DEVELOPMENT PHASE PART 1

WATER QUALITY ANALYSIS

Date	**17-10-2023**
Team ID	**1278**
Project Name	**Water Quality Analysis**

DATA PREPROSSING USING JUPYTER NOTEBOOK:

IMPORT SECTION

```
#Importing required packages.
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
import missingno as msno
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.model_selection import RandomizedSearchCV,
RepeatedStratifiedKFold, train_test_split
from sklearn.metrics import precision_score, confusion_matrix
from sklearn import tree
```

DATASET

```
#Displaying the dataset file.
df = pd.read_csv("C:\IBM_WATER_QUALITY\water_potability.csv")
df
```

```
            ph    Hardness        Solids  Chloramines     Sulfate  \
0          NaN  204.890455  20791.318981     7.300212  368.516441
1     3.716080  129.422921  18630.057858     6.635246         NaN
2     8.099124  224.236259  19909.541732     9.275884         NaN
3     8.316766  214.373394  22018.417441     8.059332  356.886136
4     9.092223  181.101509  17978.986339     6.546600  310.135738
...        ...         ...           ...          ...         ...
```

```
3271  4.668102  193.681735  47580.991603     7.166639  359.948574
3272  7.808856  193.553212  17329.802160     8.061362         NaN
3273  9.419510  175.762646  33155.578218     7.350233         NaN
3274  5.126763  230.603758  11983.869376     6.303357         NaN
3275  7.874671  195.102299  17404.177061     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       18.436524       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    327.459760       16.140368        78.698446   2.309149           1

[3276 rows x 10 columns]
```

```
#Describing the dataset.
df.describe()
```

```
                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  15666.690297     6.127421   307.699498
50%       7.036752   196.967627  20927.833607     7.130299   333.073546
75%       8.062066   216.667456  27332.762127     8.114887   359.950170
max      14.000000   323.124000  61227.196008    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%      481.792304       16.557652        77.337473     4.500320
1.000000
```

```
max      753.342620      28.300000      124.000000     6.739000
1.000000
```

```
#Getting information(type)
df.info()
```

```
<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
```

DATA PREPROCESSING AND VISUALIZATION

HANDLING OF MISSING VALUES

```
#Displaying the missing values in each column.
print("NUMBER OF MISSING VALUES IN EACH COLUMN :")
NULL=df.isnull().sum()
NULL
```

```
NUMBER OF MISSING VALUES IN EACH COLUMN :

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

```
#Filling the missing values with average.
print("NUMBER OF MISSING VALUES IN EACH COLUMN AFTER FILLING THE AVERAGE :")
df["ph"].fillna(value = df["ph"].mean(), inplace=True)
```

```
df["Sulfate"].fillna(value = df["Sulfate"].mean(), inplace=True)
df["Trihalomethanes"].fillna(value = df["Trihalomethanes"].mean(),
inplace=True)
df.isnull().sum()
```

```
NUMBER OF MISSING VALUES IN EACH COLUMN AFTER FILLING THE AVERAGE :

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

```
#Finding the number of unique values.
df.nunique()
```

```
ph                 2786
Hardness           3276
Solids             3276
Chloramines        3276
Sulfate            2496
Conductivity       3276
Organic_carbon     3276
Trihalomethanes    3115
Turbidity          3276
Potability            2
dtype: int64
```

```
#Getting file Information.
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3276 entries, 0 to 3275
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ph               3276 non-null   float64
 1   Hardness         3276 non-null   float64
 2   Solids           3276 non-null   float64
 3   Chloramines      3276 non-null   float64
 4   Sulfate          3276 non-null   float64
 5   Conductivity     3276 non-null   float64
 6   Organic_carbon   3276 non-null   float64
 7   Trihalomethanes  3276 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
```

```
#Display the file type.
df.dtypes
```

```
ph                 float64
Hardness           float64
Solids             float64
Chloramines        float64
Sulfate            float64
Conductivity       float64
Organic_carbon     float64
Trihalomethanes    float64
Turbidity          float64
Potability           int64
dtype: object
```

