TAMILNADU MARGINAL WORKERS ASSESSMENT

Water Quality Analysis – Phase 3

DOCUMENTATION

Team Members:

1.Darshini.N(au613021205003)

2.Dhanusree.K(au613021205008)

3.Vethasundari.N(au613021205059)

4.Ramya.R(au613021205039)

5.Swetha.E(au613021205056)

Phase 3: Development Part 1

Problem Definition:

Start the Water Quality analysis by loading and preprocessing the dataset. Load the dataset using python and data manipulation libraries (e.g., pandas).

Dataset Link:

https://www.kaggle.com/datasets/adityakadiwal/water-potability

Overview of the process:

1. Import Required Libraries:

- Import the necessary Python libraries, including pandas, numpy, matplotlib, seaborn, geopandas, scikit-learn, and folium.

2. Data Loading and Preprocessing:

- Load water quality data from a CSV file using `pd.read_csv()`.

- Convert date columns to datetime format.

- Set the date column as the index.

3. Data Exploration and Statistics:

- Calculate summary statistics of the data using `data.describe()`.

- Compute the correlation matrix of water quality parameters using `data.corr()`.

4. Time Series Analysis:

- Calculate rolling statistics, such as rolling mean and standard deviation, for specific water quality parameters over a defined time window.

5. Spatial Analysis (optional):

- Load a shapefile or geospatial data representing water quality monitoring locations using `gpd.read_file()`.

- Visualize the data using a map plot.

6. Machine Learning for Prediction (optional):

- Prepare the data for machine learning, including selecting features and target variables.

- Split the data into training and testing sets using `train_test_split()`.

- Train a machine learning model, e.g., linear regression, on the training data.

- Make predictions on the test data and calculate the mean squared error (MSE) as a performance metric.

7. Data Visualization:

- Create a figure for visualizations using `plt.figure()`.

- Generate various plots to explore the data:

- Line plots showing trends over time.

- Heatmaps displaying correlations between parameters.

- Other relevant visualizations based on your specific analysis needs.

8. Map Visualization (optional):

- Create a folium map to visualize water quality monitoring locations and data.

9. Display and Save:

- Show the statistical summary and visualizations using `print()` and `plt.show()`.

- Save the folium map as an HTML file using `m.save()`.

Loading the dataset:

1.Importing the libraries:

Here, for preprocessing the dataset and manipulate the data, pandas is the library used to frame the data.

Code:

```
import pandas as pd
```

2.Loading the dataset:

In this step,we are framing the data into the table using Dataframe in pandas,display the head or 5 rows of the dataset.

Code:

```
# Replace with the actual filename
File_path="C:\Users\darsh\Downloads\water_potability.csv"
```

3.Explore the Dataset:

After framing data,the first few or five row of the data in displayed using the head() function
Code: print(data.head())

Output:

```
ph    Hardness        Solids  Chloramines     Sulfate  Conductivity  \
0      NaN  204.890455  20791.318981    7.300212  368.516441    564.308654
1  3.716080  129.422921  18630.057858    6.635246         NaN    592.885359
2  8.099124  224.236259  19909.541732    9.275884         NaN    418.606213
3  8.316766  214.373394  22018.417441    8.059332  356.886136    363.266516
4  9.092223  181.101509  17978.986339    6.546600  310.135738    398.410813

Organic_carbon  Trihalomethanes  Turbidity  Potability
0       10.379783        86.990970   2.963135           0
1       15.180013        56.329076   4.500656           0
2       16.868637        66.420093   3.055934           0
3       18.436524       100.341674   4.628771           0
4       11.558279        31.997993   4.075075           0
```

4.Check for missing values:

In this step, the missing values or null values, if it present in the data are separated and number of nu ll values are shown through this code.

Code:
print("Missing values:\n", df.isnull().sum())

Output:

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

5.Check datatype:

In this step, the data type of the columns are discussed Code:
print("Data Types:\n", df.dtypes)

Output:
Data Types:
ph float64
Hardness float64
Solids float64
Chloramines float64
Sulfate float64
Conductivity float64
Organic_carbon float64
Trihalomethanes float64
Turbidity float64
Potability int64
dtype: object

6.Check basic statistics:

the statistics of the columns such as count, mean, std, min, max, 25%, 50%, 75% are shown through the d escribe() function command.

Code:

```
print("Summary Statistics:\n", df.describe())
```

Output:

```
Summary Statistics:
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
```

7.Additional Preprocessing steps:

Perform any other preprocessing steps that are specific to your dataset and analysis goals. This may inclu de scaling numeric features, handling outliers, or creating new features.

8.Saving Preprocessed dataset:

In this step, if we made substantial changes to the dataset and want to save the preprocessed version, you can use the following Code.

