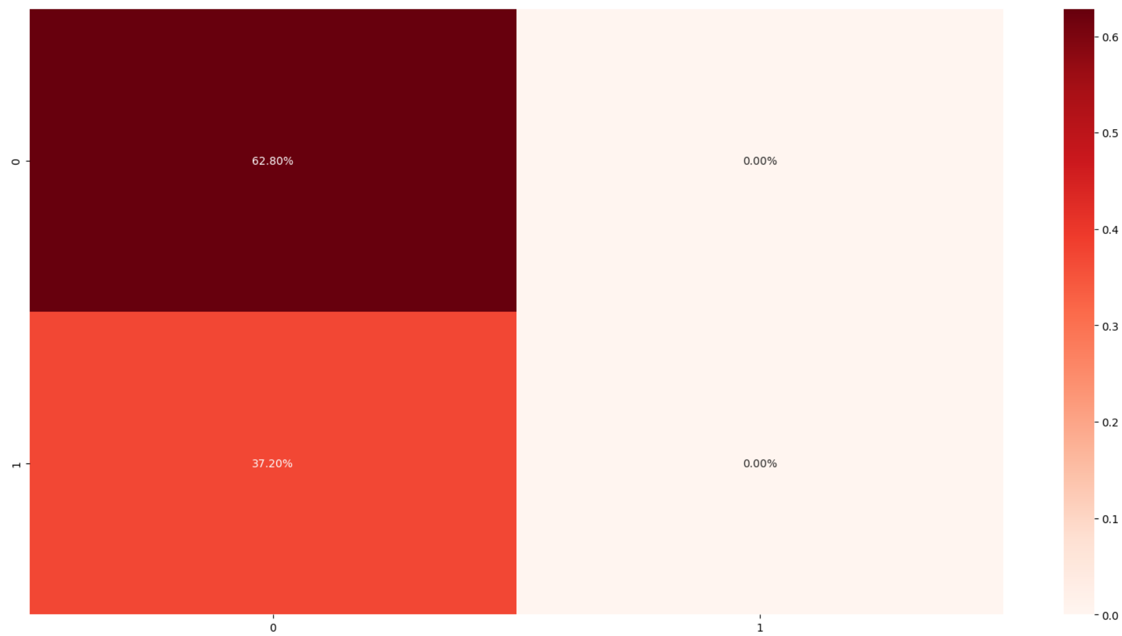
Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.

C:\Users\lab\anaconda3\lib\site-
packages\sklearn\metrics_classification.py:1318: UndefinedMetricWarning:

Precision and F-score are ill-defined and being set to 0.0 in labels with no
predicted samples. Use `zero_division` parameter to control this behavior.

[24]:
```
cm1 = confusion_matrix(y_test, pred_lg)
sns.heatmap(cm1/np.sum(cm1), annot = True, fmt=  '0.2%', cmap = 'Reds')
```

[24]: <AxesSubplot:>



3 RandomForest

[25]:
```
from sklearn.ensemble import RandomForestClassifier
```

[26]:
```
model_rf = RandomForestClassifier(n_estimators=300,min_samples_leaf=1,␣
↪random_state=32)
```

[27]:
```
model_rf.fit(X_train, y_train)
```

[27]:
```
RandomForestClassifier(n_estimators=300, random_state=32)
```

[28]:
```
pred_rf = model_rf.predict(X_test)
```

[29]:
```
rf = accuracy_score(y_test, pred_rf)
print(rf)
```

```
0.6722560975609756
```