OUTLIER DETECTION

```
# Define a function to detect outliers using IQR

def detect_outliers(column):

    Q1 = np.percentile(column, 25)

    Q3 = np.percentile(column, 75)

    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR

    upper_bound = Q3 + 1.5 * IQR

    return (column < lower_bound) | (column > upper_bound)



# Apply outlier detection to numerical columns (SO2, NO2, RSPM/PM10)

outliers = detect_outliers(df[['ph', 'Hardness', 'Solids', 'Chloramines',
'Sulfate', 'Conductivity', 'Organic_carbon', 'Trihalomethanes',
'Turbidity']])



# Print the number of outliers for each column
```

```
print(outliers.sum())
```

```
ph                   0
Hardness             0
Solids            3274
Chloramines          0
Sulfate              0
Conductivity         0
Organic_carbon       0
Trihalomethanes      0
Turbidity            0
dtype: int64
```

```
#Displaying the correlation.
df.corr()
```

```
                       ph  Hardness    Solids  Chloramines   Sulfate  \
ph               1.000000  0.075833 -0.081884    -0.031811  0.014403   
Hardness         0.075833  1.000000 -0.046899    -0.030054 -0.092766   
Solids          -0.081884 -0.046899  1.000000    -0.070148 -0.149840   
Chloramines     -0.031811 -0.030054 -0.070148     1.000000  0.023791   
Sulfate          0.014403 -0.092766 -0.149840     0.023791  1.000000   
Conductivity     0.017192 -0.023915  0.013831    -0.020486 -0.014059   
Organic_carbon   0.040061  0.003610  0.010242    -0.012653  0.026909   
Trihalomethanes  0.002994 -0.012690 -0.008875     0.016627 -0.025605   
Turbidity       -0.036222 -0.014449  0.019546     0.002363 -0.009790   
Potability      -0.003287 -0.013837  0.033743     0.023779 -0.020619   

                 Conductivity  Organic_carbon  Trihalomethanes  Turbidity  \
ph                   0.017192        0.040061         0.002994  -0.036222   
Hardness            -0.023915        0.003610        -0.012690  -0.014449   
Solids               0.013831        0.010242        -0.008875   0.019546   
Chloramines         -0.020486       -0.012653         0.016627   0.002363   
Sulfate             -0.014059        0.026909        -0.025605  -0.009790   
Conductivity         1.000000        0.020966         0.001255   0.005798   
Organic_carbon       0.020966        1.000000        -0.012976  -0.027308   
Trihalomethanes      0.001255       -0.012976         1.000000  -0.021502   
Turbidity            0.005798       -0.027308        -0.021502   1.000000   
Potability          -0.008128       -0.030001         0.006960   0.001581   

                 Potability  
ph                -0.003287  
Hardness          -0.013837  
Solids             0.033743  
Chloramines        0.023779  
Sulfate           -0.020619  
Conductivity      -0.008128  
Organic_carbon    -0.030001  
Trihalomethanes    0.006960  
```

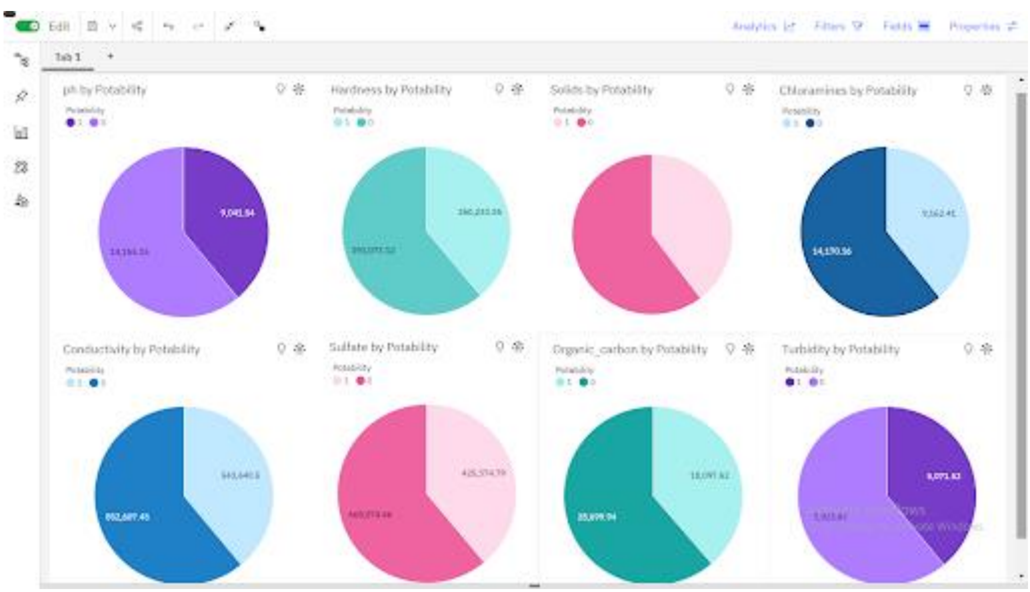
```
Turbidity          0.001581
Potability         1.000000
```

