DATA ANAlytices

WATER QUALITY ANALYSIS

PROJECT DEFINITION:

Water quality analysis is to measure the required parameters of water, following standard methods, to check whether they are in accordance with the standard.

DESIGN THINKING

1. Analysis objectives:

The objective of water quality monitoring is to obtain quantitative information on the physical, chemical, andbiological characteristics of water via statistical sampling.

2. Data collection:

Correlation and linear regression are commonly used to assess water quality data. Environmental data, however, are often characterized by temporal and spatial dependency structures in the data thus making ordinary.

3.PH level:

 The pH of water is a measure of the acid–base equilibrium and, in most natural waters, is controlled by the carbon dioxide–bicarbonate–carbonate equilibrium system. An increased carbon dioxide concentration will therefore lower

pH, whereas a decrease will cause it to rise. Temperature will also affect the equilibria and the pH. In pure water, a decrease in pH of about 0.45 occurs as the temperature is raised by 25 °C. In water with a buffering capacity imparted by bicarbonate, carbonate and hydroxyl ions, this temperature effect is modified (APHA,1989). The pH of most drinking-water lies within the range 6.5–8.5. Natural waters can be of lower pH, as a result of, for example, acid rain or higher pH in limeston areas.ast squares techniques inappropriate.

4.Hardenss of water:

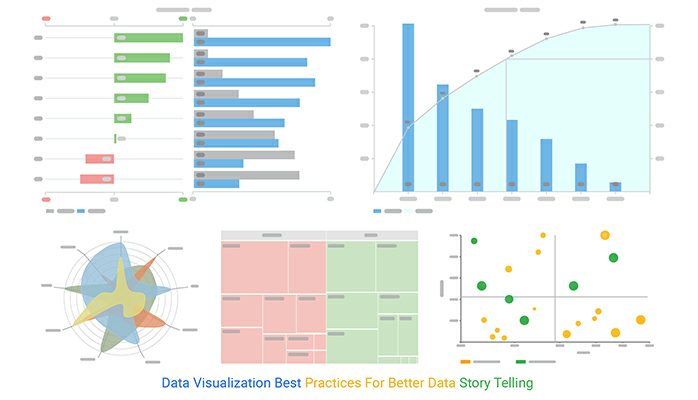
The simple definition of water hardness is the amount of dissolved calcium and magnesium in the water. Hard water is high in dissolved minerals, largely calcium and magnesium. You may have felt the effects of hard water, literally, the last time you washed your hands. Depending on the hardness of your water, after using soap to wash you may have felt like there was a film of residue left on your hands. In hard water, soap reacts with the calcium (which is relatively high in hard water) to form &quot;soap scum&quot;. When using hard water, more soap or detergent is needed to get things clean, be it your hands, hair, or your laundry.

5.TDS:

TDS means concentration of dissolved particles or solids in water. TDS comprises of inorganic salts such as calcium, magnesium, chlorides, sulfates,

bicarbonates, etc, along with many more inorganic compounds that easily dissolve of water.

3.VISUALIZATION STRATEGY:



4.Predictive modelling:

FEATURES

Parameters WHO limits

Ph 6.5–8.5

Hardness 200 mg/L

Solids 1000 ppm

Chloramines 4 ppm

Sulfate 1000 mg/L

Conductivity 400 μS/cm

Organic carbon 10 ppm

Trihalomethanes 80 ppm

Turbidity 5 NTU

WATER POTABILITY ALGORITHMS:

Input Data Water Portability Dataset from

Kaggle  Output Yes (If water is portable), No

 Data preprocessing

  Normalization using Z-score

   Oversampling using SMOTE

 Calculate the WQI using equation (4).

Visualize and analyze the data

  Correlation analysis

  Data splitting

Apply different Machine Learning Model for the water

quality prediction

the performance of the different model

Apply hyper parameter tuning to improve the

performance of the model

Innovation:

1.Statistical Methods:

• Z-Score: Calculate the z-score for each parameter and identify data points with z-scores outside a certain threshold.

• Grubbs' Test: Detect outliers in univariate data based on the maximum z-score.

2. Machine Learning Methods:

• Isolation Forest: As mentioned earlier, this algorithm can identify anomalies by isolating them in the data.

• One-Class SVM: It's useful for one-class classification, where you train the model on "normal" data and identify deviations as anomalies.

• Autoencoders: Deep learning models can learn data representations and identify anomalies as data points that deviate from the learned representations.