[30]:
```
print(classification_report(y_test,pred_rf))
```

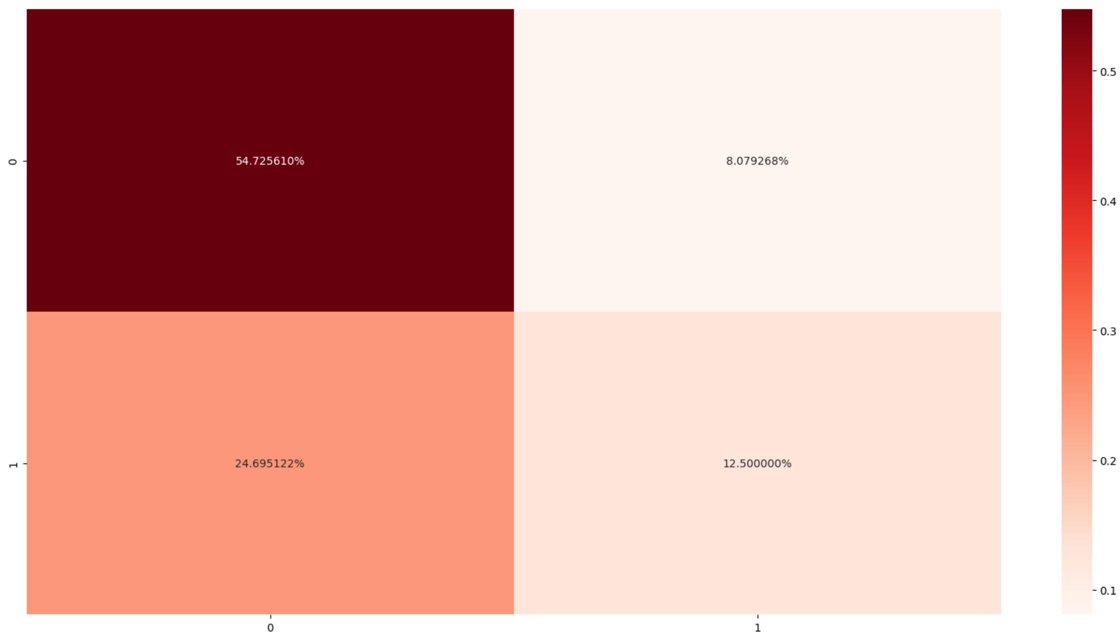
```
              precision    recall  f1-score   support

           0       0.69      0.87      0.77       412
           1       0.61      0.34      0.43       244

    accuracy                           0.67       656
   macro avg       0.65      0.60      0.60       656
weighted avg       0.66      0.67      0.64       656
```

[31]:
```
cm3 = confusion_matrix(y_test, pred_rf)
sns.heatmap(cm3/np.sum(cm3), annot = True, fmt=  '1%', cmap = 'Reds')
```