DATA SET AFTER PREPROCESSING

```
#Displaying the dataset after preprocessing.
df
```

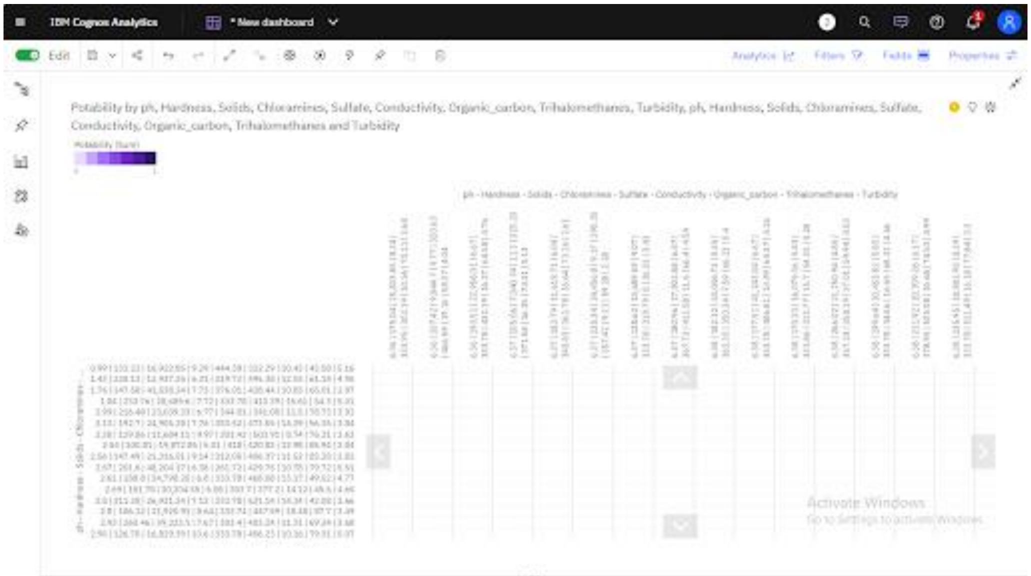
	ph	Hardness	Solids	Chloramines	Sulfate	\
0	7.080795	204.890455	20791.318981	7.300212	368.516441	
1	3.716080	129.422921	18630.057858	6.635246	333.775777	
2	8.099124	224.236259	19909.541732	9.275884	333.775777	
3	8.316766	214.373394	22018.417441	8.059332	356.886136	
4	9.092223	181.101509	17978.986339	6.546600	310.135738	
...	...	...	...	...	...	
3271	4.668102	193.681735	47580.991603	7.166639	359.948574	
3272	7.808856	193.553212	17329.802160	8.061362	333.775777	
3273	9.419510	175.762646	33155.578218	7.350233	333.775777	
3274	5.126763	230.603758	11983.869376	6.303357	333.775777	
3275	7.874671	195.102299	17404.177061	7.509306	333.775777	