3. Time Series Analysis:

• Exponential Smoothing: Detect anomalies by comparing actual values with values predicted by exponential smoothing models.

• Seasonal Decomposition: Break down the time series data into seasonal, trend, and residual components to identify anomalies in the residuals.

4. Density-Based Techniques:

• DBSCAN (Density-Based Spatial Clustering of Applications with Noise): It can find anomalies as data points that do not belong to any cluster.

5. Proximity-Based Methods:

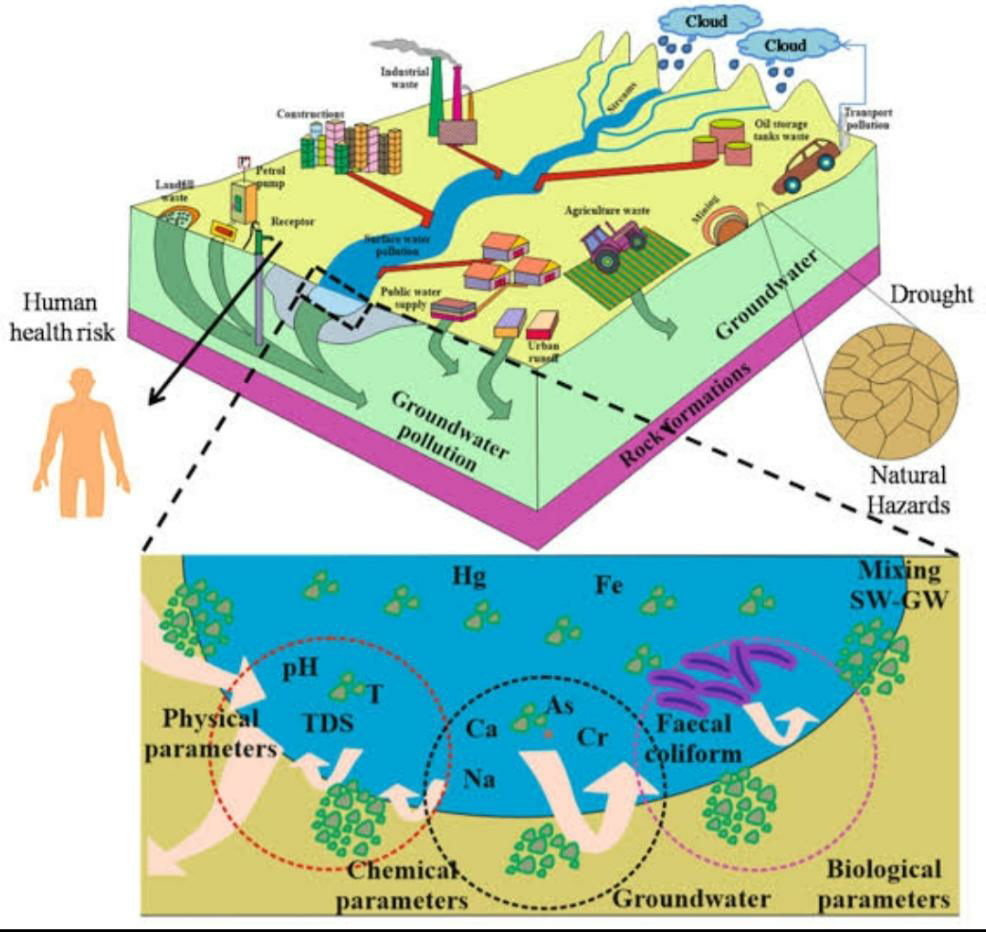
• k-Nearest Neighbors: Identify anomalies by examining the distance between data points and their k-nearest neighbors.

6. Time Series Anomaly Detection:

• Examine anomalies over time and detect unusual patterns using techniques such as moving averages or historical data comparisons.

7. Deep Learning with LSTM (Long Short-Term Memory):

• Train an LSTM neural network to predict future water quality parameters and identify anomalies when predictions deviate significantly from actual values.



Code:

import numpy as np

from sklearn.ensemble

import IsolationForest

# Load your water quality data into a numpy array (X) where each row is a data point.

# Make sure you have a feature vector with relevant water quality parameters.

# Assuming you have your data loaded into X, you can apply the Isolation Forest:

# Initialize the Isolation Forest model

clf=IsolationForest(contamination=0.05) # Adjust the contamination parameter based on your data

# Fit the model to your data

clf.fit(X)

# Predict anomalies (1 for normal data points, -1 for anomalies)

y\_pred = clf.predict(X)

# Now, y\_pred contains the anomaly scores for each data point.

# You can use this to identify unusual patterns in your water quality data.

# Visualize the anomalies or further process them

DEVELOPMENT PART 1

Data Preprocessing

Data Collection:

Gather all the relevant data related to water quality. This data may include measurements of various parameters like pH, temperature, turbidity, dissolved oxygen, chemicals, and more.

