A Project Report On

“WATER QUALITY ANALYSIS USING MACHINE LEARNING”

Developed by

Abstract:

In an age characterized by data-driven insights and environmental stewardship, the "Water Quality Analysis Using Machine Learning" project stands as a pioneering initiative that converges machine learning and water quality assessment. Developed through rigorous data analysis and advanced machine learning models, this project reimagines the way we categorize and understand the quality of our water resources. By harnessing the potential of modern technology, it offers a robust and efficient means of categorizing water samples, safeguarding public health, and ensuring the sustainability of our natural water sources.

In this collaborative undertaking, the "Water Quality Analysis Using Machine Learning" not only addresses current challenges but also sets the course for future water quality management. It positions researchers, environmental agencies, and policymakers at the forefront of innovative practices, where machine learning becomes a strategic tool for enhancing water quality analysis.

The multifaceted methodology of this project encompasses data preprocessing, clustering, visualization, and real-time prediction, forming a comprehensive solution for water quality assessment. It enables the swift classification of water samples into distinct categories and extends the benefits of machine learning to real-time water quality prediction, offering an invaluable resource for those tasked with water quality management.

With the "Water Quality Analysis Using Machine Learning," we embark on a journey that propels environmental stewardship into a new era of data-driven efficiency and resource preservation. It serves as a testament to the transformative potential of machine learning in redefining how we manage and assess water quality, ensuring that our water resources remain safe, sustainable, and secure.

Introduction:

In an era where environmental preservation and data-driven decision-making are paramount, the "Water Quality Analysis Using Machine Learning" project emerges as a visionary response to the evolving challenges in water quality management. With a firm commitment to harness the potential of machine learning and data-driven solutions, this initiative seeks to redefine the assessment and categorization of water quality, ensuring the safeguarding of our natural water resources.

The project is rooted in the recognition of the increasing significance of water quality analysis and its role in environmental sustainability and public health. By leveraging advanced machine learning models, it aims to automate and enhance the categorization of water samples, offering an intelligent, data-backed approach to water quality assessment.

Purposes:

The primary purpose of the "Water Quality Analysis Using Machine Learning" project is to revolutionize water quality assessment through the power of machine learning. By automating the categorization of water samples, this project streamlines environmental monitoring and offers a data-driven solution for water quality management. The project also serves the purpose of creating a roadmap for stakeholders, aligning their efforts toward the shared goal of data-driven water quality assessment and sustainable resource management.

Overview:

This project revolves around the central objective of advancing water quality analysis through machine learning. In an age where the quality and safety of our water resources are of paramount concern, the project seeks to harness the capabilities of machine learning to provide accurate categorization and real-time prediction for water samples.

The implications of this project are far-reaching. By automating water sample categorization, it empowers environmental agencies, researchers, and policymakers to make informed decisions and enhance resource management. The "Water Quality Analysis Using Machine Learning" project epitomizes our dedication to data-driven environmental stewardship, setting the stage for a smarter and more sustainable approach to water quality management.

Objectives:

The "Water Quality Analysis Using Machine Learning" project is underpinned by a set of clear and well-defined objectives designed to address key challenges in water quality analysis, enhance understanding, and provide actionable insights. These objectives define the project's scope and serve as a roadmap for its successful execution.

1. **Automated Categorization:** The primary objective of this project is to automate the categorization of water samples using machine learning. Through the utilization of advanced algorithms, the system aims to accurately and efficiently classify water samples into distinct categories. This automation reduces the time and effort required for manual categorization and enhances the accuracy of water quality assessment.
2. **Data-Driven Insights:** The project seeks to uncover data-driven insights by exploring the correlations and patterns among water quality parameters. By leveraging visualization and clustering techniques, it aims to provide a comprehensive understanding of the interplay between different metrics and their impact on water quality.
3. **Data Integrity and Accessibility:** The project aims to ensure data integrity and accessibility by integrating with a structured database system. This objective involves designing a robust database schema to store and manage water quality data efficiently. The integration of a database enhances the reliability and accessibility of data for future analysis and reporting.
4. **Enhanced Visualizations:** To facilitate data interpretation, the project strives to enhance data visualizations, making complex information more accessible. This objective includes the development of visually engaging representations of water quality parameters, enabling stakeholders to grasp insights easily.
5. **Real-time Predictive Model:** An essential objective of the project is to develop a real-time predictive model for water quality assessment. This model enables the immediate classification of newly acquired water samples, providing rapid insights into their quality. It streamlines decision-making processes for researchers and environmental agencies.
6. **Adaptability and Future Applications:** The "Water Quality Analysis Using Machine Learning" project is designed to be adaptable for future enhancements and integration with other environmental monitoring systems. It seeks to provide a platform for expanding the scope of water quality assessment to address evolving needs.

These well-defined objectives guide the development and execution of the project, emphasizing the importance of automation, data-driven insights, and adaptability in the field of water quality analysis. Achieving these objectives will not only streamline the assessment of water quality but also contribute to more informed decision-making and sustainable water resource management.

Methodology:

The "Water Quality Analysis Using Machine Learning" project employs a structured methodology to develop and implement an intelligent system for water quality assessment. This methodology is designed to cover every crucial aspect of the project, ensuring its effectiveness. The system framework is illustrated in the figure below:

1. **Data Collection and Preprocessing:**

- Data Gathering: Open-source collection of water quality data from various sources.
- Data Preprocessing: Raw data undergoes preprocessing steps such as cleaning, missing value handling, and outlier removal. Data is formatted and transformed for model input.

2. **Exploratory Data Analysis (EDA):**

- Data Visualization: Visual exploration of the dataset through histograms, box plots, and correlation matrices to gain insights into the distribution and relationships between water quality parameters.

3. **Feature Engineering:**

- Feature Selection: Identifying and selecting relevant features that contribute to water quality assessment.
- Scaling and Transformation: Scaling features to ensure they have similar magnitudes and transforming them if necessary to meet model requirements.

4. **Machine Learning Model Development:**

- Model Selection: Choosing appropriate machine learning algorithms for classification tasks based on dataset characteristics.
- Model Training: Training machine learning models on the preprocessed data.
- Hyperparameter Tuning: Fine-tuning model hyperparameters for optimal performance.

5. **Model Evaluation:**

- Cross-Validation: Implementing cross-validation techniques to assess model generalization.
- Performance Metrics: Utilizing evaluation metrics such as accuracy, precision, recall, F1-score, and confusion matrices to gauge model performance.

6. **Real-time Prediction System:**

- Integration: Integrating the trained machine learning model into a real-time prediction system.
- Web Interface: Developing a user-friendly web interface for real-time water quality prediction.

7. **Database Integration:**

- Data Storage: Storing historical and real-time water quality data in a structured database for record-keeping and future analysis.

8. **Scalability and Future Enhancements:**

- Adaptability: Ensuring that the system is designed with scalability in mind to accommodate future enhancements and integration with other environmental monitoring systems.

This methodology blends data preprocessing, exploratory data analysis, machine learning model development, real-time implementation, and database integration to create a robust and intelligent system for water quality assessment. It facilitates data-driven insights, streamlines water quality prediction, and positions the project for future adaptability and enhancement.

K-Means Clustering:

K-Means is an unsupervised machine learning algorithm used for clustering, which is the process of grouping data points into distinct categories or clusters based on their similarity. In your water quality analysis project, K-Means is applied to categorize water samples into different groups based on the similarity of their feature attributes.