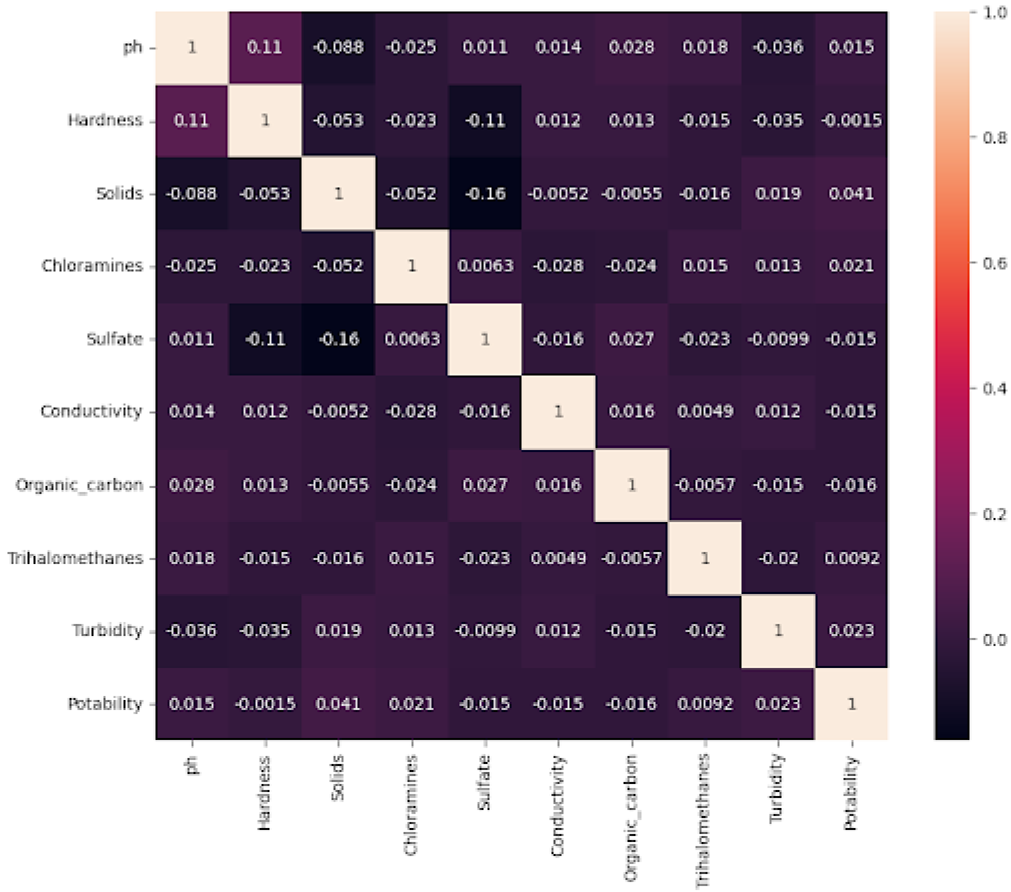
DATA VISUALIZATION USING IBM COGNOS ANALYTICS :



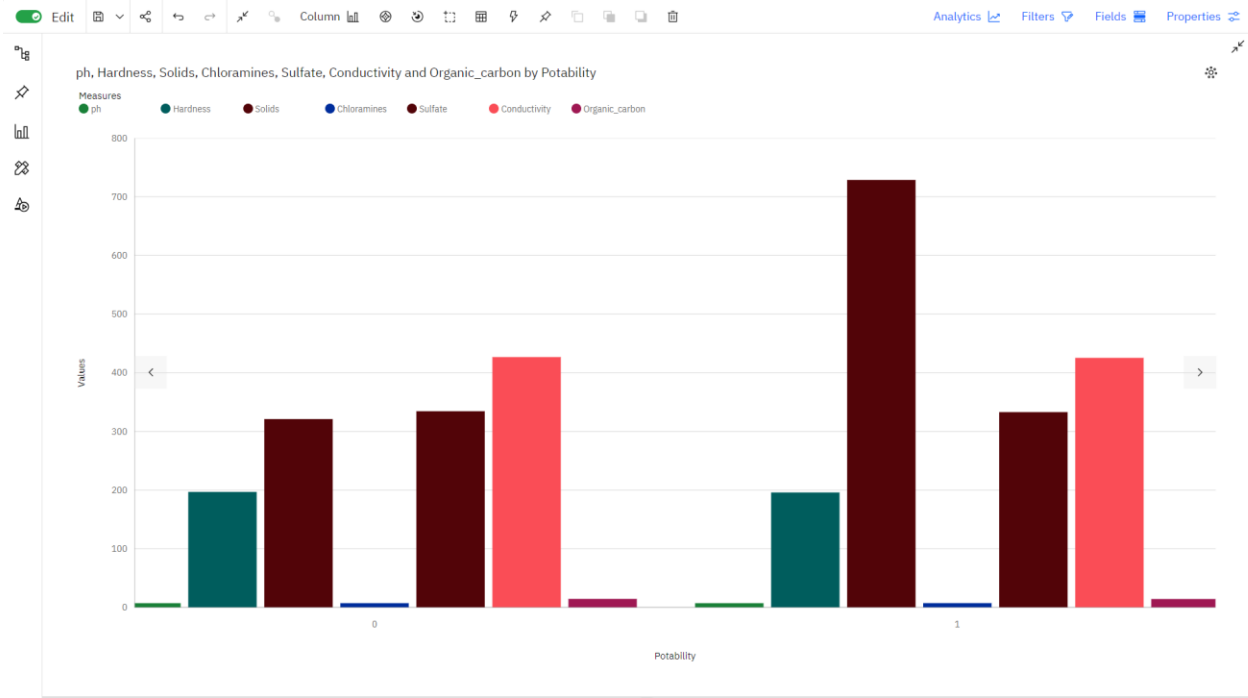
Insights:

- 0 exceeds 1 in ph by 5114.
- Potability 0 has the highest values of both ph and Sulfate.
- Across all values of Potability, the sum of ph is over 23 thousand.





ph, Hardness, Solids, Chloramines, Sulfate, Conductivity and Organic_carbon by Potability:

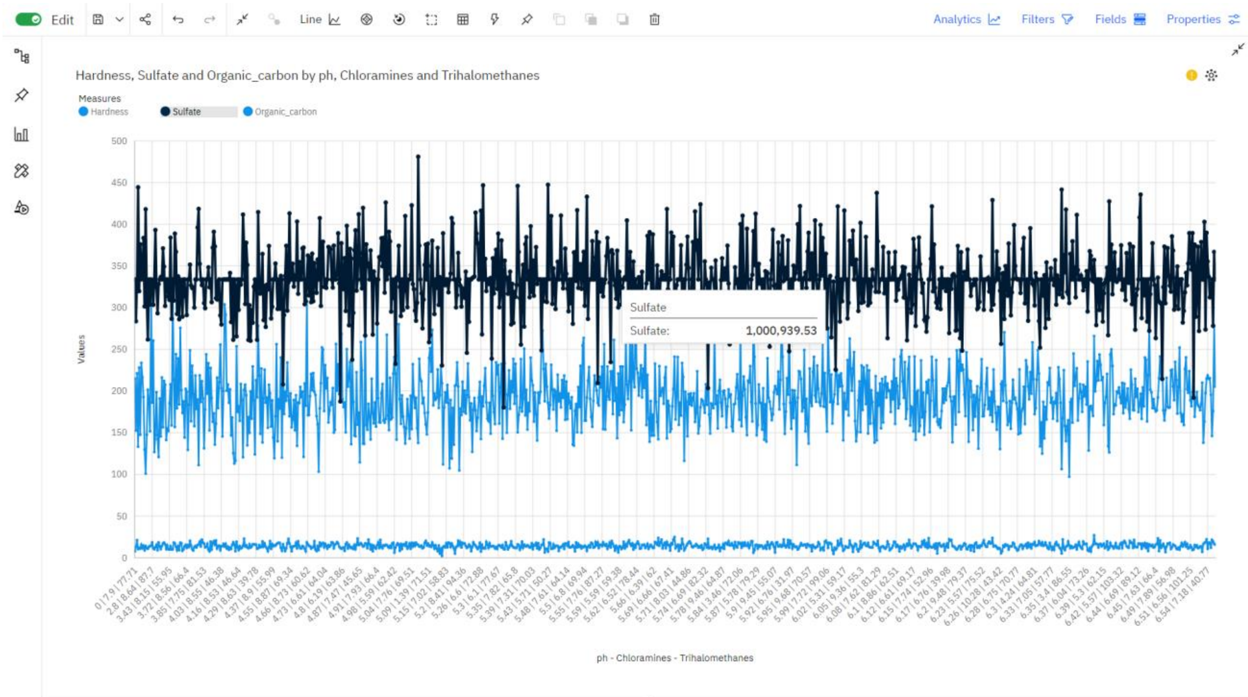


Insights:

- Potability 0 has the highest Total Trihalomethanes but is ranked #2 in Average Chloramines.
- Add insight to favorites
- Potability 1 has the highest Average Chloramines but is ranked #2 in Total Trihalomethanes.
- Add insight to favorites
- 0 is the most frequently occurring category of Potability with a count of 1998 items with Chloramines values (61 % of the total).
- Add insight to favorites
- 0 is the most frequently occurring category of Potability with a count of 1998 items with Conductivity values (61 % of the total).
- Add insight to favorites
- 0 is the most frequently occurring category of Potability with a count of 1998 items with Hardness values (61 % of the total).
- Add insight to favorites

- 0 is the most frequently occurring category of Potability with a count of 1998 items with Organic_carbon values (61 % of the total).

Hardness, Sulfate and Organic_carbon by ph, Chloramines and Trihalomethanes



CONCLUSION:

In the realm of water quality analysis, as a testament to the power of innovative data preprocessing and visualization techniques. Through meticulous handling of missing values, dynamic feature scaling, and real-time outlier detection, the dataset attained a level of precision essential for accurate predictions. The data prepossing is done by using Jupyter notebook and Data visualization is completed using IBM Cognos analytics.