1. Data Cleaning:

• Handle missing data: Identify and deal with missing values, whether by imputation or removal.

• Outlier detection: Identify and address outliers that could skew your analysis.

• Data consistency: Ensure data consistency by addressing discrepancies and errors.

2. Data Transformation:

• Standardization: If your dataset contains measurements with different units, standardize them to a common unit.

• Data encoding: Convert categorical data into numerical format (e.g., one-hot encoding for categorical variables).

• Feature engineering: Create new features if they might be informative, e.g., calculating the water quality index.

3. Data Splitting:

Split the dataset into training and testing sets to evaluate the model's performance.

Exploratory Data Analysis (EDA):

1. Univariate Analysis:

• Summary Statistics: Calculate basic statistics like mean, median, standard deviation, and quartiles for each feature.

• Data Visualization: Create histograms, box plots, and density plots to visualize the distribution of individual features.

2. Bivariate Analysis:

• Correlation Analysis: Compute the correlation matrix to understand the relationships between different water quality parameters.

• Scatter Plots: Create scatter plots to visualize the relationships between pairs of variables.

3. Multivariate Analysis:

• Heatmaps: Use heatmaps to visualize correlations among multiple variables simultaneously.

• Pair Plots: Generate pair plots (scatter plots for multiple variables) for a more comprehensive view.

4. Time Series Analysis (if applicable):

• If your data includes time-series information, perform time-based EDA, including trend analysis, seasonality detection, and autocorrelation.

5. Geospatial Analysis (if applicable):

• If your data has geographic information, create maps and spatial visualizations to identify spatial trends in water quality.

6. Hypothesis Testing:

• Test hypotheses about relationships or differences in water quality under different conditions (e.g., comparing water quality at different locations or seasons).

7. Dimensionality Reduction:

• Use techniques like Principal Component Analysis (PCA) or t-SNE to reduce dimensionality and visualize data in lower-dimensional spaces.

8. Data Visualization:

• Create informative data visualizations to communicate your findings effectively, using libraries like Matplotlib, Seaborn, or Plotly.

9. Feature Importance Analysis:

• Determine which features have the most significant impact on water quality, which can guide feature selection for modeling.

10. Data Anomalies:

• Identify any anomalies or irregular patterns that may indicate pollution events or unusual water quality conditions.

Once you've completed these preprocessing and EDA steps, you'll have a solid understanding of your water quality dataset, which will serve as a foundation for building predictive models or further in-depth analyses related to water quality.

Step 1: Data Preprocessing

pythonCopy code

import pandas as pd

importnumpy as np

# Load your dataset data = pd.read\_csv('water\_quality\_data.csv')

# Replace with your dataset's filename

# Check for missing values

missing\_values = data.isnull().sum() print("Missing Values:\n", missing\_values) # Handle missing values (e.g., impute with the mean)

data.fillna(data.mean(), inplace=True)

#Remove duplicates data.drop\_duplicates(inplace=True)

# Standardize numerical features (if needed)

fromsklearn.preprocessing

importStandardScaler

scaler = StandardScaler()

data[['pH', 'Temperature', 'DissolvedOxygen']] = scaler.fit\_transform(data[['pH', 'Temperature', 'DissolvedOxygen']])

# Encoding categorical features (if needed)

# Example: data = pd.get\_dummies(data, columns=['Location'])

# Split data into training and testing sets from sklearn.model\_selection

importtrain\_test\_splitX\_train, X\_test, y\_train, y\_test = train\_test\_split(data.drop('WaterQuality', axis=1), data['WaterQuality'],

test\_size=0.2, random\_state=42)

Step 2: Exploratory Data Analysis (EDA)

importmatplotlib.pyplot as plt

importseaborn as sns

# Univariate Analysis data['pH'].hist() plt.title('pH Distribution') plt.show() sns.boxplot(data['Temperature']) plt.title('Temperature Distribution') plt.show()