[31]: <AxesSubplot:>



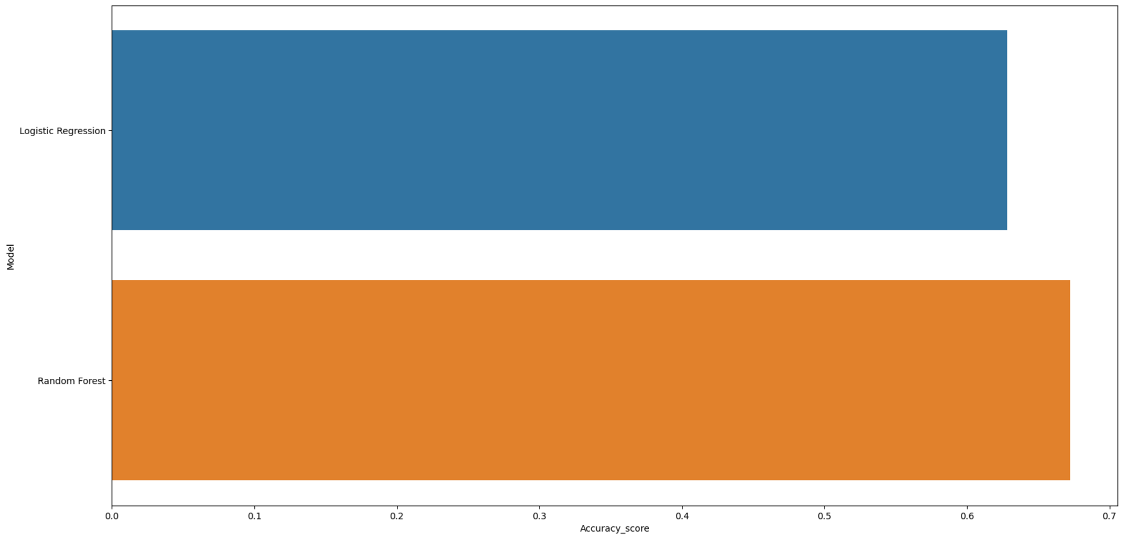
4 logestic regression vs random forest

[32]:
```
models = pd.DataFrame({
    'Model':['Logistic Regression', 'Random Forest'],
    'Accuracy_score' :[lg, rf]
})
models
sns.barplot(x='Accuracy_score', y='Model', data=models)

models.sort_values(by='Accuracy_score', ascending=False)
```

[32]:

	Model	Accuracy_score
1	Random Forest	0.672256
0	Logistic Regression	0.628049

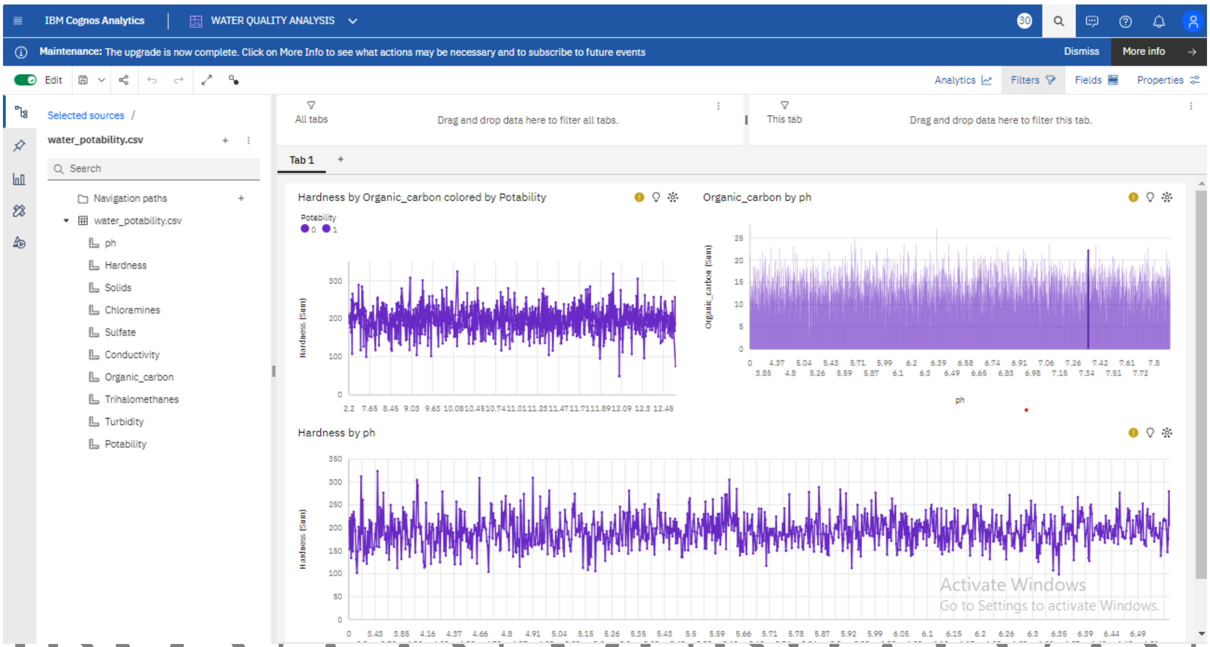


College code: 4212
Register num: 421221243005

WATER QUALITY ANALYSIS

DATA ANALYTICS WITH COGNOS:GROUP2

PHASE:4



IBM NAAN MUDHALVAN