WATER QUALITY ANALYSIS

**PHASE 3: DEVELOPMENT**

**SUBMITTED BY:**

**Chinthakunta matam praveenkumar**

**Au723921243012**

**Praveen9949328880@gmail.com**

**Start building the water quality analysis by preprocessing the data and performing exploratory data analysis:**

**1.Start building the water quality analysis by preprocessing the data and performing exploratory data analysis.**

**import pandas as pd**

**# Load the data into a pandas DataFrame**

**data = pd.read\_csv('water\_quality\_data.csv')**

**# Check the data types**

**print(data.dtypes)**

**# Check for missing values**

**print(data.isnull().sum())**

**# Display the first few rows of the data**

**print(data.head())**

**2.Exploratory Data Analysis: Conduct exploratory data analysis to better understand the data. This may include analyzing the distributions of different parameters, identifying outliers, and visualizing relationships between parameters.**

**# Analyze the distributions of different parameters**

**data.describe()**

**# Visualize the distributions of different parameters**

**data.hist(figsize=(15,10))**

**plt.show()**

**# Identify outliers using box plots**

**data.boxplot(figsize=(15,10))**

**plt.show()**

**# Visualize the relationships between parameters using a correlation matrix**

**corr = data.corr()**

**sns.heatmap(corr, annot=True, cmap='coolwarm')**

**plt.show()**

**3.Data Preparation: Based on the exploratory data analysis, perform necessary data preprocessing tasks such as encoding categorical variables, normalizing numerical variables, and handling missing values.**

**# Encoding categorical variables**

**from sklearn.preprocessing import LabelEncoder**

**encoder = LabelEncoder()**

**data['categorical\_variable'] = encoder.fit\_transform(data['categorical\_variable'])**

**# Normalizing numerical variables**

**from sklearn.preprocessing import StandardScaler**

**scaler = StandardScaler()**

**data[['numerical\_variable1', 'numerical\_variable2']] = scaler.fit\_transform(data[['numerical\_variable1', 'numerical\_variable2']])**

**# Handling missing values**

**data.fillna(method='ffill', inplace=True)**

**Obtain the water quality dataset and preprocess it by handling missing values and outliers:**

**1.Obtain the Water Quality Dataset: You can download the Water Quality Dataset from various sources like Kaggle or the United States Environmental Protection Agency (EPA) website. The dataset usually contains information about different water quality parameters, geographical locations, and time.**

**2.Preprocess the Dataset: The next step is to preprocess the dataset by handling missing values and outliers. Here is an example of how to do this:**

**import pandas as pd**

**import numpy as np**

**from sklearn.preprocessing import StandardScaler**

**# Load the Water Quality Dataset**

**data = pd.read\_csv('water\_quality\_data.csv')**

**# Check for missing values**

**print(data.isnull().sum())**

**# Impute missing values with the mean value of each column**

**for column in data.columns:**

**data[column].fillna(data[column].mean(), inplace=True)**

**# Detect and handle outliers using Z-score method**

**z\_scores = np.abs((data - data.mean()) / data.std())**

**# Define a threshold for the Z-score**

**threshold = 3**

**# Remove outliers based on the threshold**

**data = data[(z\_scores < threshold).all(axis=1)]**

**# Normalize numerical variables using StandardScaler**

**scaler = StandardScaler()**

**data[['numerical\_variable1', 'numerical\_variable2']] = scaler.fit\_transform(data[['numerical\_variable1', 'numerical\_variable2']])**

**Conduct EDA to visualize parameter distributions, correlations, and potential deviations from:**

**1.First, let's import the necessary libraries and load the dataset.**

**import pandas as pd**

**import numpy as np**

**import matplotlib.pyplot as plt**

**import seaborn as sns**

**# Load the dataset**

**df = pd.read\_csv('water\_quality\_data.csv')**

**2.Next, let's examine the basic information about the dataset.**

**# Print the first few rows of the dataset**

**print(df.head())**

**# Print the data types of each column**

**print(df.dtypes)**

**# Check for missing values**

**print(df.isnull().sum())**

**3.Now, let's visualize the distribution of parameter values using histograms.**

**# Plot histograms for each parameter**

**df.hist(bins=30, figsize=(20,15))**

**plt.show()**

**4.To examine correlations between different parameters, let's plot a correlation matrix.**

**# Calculate the correlation matrix**

**corr = df.corr()**

**# Generate a mask for the upper triangle of the correlation matrix**

**mask = np.triu(np.ones\_like(corr, dtype=bool))**

**# Plot the correlation matrix using a heatmap**

**plt.figure(figsize=(10,8))**

**sns.heatmap(corr, mask=mask, cmap='coolwarm', annot=True, fmt=".2f")**

**plt.show()**

**5.Finally, let's visualize potential deviations from normality in the parameter distributions by plotting Q-Q plots:**

**Python.**

**# Generate Q-Q plots for each parameter**

**df.plot(kind='qq', figsize=(20,15), subplots=True)**

**plt.show()**

**These steps should provide a comprehensive overview of the dataset's parameters, correlations, and potential deviations from normality.**