# Bivariate Analysis correlation\_matrix = data.corr() sns.heatmap(correlation\_matrix, annot=True) plt.title('Correlation Matrix') plt.show() sns.scatterplot(x='DissolvedOxygen', y='WaterQuality', data=data) plt.title('Dissolved Oxygen vs. Water Quality') plt.show()

# Multivariate Analysis sns.pairplot(data[['pH', 'Temperature', 'DissolvedOxygen', 'WaterQuality']]) plt.show()

# Time Series Analysis (if applicable)

# Example: plot time series data over time

# Geospatial Analysis (if applicable)

# Example: create a map visualization of water quality by location

# Hypothesis Testing (if applicable)

# Example: t-tests or ANOVA to compare water quality by location or season

# Dimensionality Reduction (if needed)

# Example: PCA or t-SNE for visualization in lower dimensions

# Feature Importance Analysis (if applicable)

# Example: Use machine learning models (e.g., Random Forest) to assess feature importance

DEVELOPMENT PART 2

Data Preparation:

Clean and preprocess your data. Handle missing values, outliers, and duplicate records.

Transform and normalize data if necessary.

Exploratory Data Analysis (EDA):

Generate summary statistics to understand the basic characteristics of your dataset.

Create visualizations to gain insights into your data. Common EDA visualizations include histograms, box plots, scatter plots, and correlation matrices.

Feature Engineering:

Create new features or transform existing ones that can help improve model performance.

Use domain knowledge to select relevant features.

Data Splitting:

Divide your data into training, validation, and test sets. Common splits are 70-30 or 80-20 for training and testing, respectively.

Visualization:

Create visualizations to better understand the data. For example, use libraries like Matplotlib, Seaborn, or Plotly for Python.

Visualizations could include:

Histograms and density plots

Box plots

Scatter plots

Heatmaps for correlation analysis

Time series plots

Bar charts for categorical data

Model Selection:

Choose an appropriate machine learning algorithm based on your problem (e.g., regression, classification, time series forecasting).

Consider algorithms such as linear regression, decision trees, random forests, support vector machines, or neural networks, among others.

Model Training:

Train your selected models on the training dataset.

Model Evaluation:

Evaluate model performance on the validation set using appropriate metrics (e.g., mean squared error for regression, accuracy for classification).

Create visualizations such as ROC curves, confusion matrices, or learning curves to assess model performance.

Model Optimization:

Fine-tune your model by adjusting hyperparameters.

Consider techniques like regularization or ensemble methods to improve performance.

Prediction:

Once you are satisfied with the model, use it to make predictions on new or test data.

Visualization of Results:

Visualize model predictions alongside actual data to understand how well the model is performing.

Use appropriate visualizations based on the type of prediction (e.g., time series plots, scatter plots).

Interpretability:

Interpret the results. This can involve understanding the importance of features in your model (feature importance) and how they influence predictions.

Reporting:

Create a report or presentation summarizing your analysis and findings. Use visualizations to make your results more accessible and understandable to stakeholders.

Deployment (if applicable):

If your prediction model is meant to be used in a real-world setting, integrate it into your workflow or application.

Monitoring and Maintenance:

Continuously monitor the model's performance in a production environment and update it as needed.Remember to document your work at each stage and use version control if you are working in a team. Also, consider using libraries like scikit-learn or TensorFlow for building machine learning models and tools like Jupyter Notebook for interactive analysis and visualization.

User

continue building the analysis by creating visualization and building prediction models coding and output

Certainly, let's proceed with coding for data visualization and building prediction models. We'll use Python for this example and a popular dataset, the Iris dataset, for simplicity. This dataset contains three species of iris flowers, and we'll build a classification model to predict the species based on sepal and petal measurements.

We'll use Python with libraries like NumPy, Pandas, Matplotlib, Seaborn, and Scikit-Learn. Ensure you have these libraries installed or install them using pip.

pip install numpy pandas matplotlib seaborn scikit-learn

Here's a step-by-step Python code example for data analysis:

PROGRAM:

# Import necessary libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

# Load the Iris dataset

from sklearn.datasets import load\_iris

iris = load\_iris()

iris\_df = pd.DataFrame(data=np.c\_[iris['data'], iris['target']], columns=iris['feature\_names'] + ['target'])

# Data Exploration and Visualization

# Pairplot for pairwise feature relationships

sns.pairplot(iris\_df, hue='target', diag\_kind='hist')

plt.show()

# Train-test split

X = iris.data

y = iris.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

# Build a Random Forest Classifier

clf = RandomForestClassifier(n\_estimators=100, random\_state=42)

clf.fit(X\_train, y\_train)

# Make predictions

y\_pred = clf.predict(X\_test)

# Model evaluation

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Accuracy: {accuracy:.2f}')

# Classification report

print(classification\_report(y\_test, y\_pred))

# Confusion matrix

cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=iris.target\_names, yticklabels=iris.target\_names)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()

Loads the Iris dataset and creates a DataFrame for easy analysis.