K-Means Clustering Process:

The K-Means algorithm follows these key steps:

- **Initialization:**
  Start by selecting the number of clusters (K) you want the data to be divided into. In your project, K is set to 3, as you aim to classify water samples into three distinct categories.
- **Centroid Initialization:**
  Randomly initialize the centroids for each cluster. Centroids represent the center points of the clusters and will be updated iteratively throughout the process.
- **Assignment:**
  Assign each data point (water sample) to the cluster whose centroid is closest to it. This assignment is based on a distance metric, typically Euclidean distance.
- **Update Centroids:**
  Recalculate the centroids for each cluster. The new centroid is determined as the mean of all data points assigned to that cluster.
- **Reassignment and Centroid Update Iteration:**
  Repeat the assignment and centroid update steps until convergence. In each iteration, data points are reassigned to the nearest centroid, and centroids are recalculated based on the updated assignments.
- **Convergence:**
  The algorithm converges when the centroids no longer change significantly between iterations, or when a predefined number of iterations is reached.
- **Output:**
  After convergence, the K-Means algorithm outputs the final clusters, each containing a set of water samples that share similar characteristics.

Application in Water Quality Analysis:

- K-Means clustering in your project is applied to categorize water samples into different groups (clusters) based on their feature attributes, such as pH, temperature, turbidity, dissolved oxygen, and conductivity. The algorithm identifies natural groupings within the water quality data, allowing for the automatic classification of samples into three categories (in your specific case).

Benefits:

- K-Means is an unsupervised learning method, making it suitable for exploratory data analysis and pattern recognition.
- It can uncover hidden patterns or trends in your water quality data that may not be apparent through other methods.
- The algorithm provides a data-driven approach to categorizing water samples, reducing the need for manual classification.

System Requirements:

Hardware Requirements

Processor : Intel Core i3-1005G1

Memory [RAM] : 4GB

Hard Disk Space : 100GB

Software Requirements

Operating System : Windows 7/8/10

Coding Language : Python

Coding software : Google Collabaratory, Pycharm

SOFTWARE ENVIRONMENT

Python

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation Python is dynamically-typed and garbage-collected it supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library Features in Python

There are many features in Python, some of which are discussed below as follows:

1. Easy to code: Python is a high-level programming language. Python is very easy to learn the language as compared to other languages like C, C#, Javascript, Java, etc. It is very easy to code in python language and anybody can learn python basics in a few hours or days. It is also a developer friendly language.

2. Free and Open Source: Python language is freely available at the official website and you can download it from the given download link below click on the Download Python keyword. Download Python Since it is open-source, this means that source code is also available to the public. So you can download it as, use it as well as share it.

3. Object-Oriented Language: One of the key features of python is Object-Oriented programming. Python supports object-oriented language and concepts of classes, objects encapsulation, etc.

4. GUI Programming Support: Graphical User interfaces can be made using a module such as PyQt5, PyQt4, wxPython, or Tk in python. PyQt5 is the most popular option for creating graphical apps with Python.

5. High-Level Language: Python is a high-level language. When we write programs in python, we do not need to remember the system architecture, nor do we need to manage the memory.

6. Extensible feature: Python is a Extensible language. We can write some Python code into C or C++ language and also we can compile that code in C/C++ language.

7. Python is Portable language: Python language is also a portable language. For example, if we have python code for windows and if we want to run this code on other platforms such as Linux, Unix, and Mac then we do not need to change it, we can run this code on any platform.

8. Python is Integrated language: Python is also an Integrated language because we can easily

integrated python with other languages like c, c++, etc.

9. Interpreted Language: Python is an Interpreted Language because Python code is executed line by line at a time. like other languages C, C++, Java, etc. there is no need to compile python code this makes it easier to debug our code. The source code of python is converted into an immediate form called bytecode.

10. Large Standard Library Python has a large standard library that provides a rich set of modules and functions so you do not have to write your own code for every single thing. There are many libraries present in python for such as regular expressions, unit-testing, web browsers, etc.

11. Dynamically Typed Language: Python is a dynamically-typed language. That means the type (for example- int, double, long, etc.) for a variable is decided at run time not in advance because of this feature we don't need to specify the type of variable.

12. Frontend and backend development: With a new project pyscript you can run and write python codes in html with the help of some simple tags <py-script>, <py-env>, etc. This will help you do frontend development work in python like javascript. Backend is the strong forte of python it's extensively used for this work cause of its framework like django and flask.

WINDOW 10

Is a computer operating system by Microsoft as part of its windows family of operation system. It was known as Threshold when it was being developed and announced at a press event on 30 September 2014. It came out for PCs on July 29, 2015. Beginning on that day, Windows 10 was available as a free upgrade for users running Windows 7 and Windows 8.1 for one year.

Windows 10 is designed to provide a common, "universal" user interface between desktop, laptop, and all-in-one PCs, tablet computer, smartphone, and embedded system such as its Xbox game console. Many of its features have been added based upon feedback from users, who are testing the software before it is released. It is being designed under the software as a service principle, in which the software will receive updates on a frequent basis throughout its life span. Windows 10 is the most powerful operation system that Microsoft has ever made, but it's also the most complex. This course will help you get to grips with its features and enable you to make the most of this features such as the Cortana voice assistant, Edge brower and multiple desktops. You will focus on the different menus including the start menu and the learn about the desktop environment and how it differs from previous versions of windows. You will learn how to customise windows 10 – for example by adding keyboard shortcuts – and change setting. You will focus on and learn how to make the most of the 'tiles' system and the different apps. We will look at how to pin items to the start menu and taskbar, how to use file explorer and how to manage your files and folders as well as search and view your drives and files. You will learn how to make the most of Cortana, Windows 10's built-in digital assistant and how it can help you gather important information, manage your schedule, send messages, ring your phone and more. We will also look at Edge, the default Microsoft browser, and practise commonActions like blocking pop-ups, managing favourites and sending web notes. Finally , we will focus on security issues such as protecting your passwords and also troubleshooting any issues.

Basic data of Windows 10:

• 64 bit and 32 bit version

• Kernel is based on "MinWin", introduced by Eric Traut in October 2007

• new graphic system

• improved language and handwriting recognition, useable over touch screen
• new user interface
• new program menus, with a recent list of the latest file and program functions used
• Windows XP mode (Windows 10 Professional or higher)

Google Colab

What is Google Colab?

Google Colab was developed by Google to provide free access to GPU's and TPU's to anyone who needs them to build a machine learning or deep learning model. Google Colab can be defined as an improved version of Jupyter Notebook.

What is Jupyter Notebook?

Jupyter Notebook is an application that allows editing and running Notebook documents through a web browser or an Integrated Development Environment (IDE). Instead of files, you will work with Notebooks.

What is a Notebook?

Programming Languages are an intermediate form between human-understandable language and machine understandable language. Every application is built using one of the many programming languages available. Maybe a person with a computer science background can understand, but not everyone can. Remember, as Software Developers, we develop applications for people with little computer science knowledge.

Consider you are creating a machine learning model to improve customer satisfaction for a local store, in that case you will have to explain how the model can do this task, and you can't just explain him with your code base. Most people facing this situation will prepare a separate presentation. Notebooks were created so that it is not necessary. Notebook documents can include executable lines of code along with text, images, figures, tables, graphs, equations, and much more graphical data. In simple words, Notebook documents are a way of creating human-readable executable documents.