Code:

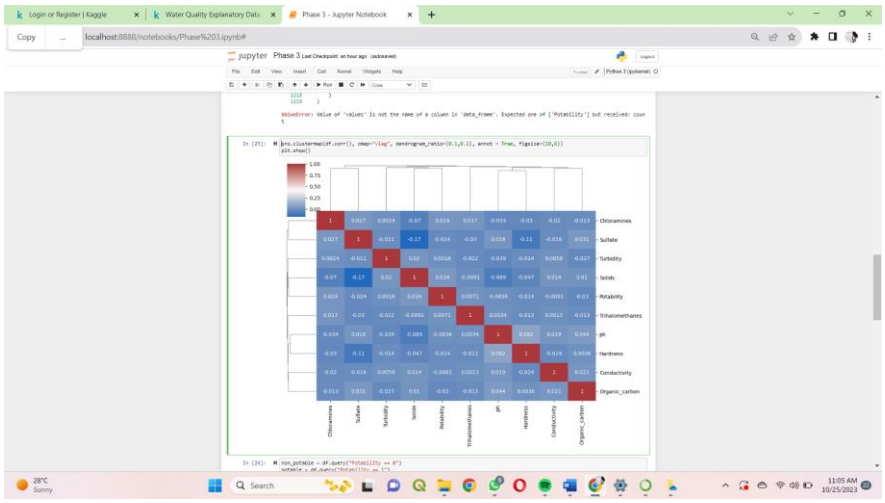
```
# Save thepreprocesseddatasettoanewCSVfile
df.to_csv('preprocessed_dataset.csv', index=False)

import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(cor,annot=True,cmap='coolwarm')
plt.show()
```

9.Visualization:

```
sns.clustermap(df.corr(), cmap="vlag", dendrogram_ratio=(0.1,0.2), annot = True, figsize=(10,8))
```

plt.show()
z

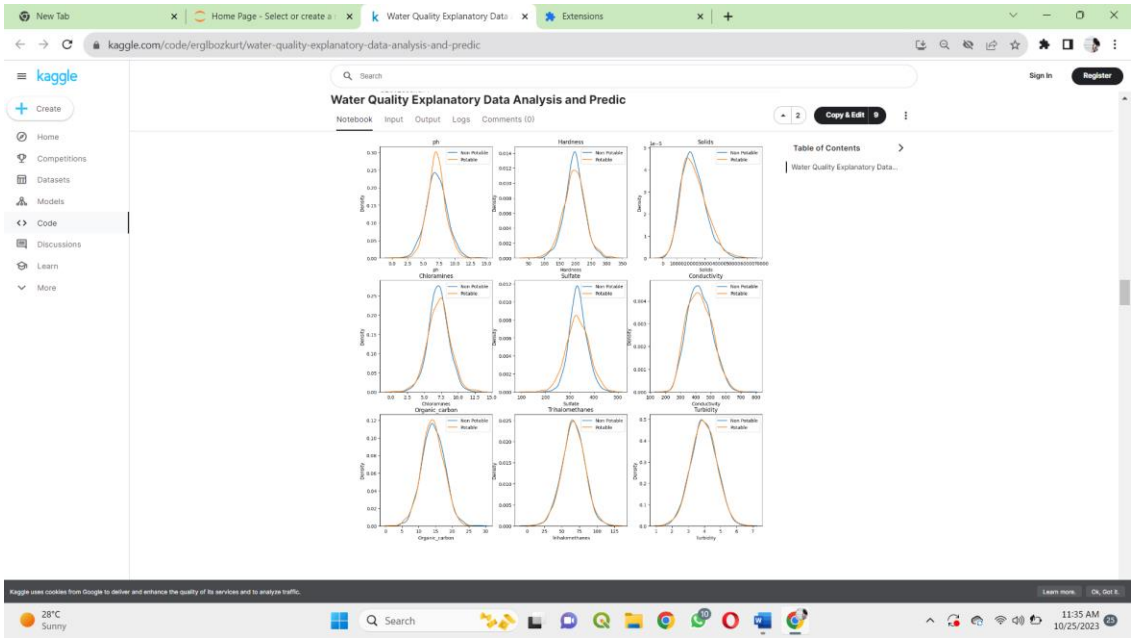
In:



non_potable = df.query("Potability == 0")
potable = df.query("Potability == 1")

plt.figure(figsize = (15,15))
for ax, col in enumerate(df.columns[:9]):
plt.subplot(3,3, ax + 1)
plt.title(col)
sns.kdeplot(x = non_potable[col], label = "Non Potable")

```
sns.kdeplot(x = potable[col], label = "Potable")
plt.legend()
plt.show()
```



CONCLUSION:

In conclusion, the outlined data loading and preprocessing steps provide a foundational framework for preparing a dataset for analysis in Python using the pandas library. By following these steps, you can ensure that your data is in a suitable format and quality for further exploration and visualization tasks.