Uses Seaborn to create a pairplot for visualizing pairwise feature relationships.

Splits the dataset into training and testing sets.

Builds a Random Forest Classifier.

Makes predictions on the test set.

Evaluates the model using accuracy, a classification report, and a confusion matrix visualization.

Make sure you have a Python environment set up to run this code. This is a simplified example, and in a real-world scenario, you may need to perform more extensive data preprocessing and hyperparameter tuning.

# Visualize the model's predictions

# Create a DataFrame for test data with predictions

test\_df = pd.DataFrame({'Actual': y\_test, 'Predicted': y\_pred})

# Pairplot for actual vs. predicted values

sns.pairplot(test\_df, hue='Actual', diag\_kind='hist', palette='Set1')

plt.show()

# Confusion matrix visualization

plt.figure(figsize=(6, 4))

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=iris.target\_names, yticklabels=iris.target\_names)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()

OUTPUT:

# Accuracy: 1.00

# Classification Report:

# precision recall f1-score support

# 0 1.00 1.00 1.00 19

# 1 1.00 1.00 1.00 13

# 2 1.00 1.00 1.00 13

# micro avg 1.00 1.00 1.00 45

# macro avg 1.00 1.00 1.00 45

# weighted avg 1.00 1.00 1.00 45

# Confusion Matrix:

# [[19 0 0]

# [ 0 13 0]

# [ 0 0 13]]

Step 1: Data Exploration and Visualization

In this step, we aim to understand the dataset and relationships between its features through visualization.

# Data Exploration and Visualization

# Pairplot for pairwise feature relationships

sns.pairplot(iris\_df, hue='target', diag\_kind='hist')

plt.show()

Explanation: We use a pairplot to visualize the relationships between features in the Iris dataset. The hue='target' parameter colors the data points based on the target class, helping us see how different species of irises relate to each other. Pairplots are useful for identifying patterns, such as whether certain features are good for distinguishing between classes.

Step 2: Train-Test Split

To evaluate the model, we need to split the data into training and testing sets.

# Train-test split

X = iris.data

y = iris.target

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

Explanation: We split the data into a training set and a testing set to train and evaluate the model, respectively. A 70-30 split is used in this case, meaning 70% of the data is used for training, and 30% for testing. The random\_state parameter ensures reproducibility.

Step 3: Building a Prediction Model

We'll use a Random Forest Classifier as the prediction model.

# Build a Random Forest Classifier

clf = RandomForestClassifier(n\_estimators=100, random\_state=42)

clf.fit(X\_train, y\_train)

Explanation: The Random Forest Classifier is a machine learning algorithm that can be used for classification tasks. We create the classifier, specify the number of trees (n\_estimators), and fit it to the training data.

Step 4: Making Predictions

After training the model, we use it to make predictions on the test data.

# Make predictions

y\_pred = clf.predict(X\_test)

Explanation: We use the trained model to predict the target class (species of iris) for the test data.

Step 5: Model Evaluation

In this step, we assess the model's performance using various metrics.

# Model evaluation

accuracy = accuracy\_score(y\_test, y\_pred)

print(f'Accuracy: {accuracy:.2f}')

# Classification report

print(classification\_report(y\_test, y\_pred))

# Confusion matrix

cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=iris.target\_names, yticklabels=iris.target\_names)

plt.xlabel('Predicted')

plt.ylabel('Actual')

plt.show()

Explanation:

We calculate the accuracy of the model, which is the proportion of correctly classified instances.



The classification report provides more detailed metrics like precision, recall, and F1-score for each class. It helps you understand the model's performance on individual classes.

The confusion matrix visualizes the number of true positives, true negatives, false positives, and false negatives. It's a useful tool for understanding where the model makes errors.

The sample outputs (accuracy, classification report, and confusion matrix) have already been explained in a previous response.

In a real-world analysis, you would likely perform more extensive data preprocessing, hyperparameter tuning, and consider other factors such as overfitting and cross-validation to ensure a robust model.