Google Colab Features

Google Colab provides tons of exciting features that any modern IDE offers, and much more.

Some of the most exciting features are listed below.

- Interactive tutorials to learn machine learning and neural networks.
- Write and execute Python 3 code without having a local setup.
- Execute terminal commands from the Notebook.
- Import datasets from external sources such as Kaggle.
- Save your Notebooks to Google Drive.
- Import Notebooks from Google Drive.
- Free cloud service, GPUs and TPUs.
- Integrate with PyTorch, Tensor Flow, Open CV.
- Import or publish directly from/to GitHub.

Coding:

Code:

```
!pip install opendatasets
import opendatasets as od
import pandas
od.download("https://www.kaggle.com/datasets/shreyanshverma27/water-quality-testing")
```

Output:

```
Collecting opendatasets
  Downloading opendatasets-0.1.22-py3-none-any.whl (15 kB)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from opendatasets) (4.66.1)
Requirement already satisfied: kaggle in /usr/local/lib/python3.10/dist-packages (from opendatasets) (1.5.16)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from opendatasets) (8.1.7)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (1.16.0)
Requirement already satisfied: certifi in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2023.7.22)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.31.0)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (8.0.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (2.0.6)
Requirement already satisfied: bleach in /usr/local/lib/python3.10/dist-packages (from kaggle->opendatasets) (6.1.0)
Requirement already satisfied: webencodings in /usr/local/lib/python3.10/dist-packages (from bleach->kaggle->opendatasets) (0.5.1)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.10/dist-packages (from python-slugify->kaggle->opendatasets) (1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle->opendatasets) (3.3.0)
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->kaggle->opendatasets) (3.4)
Installing collected packages: opendatasets
Successfully installed opendatasets-0.1.22
Please provide your Kaggle credentials to download this dataset. Learn more: http://bit.ly/kaggle-creds
Your Kaggle username: dharanigurumoorthy
Your Kaggle Key: ··········
Downloading water-quality-testing.zip to ./water-quality-testing
100%|██████████| 3.63k/3.63k [00:00<00:00, 6.90MB/s]
```

Import the required packages

Code:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

Code:

```
#load the data into dataframe
df = pd.read_csv("/content/water-quality-testing/Water Quality Testing.csv",index_col=0)
df.head()
```

Output:

Sample ID	pH	Temperature (°C)	Turbidity (NTU)	Dissolved Oxygen (mg/L)	Conductivity (µS/cm)
1	7.25	23.1	4.5	7.8	342
2	7.11	22.3	5.1	6.2	335
3	7.03	21.5	3.9	8.3	356
4	7.38	22.9	3.2	9.5	327
5	7.45	20.7	3.8	8.1	352

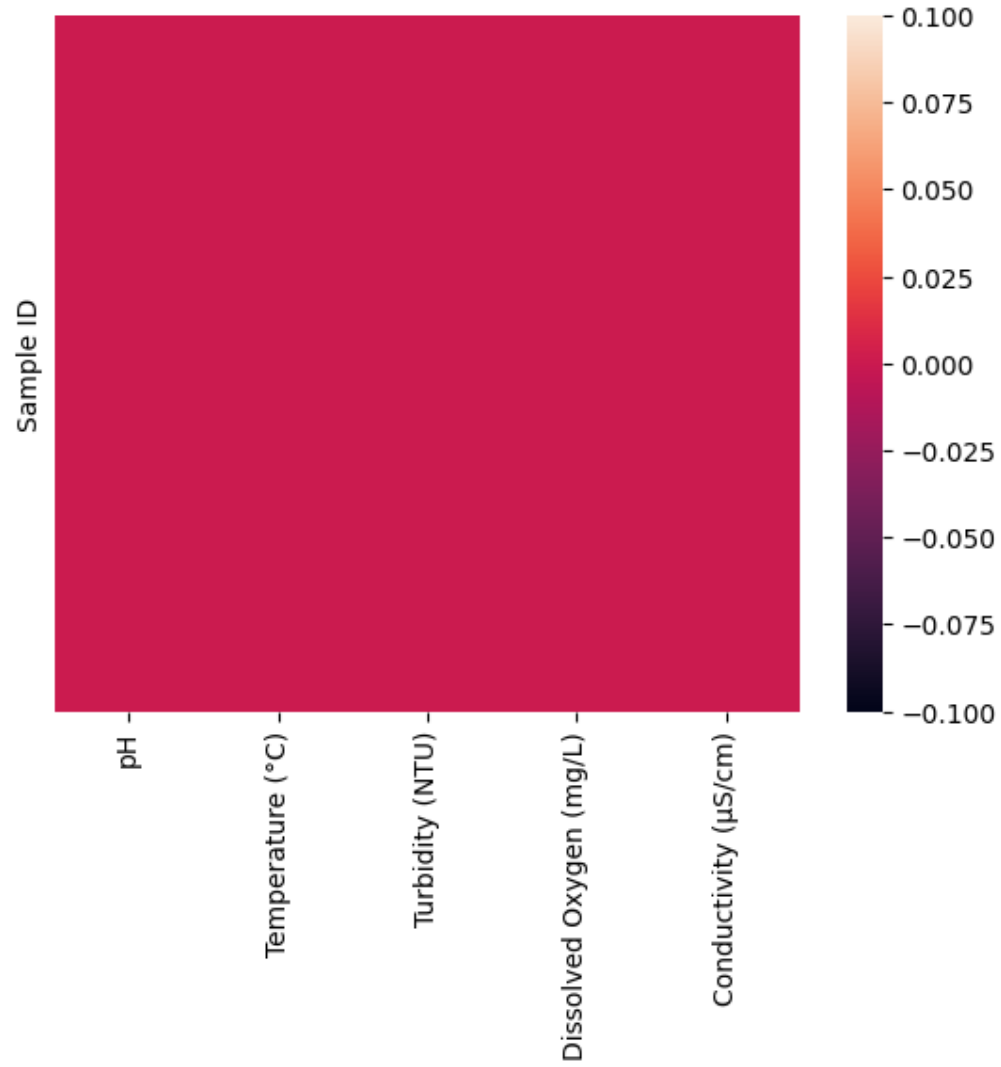
Check for the Null value

Code:

```
sns.heatmap(df.isnull(),yticklabels=False)
```

Output:

```
<Axes: ylabel='Sample ID'>
```



Code:

df.isnull().sum()

Output:

```
pH                          0
Temperature (°C)            0
Turbidity (NTU)             0
Dissolved Oxygen (mg/L)     0
Conductivity (µS/cm)        0
dtype: int64
```

Code:

#Check for the Ph distribution is the water qaulity test

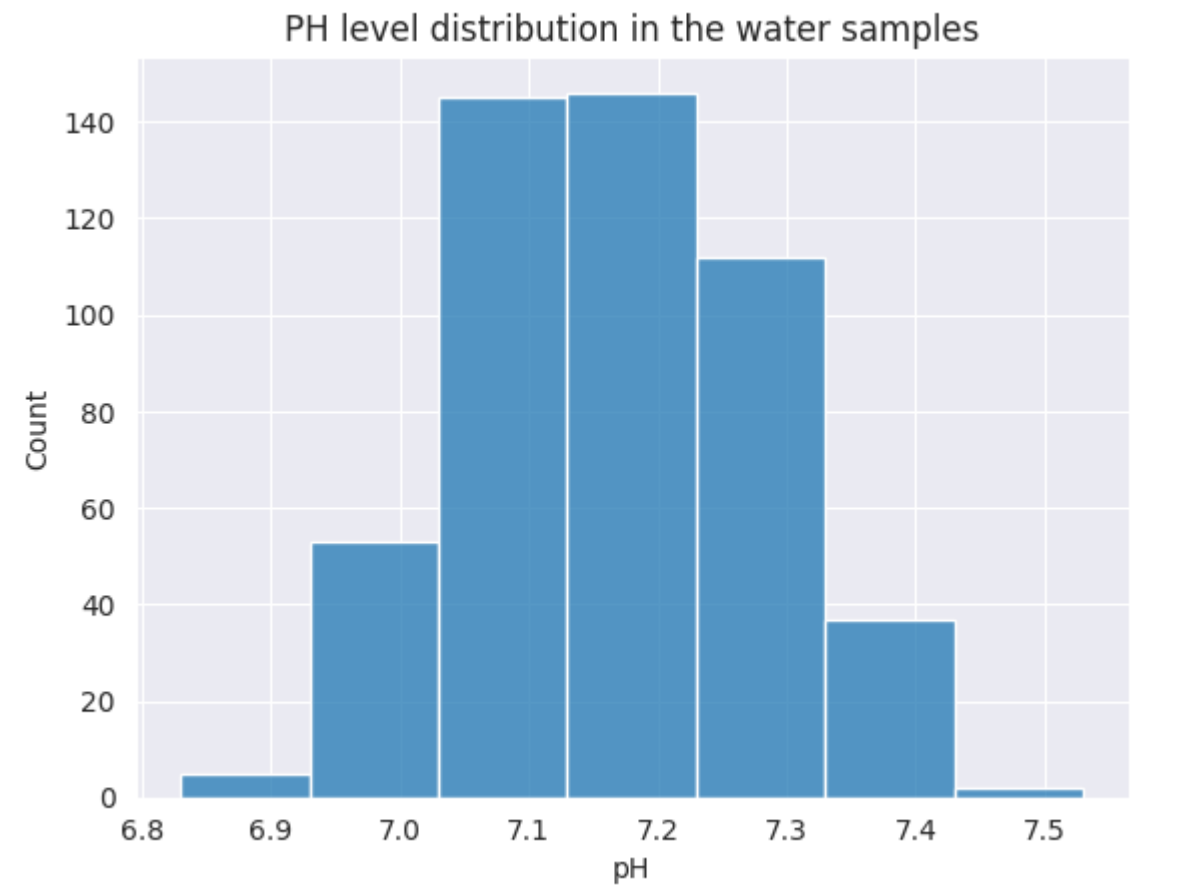
sns.set_style("darkgrid")

sns.histplot(data=df,x="pH",binwidth=0.1)

plt.title("PH level distribution in the water samples")

plt.show()

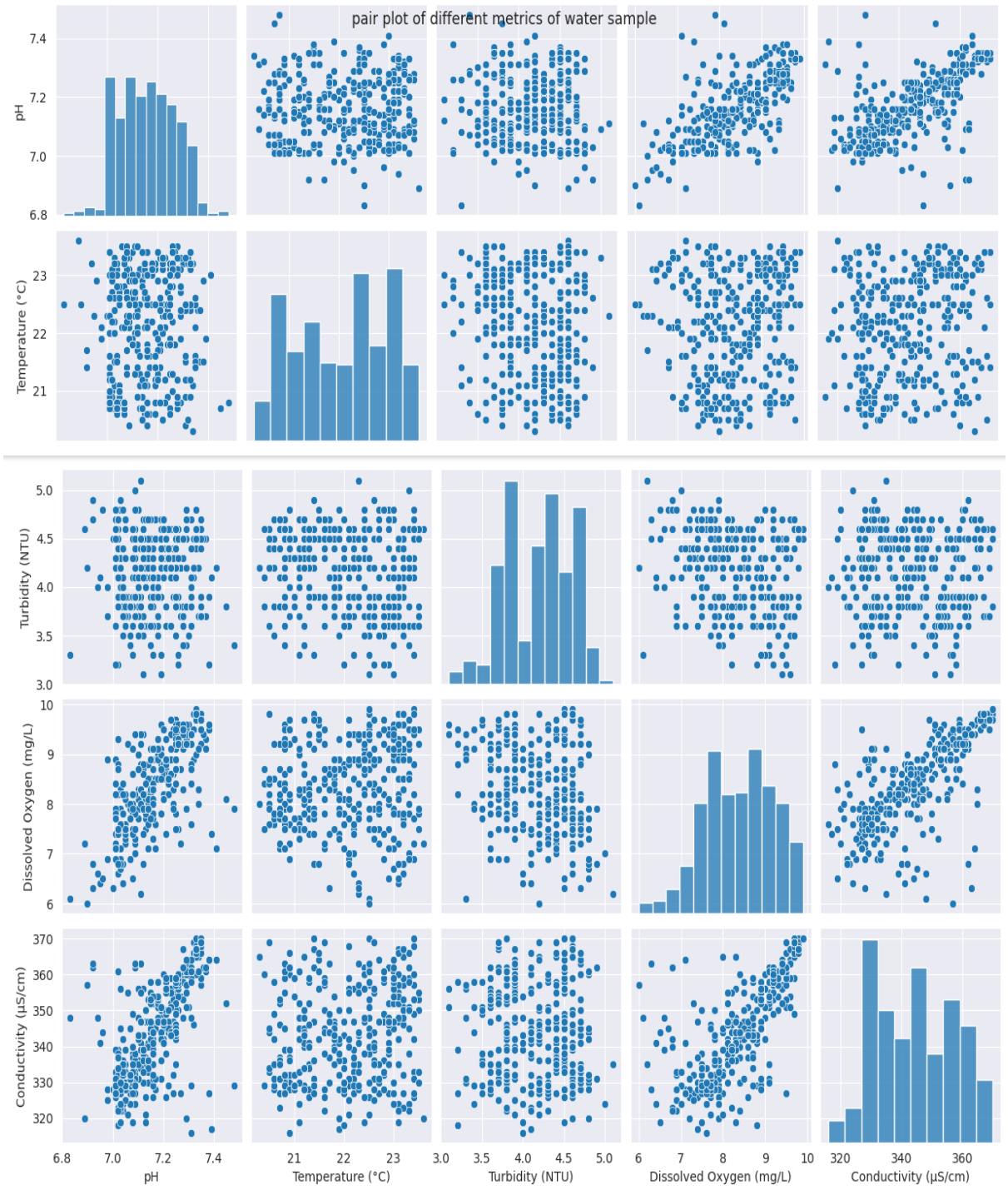
Output:



Code:

```
sns.pairplot(df)
plt.suptitle("pair plot of different metrics of water sample")
plt.show()
```

Output:



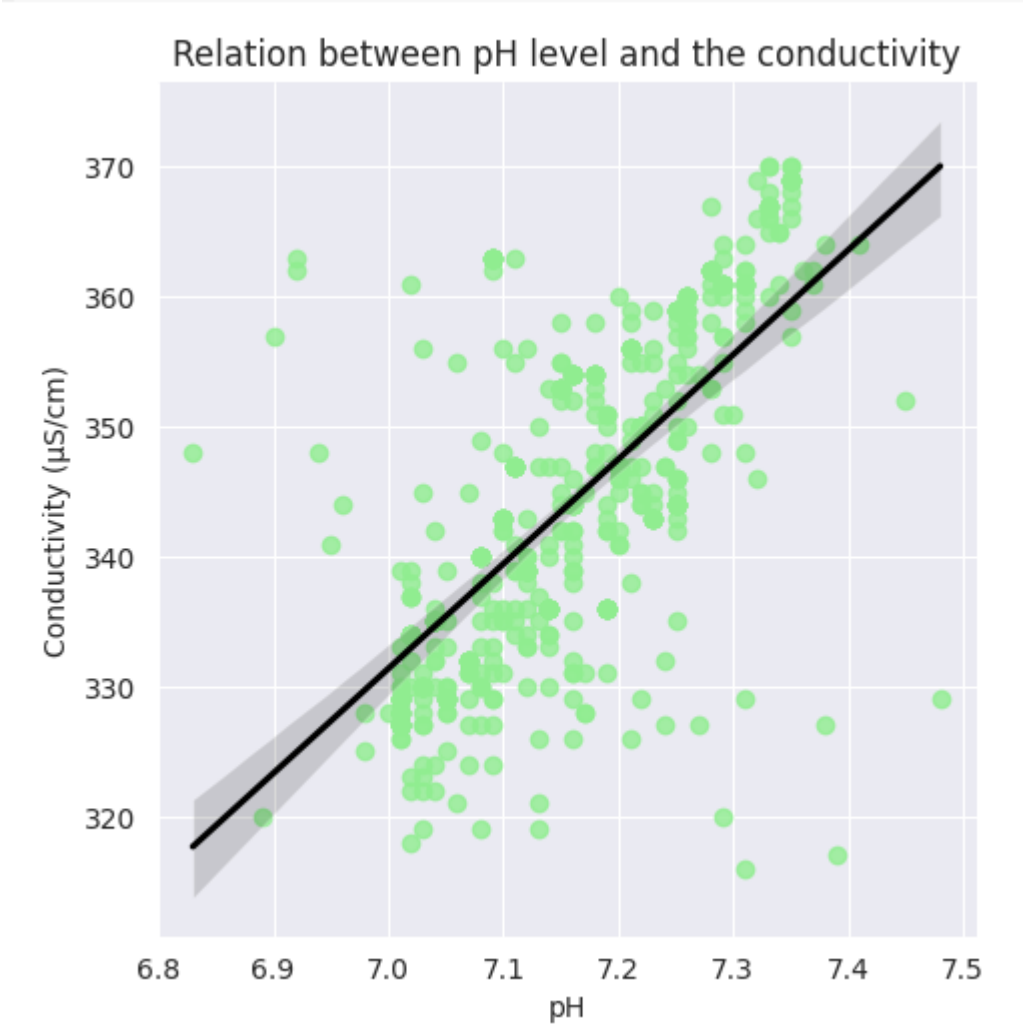
Code:

```
sns.lmplot(data=df,x="pH",y="Conductivity
(µS/cm)",scatter_kws={'color':'lightgreen'},line_kws={'color':'black'})

plt.title("Relation between pH level and the conductivity")

plt.show()
```

Output:



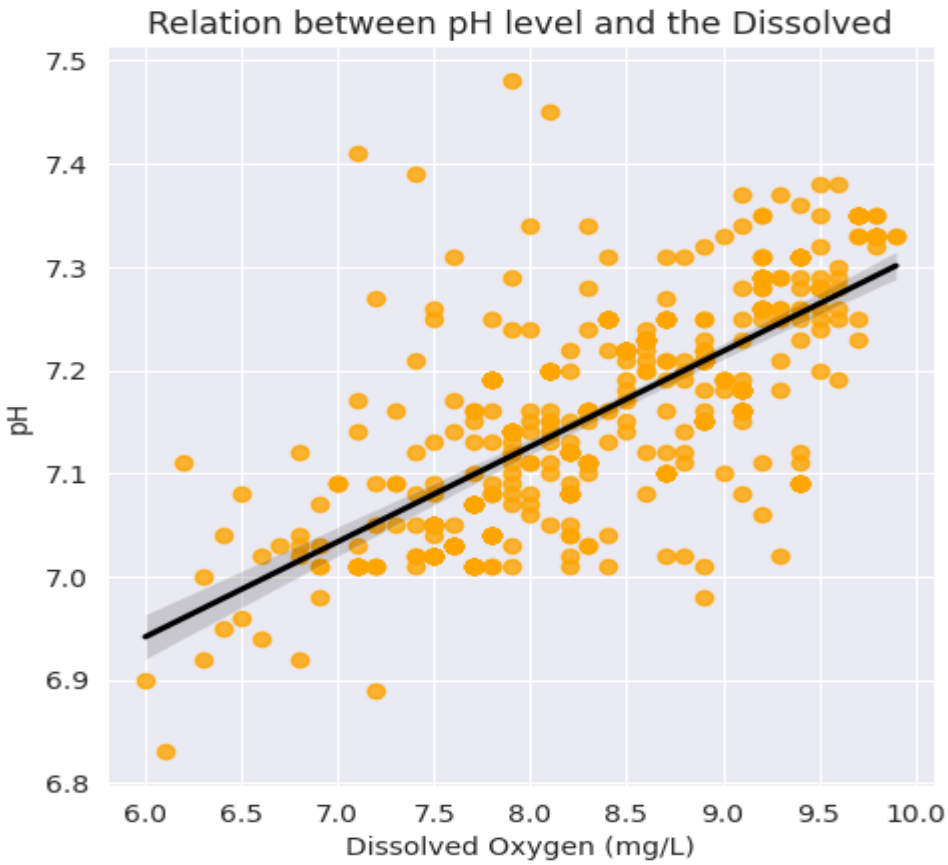
Code:

```
sns.lmplot(data=df,x="Dissolved Oxygen
(mg/L)",y="pH",scatter_kws={'color':'orange'},line_kws={'color':'black'})

plt.title("Relation between pH level and the Dissolved")

plt.show()
```

Output:



Using the Kmeans Model to classify the water sample into 3 category

Code:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(df)
StandardScaler(copy=True,with_mean=True,with_std=True)
sampled_scaler = scaler.transform(df)
```

Code:

```
from sklearn.cluster import KMeans
model = KMeans(n_clusters=3,random_state=2023)
cluster = model.fit_predict(sampled_scaler)
```

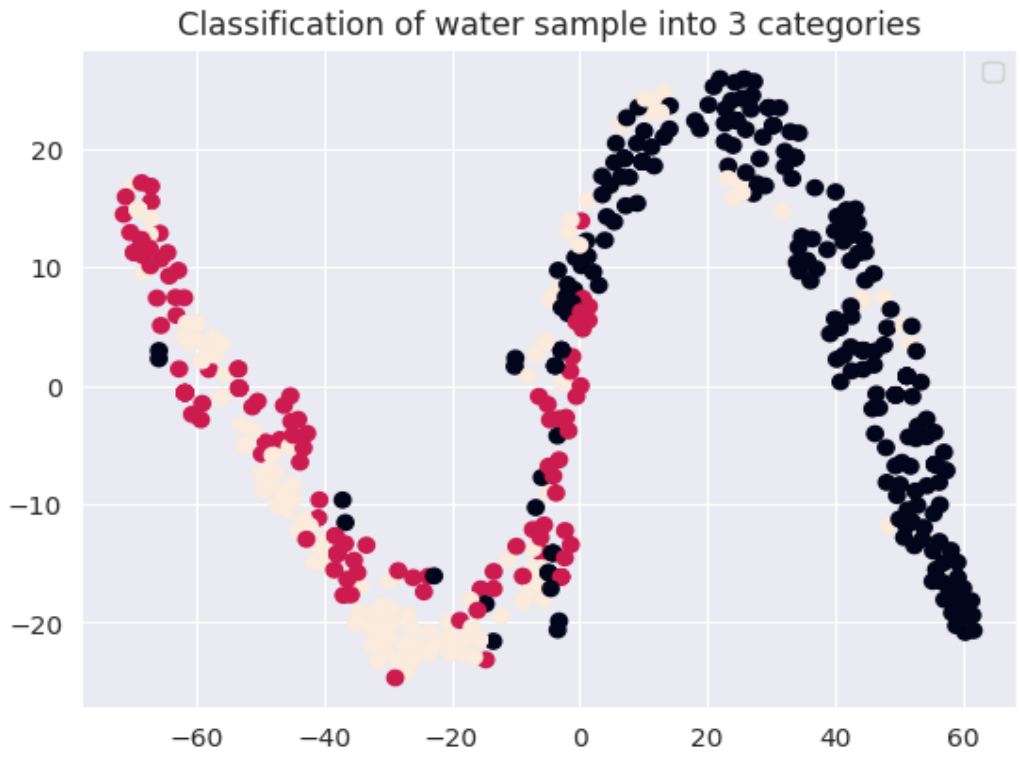
Code:

```
level = pd.Series(cluster)
```

Code:

```
from sklearn.manifold import TSNE
model = TSNE()
transformed_model = model.fit_transform(df)
xs = transformed_model[:,0]
ys =transformed_model[:,1]
plt.scatter(xs,ys,c=level)
plt.title("Classification of water sample into 3 categories")
plt.legend()
plt.show()
```

Output:

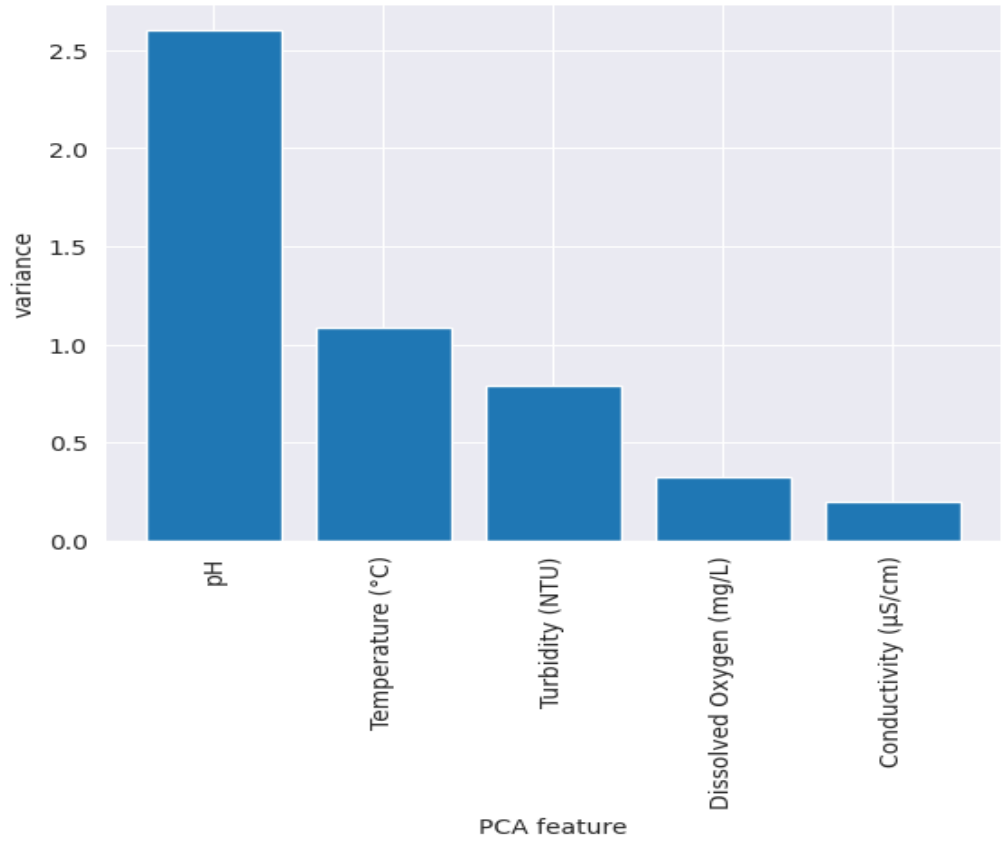


Code:

```
# Perform the necessary imports
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
import matplotlib.pyplot as plt
# Create scaler: scaler
```

```
scaler = StandardScaler()
# Create a PCA instance: pca
pca = PCA()
# Create pipeline: pipeline
pipeline = make_pipeline(scaler,pca)
# Fit the pipeline to 'samples'
pipeline.fit(df)
# Plot the explained variances
features = range(pca.n_components_)
plt.bar(features, pca.explained_variance_)
plt.xlabel('PCA feature')
plt.ylabel('variance')
plt.xticks(features,df.columns)
plt.xticks(rotation=90)
plt.show()
```

Output:



Code:

```
df['classification'] = cluster
df.head()
```

Output:

Sample ID	pH	Temperature (°C)	Turbidity (NTU)	Dissolved Oxygen (mg/L)	Conductivity (µS/cm)	classification
1	7.25	23.1	4.5	7.8	342	1
2	7.11	22.3	5.1	6.2	335	0
3	7.03	21.5	3.9	8.3	356	0
4	7.38	22.9	3.2	9.5	327	2
5	7.45	20.7	3.8	8.1	352	1

Lets rename the sample categories based on the results

Code:

```
df['classification'] = df['classification'].fillna(0)
df['classification'] = df['classification'].astype("int")
df['classification'] = df['classification'].map({0:'category1',1:'category2',2:'category3'})
df['classification'].value_counts()
```

Output:

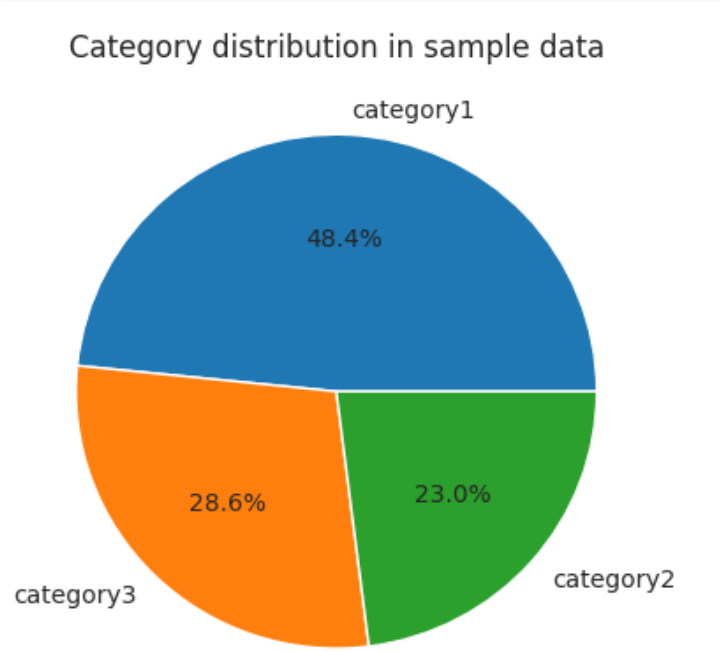
```
category1    242
category3    143
category2    115
Name: classification, dtype: int64
```

check the distribution of category in the samples

Code:

```
category = df['classification'].value_counts().reset_index(name='counts')
plt.pie(x=category['counts'],labels=category['index'],autopct='%1.1f%%')
plt.title("Category distribution in sample data")
plt.show()
```

Output:



Code:

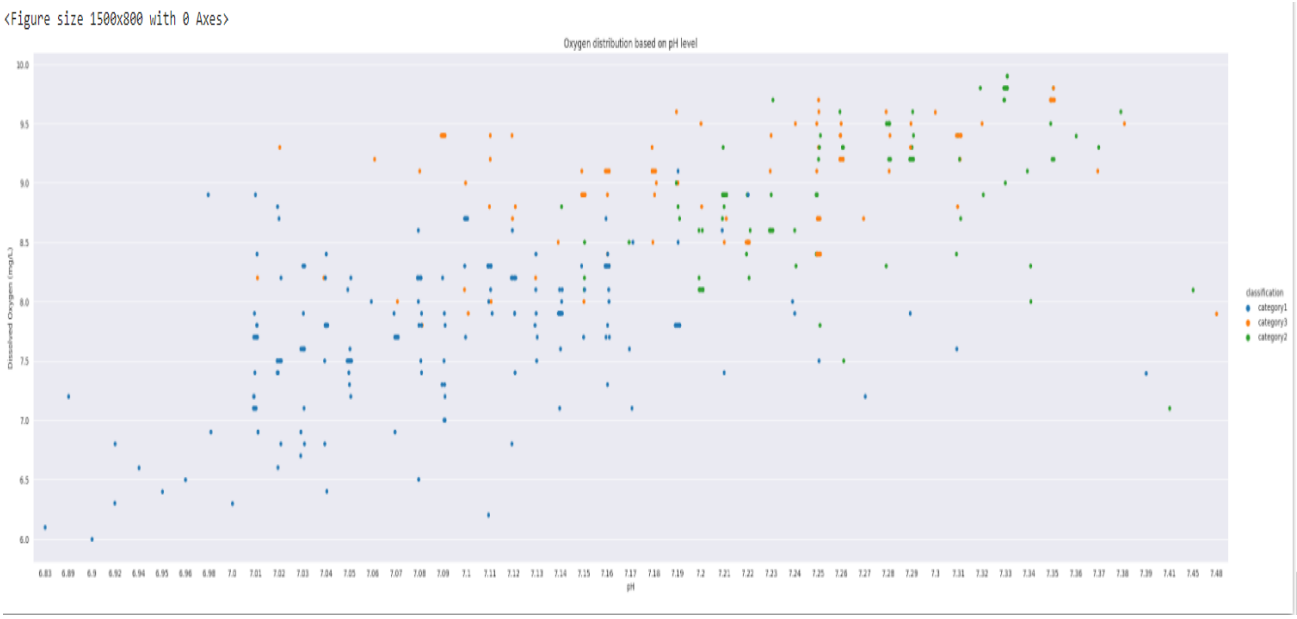
```
fig = plt.figure(figsize=(15,8))

sns.catplot(data=df,x="pH",y="Dissolved Oxygen (mg/L)",hue="classification",height=8,aspect=3.7)

plt.title("Oxygen distribution based on pH level")

plt.show()
```

Output:

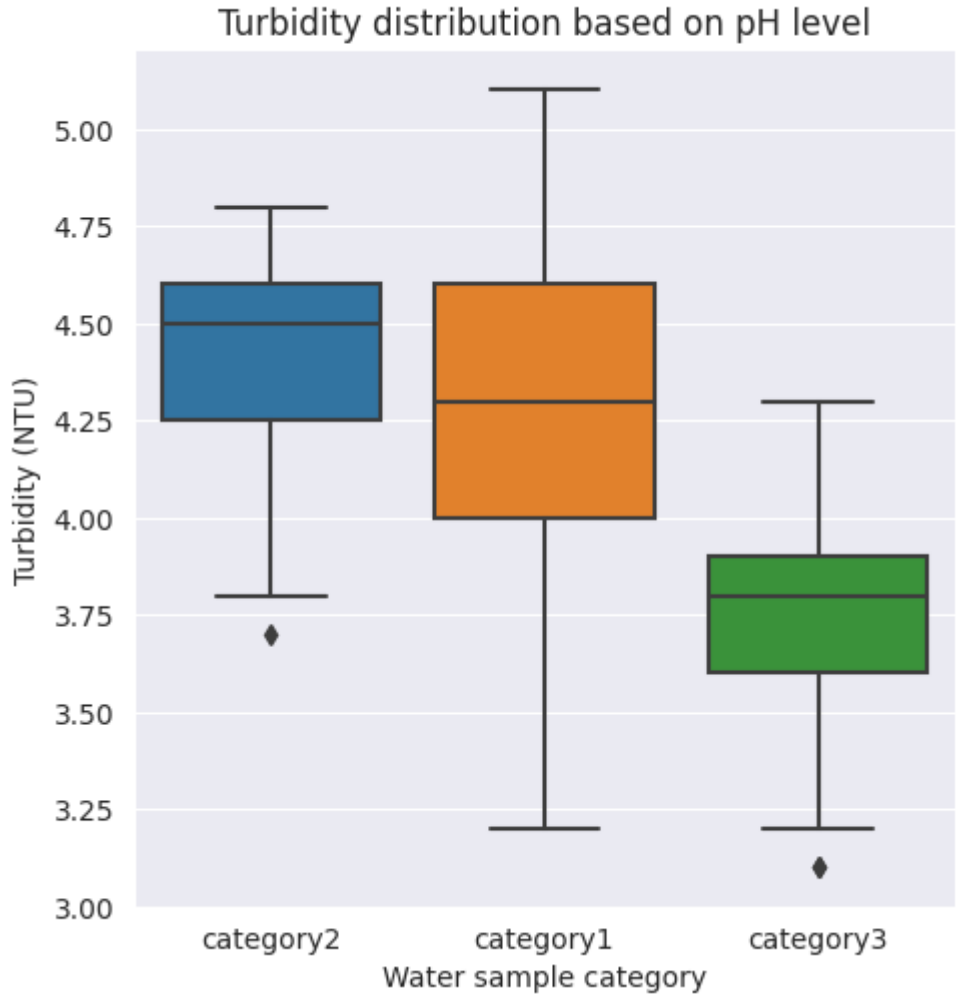


Code:

```
fig = plt.figure(figsize=(15,8))
sns.catplot(data=df,y="Turbidity (NTU)",x="classification",kind='box')
plt.title("Turbidity distribution based on pH level")
plt.xlabel("Water sample category")
plt.show()
```

Output:

```
<Figure size 1500x800 with 0 Axes>
```

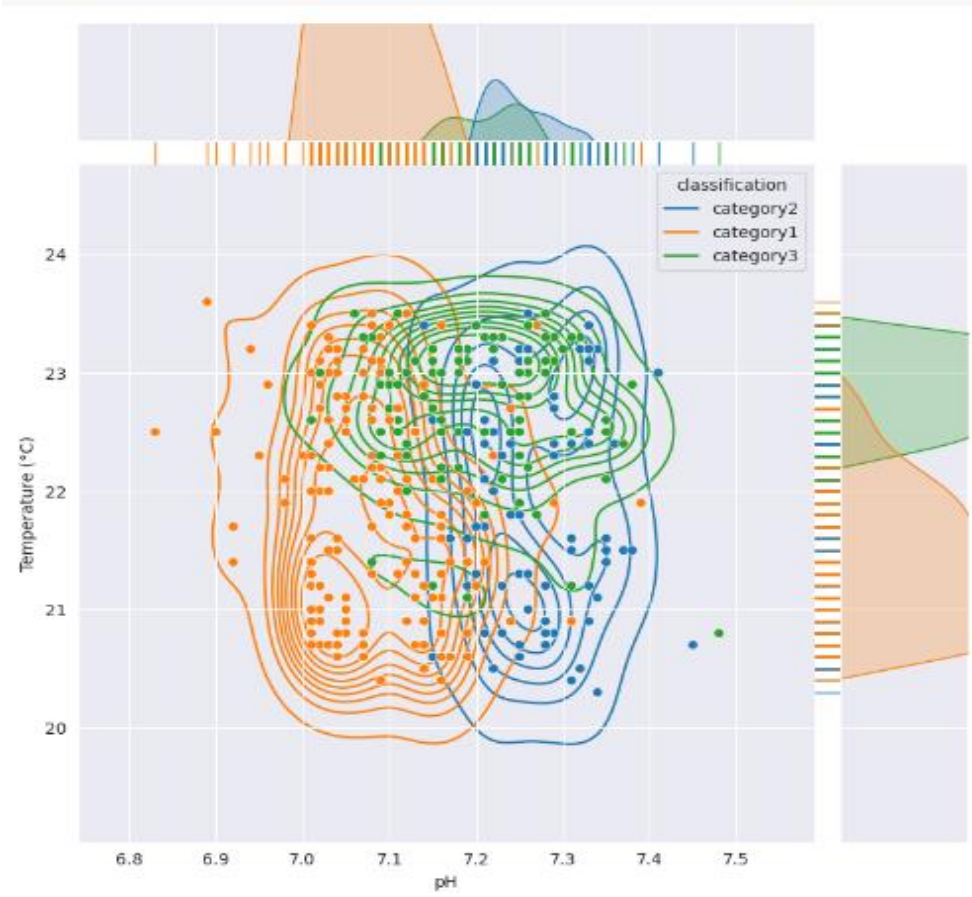


Distribution of 2 important features while diciding the sample classification

Code:

```
g = sns.jointplot(data=df,x="pH",y="Temperature (°C)",hue="classification",height=8)
g.plot_joint(sns.kdeplot,color='y',zorder=0)
g.plot_marginals(sns.rugplot,color='r',height=-0.2,clip_on=False)
plt.show()
```

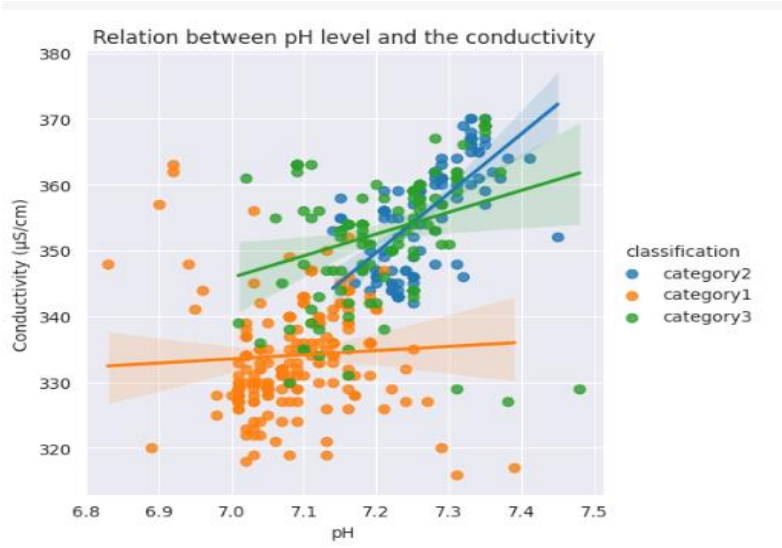
Output:



Code:

```
sns.lmplot(data=df,x="pH",y="Conductivity (µS/cm)",hue='classification')
plt.title("Relation between pH level and the conductivity")
plt.show()
```

Output:



Code:

```
df.groupby("classification")['Dissolved Oxygen (mg/L)'].mean()
```

Output:

```
classification
category1    7.741736
category2    8.964348
category3    8.997902
Name: Dissolved Oxygen (mg/L), dtype: float64
```

classification report

Code:

```
from sklearn.metrics import classification_report
# Assuming you have the true labels (y_true) and predicted labels (y_pred)
y_true = df['classification']  # Replace with the actual true labels in your dataset
y_pred = df['classification']  # Replace with the predicted labels from your KMeans model
# Generate the classification report
report = classification_report(y_true, y_pred)
# Print the classification report
print(report)
```

Output:

```
                precision    recall  f1-score   support

     category1       1.00      1.00      1.00       242
     category2       1.00      1.00      1.00       115
     category3       1.00      1.00      1.00       143

      accuracy                           1.00       500
     macro avg       1.00      1.00      1.00       500
  weighted avg       1.00      1.00      1.00       500
```

Real Time Prediction Using Water Parameter

Code:

```
# Create a new instance of KMeans for prediction
kmeans_for_prediction = KMeans(n_clusters=3, random_state=2023)
# Fit the KMeans model to the scaled data
kmeans_for_prediction.fit(sampled_scaler)
# Predict the category for the new sample using the KMeans model
new_sample_category = kmeans_for_prediction.predict(new_sample_scaled)
```

```
# Map the category to its name (e.g., 'category1', 'category2', 'category3')
category_name = {0: 'category1', 1: 'category2', 2: 'category3'}
predicted_category = category_name[new_sample_category[0]]
print(f"The new sample belongs to: {predicted_category}")
```

Output:

```
The new sample belongs to: category2
```

Conclusion:

The "Water Quality Analysis Using Machine Learning" project represents a significant milestone in the domain of environmental data analysis. By leveraging machine learning techniques, particularly the K-Means clustering algorithm, this project successfully classifies water samples into distinct categories based on their quality attributes.

The primary objectives of the project were achieved effectively. K-Means clustering was employed to categorize water samples into three distinct categories, providing valuable insights into the dataset's structure. This clustering approach streamlines the classification process, reducing the need for manual assessment and enhancing the speed and efficiency of water quality analysis.

The K-Means algorithm has demonstrated its capability to uncover inherent patterns and relationships within the water quality data. This empowers decision-makers to make more informed choices regarding water quality management, allocation of resources, and potential problem detection.

Moreover, visualizations and statistical analyses, such as histograms, pair plots, and regression plots, were employed to provide a comprehensive understanding of the dataset's distribution and relationships between water quality attributes.

The results showcase the potential for practical applications of machine learning in the domain of environmental science and water quality assessment. As organizations and institutions aim to monitor and manage water quality efficiently, this project serves as a notable example of harnessing data-driven approaches to gain insights and streamline decision-making processes.

In conclusion, the "Water Quality Analysis Using Machine Learning" project contributes to the ongoing efforts to manage and safeguard water resources effectively. The successful application of K-Means clustering and data analysis techniques signifies the project's significance in enhancing water quality assessment, aiding in the identification of potential issues, and supporting sustainable environmental practices. It sets the stage for further advancements in the field of environmental data analysis.