411521104033-Elakya.R

**WATER QUALITY ANALYSIS**

**DATA ANALYTICS WITH COGNOS-GROUP1**

Phase-2



**INTRODUCTION**

**Water quality analysis using Python:**[Water quality analysis is a major area of research in machine learning](https://thecleverprogrammer.com/2021/08/19/water-quality-analysis/). [The goal is to understand all the factors that affect water potability and train a machine-learning model that can classify whether a specific water sample is safe or unfit for consumption](https://thecleverprogrammer.com/2021/08/19/water-quality-analysis/).

In Python, you can start the water quality analysis task by importing the necessary libraries and the dataset. [The dataset contains data on all of the major factors that affect the potability of water](https://thecleverprogrammer.com/2021/08/19/water-quality-analysis/). [You can enhance a data module by creating calculations, defining filters and navigation paths, and more](https://thecleverprogrammer.com/2021/08/19/water-quality-analysis/).

[**Data Analytics with Cognos:** IBM Cognos Analytics is a web-based, self-service data modeling tool](https://www.bing.com/aclk?ld=e8miznHhYfwZObJGpxBd7ocDVUCUyTppkcU0FeilJzPTzPfUrtyKkVMCJR5FZXSZZpWpn-5z495qyK0aot1xA-vFVvdS1SuITbahEU6vu76oGbeI8sqCeryGPK2SoqAtXBnCQFoSf5R5rXvehK0y34cED_blaZumyFAU31s6eFB-kZSNX2&u=&rlid=dded8878d25f1a6c4529ed5e7b4a3b2b). [It provides AI-powered automation and insights that enable everyone in your organization to unlock the full potential of your data](https://www.ibm.com/products/cognos-analytics). [It allows users to connect, verify, and combine data from many sources, including relational databases, Hadoop-based technologies, Microsoft Excel spreadsheets, text files, and more](https://www.bing.com/aclk?ld=e8miznHhYfwZObJGpxBd7ocDVUCUyTppkcU0FeilJzPTzPfUrtyKkVMCJR5FZXSZZpWpn-5z495qyK0aot1xA-vFVvdS1SuITbahEU6vu76oGbeI8sqCeryGPK2SoqAtXBnCQFoSf5R5rXvehK0y34cED_blaZumyFAU31s6eFB-kZSNX2&u=&rlid=dded8878d25f1a6c4529ed5e7b4a3b2b).

**Integration of Python and Cognos:** While Python is used for analysis, Cognos Analytics can visualize the results. After performing the water quality analysis in Python, you can export the results as a CSV file or any other compatible format. This exported file can then be imported into Cognos Analytics for further exploration and visualization. This way, you can leverage the strengths of both Python for analysis and Cognos for visualization to get a comprehensive understanding of water quality.

Water quality analysis using Python typically involves the following steps:

1. Data collection: Collect water quality data from various sources, such as sensors, laboratory tests, and historical records.
2. Data cleaning and preparation: Clean and prepare the data for analysis. This may involve removing outliers, handling missing values, and converting data to a consistent format.
3. Exploratory data analysis (EDA): Perform EDA to understand the data and identify any patterns or trends. This may involve creating visualizations, such as histograms, line charts, and scatter plots.
4. Machine learning modeling (optional): Train a machine learning model to predict water quality parameters. This can be done using a variety of machine learning algorithms, such as linear regression, decision trees, and random forests.
5. Model evaluation: Evaluate the performance of the machine learning model on a held-out test set.
6. Model deployment: Once the machine learning model has been evaluated and deemed to be performing well, it can be deployed to production. This may involve creating a web service or integrating the model into a BI platform, such as Cognos.

Water quality analysis using Python with Cognos

Once a Python model has been trained and deployed, it can be integrated into Cognos to create interactive dashboards and reports to visualize water quality data and predictions. This can be done using the Cognos Python API.

The Cognos Python API allows developers to interact with Cognos objects, such as reports, dashboards, and data sets. This can be used to create custom data pipelines, automate tasks, and develop new features for Cognos.

**EXPLORATORY DATA ANALYSIS**

Exploratory data analysis (EDA) is a statistical approach to analyzing data sets to summarize their main characteristics, often using statistical graphics and other data visualization methods. EDA is primarily used to see what data can reveal beyond the formal modeling or hypothesis testing task and provides a provides a better understanding of data set variables and the relationships between them.

EDA is an important part of the data analysis process because it can help to:

* Identify patterns and trends in the data
* Detect outliers and anomalies
* Identify relationships between variables
* Formulate hypotheses about the data
* Determine the appropriate statistical methods to use for further analysis

EDA can be performed using a variety of tools and techniques, but some of the most common methods include:

* Univariate analysis: This involves analyzing individual variables in isolation. Common univariate analysis methods include histograms, boxplots, and cumulative distribution functions.
* Bivariate analysis: This involves analyzing two variables together. Common bivariate analysis methods include scatter plots, correlation coefficients, and regression analysis.
* Multivariate analysis: This involves analyzing three or more variables together. Common multivariate analysis methods include principal component analysis, cluster analysis, and machine learning algorithms.

**MODEL TRAINING**

Model training in water quality analysis often involves the use of machine learning algorithms to predict water quality parameters. [These parameters can include temperature, dissolved oxygen (DO), pH, conductivity, biochemical oxygen demand (BOD), nitrates (NO3), and faecal and total coli forms (TC)](https://www.jetir.org/papers/JETIR1811966.pdf).

In a study conducted in Tamil Nadu, India, five types of classification algorithms were used: Navie Bayes (NB), Decision Tree (DT), K-Nearest Neighbor (KNN), Support Vector Machine (SVM), and Random Forest (RF). [These algorithms were used to predict the water quality class based on the aforementioned parameters1](https://www.jetir.org/papers/JETIR1811966.pdf). [The study found that the Random Forest classifier achieved improved results compared to the other classifiers](https://www.jetir.org/papers/JETIR1811966.pdf).

**DECISION TREE**

Decision trees are a type of machine learning algorithm that can be used for water quality analysis. Decision trees work by constructing a tree-like structure, where each node in the tree represents a decision and each leaf node represents a prediction.

To train a decision tree for water quality analysis, the algorithm is fed a dataset of water quality data, along with the corresponding target values. The algorithm then learns to split the data into different branches based on the values of the input variables. At each split, the algorithm chooses the split that maximizes the purity of the resulting branches.

Once the decision tree has been trained, it can be used to predict water quality parameters for new data samples. To do this, the algorithm simply starts at the root node of the tree and follows the branches down until it reaches a leaf node. The prediction of the leaf node is then the predicted water quality parameter for the new data sample.

Decision trees are a popular choice for water quality analysis because they are relatively easy to train and interpret. They can also be used to predict a wide range of water quality parameters, such as pH, dissolved oxygen, nutrient levels, and bacterial contamination.

**HYPER PARAMETER TUNING**

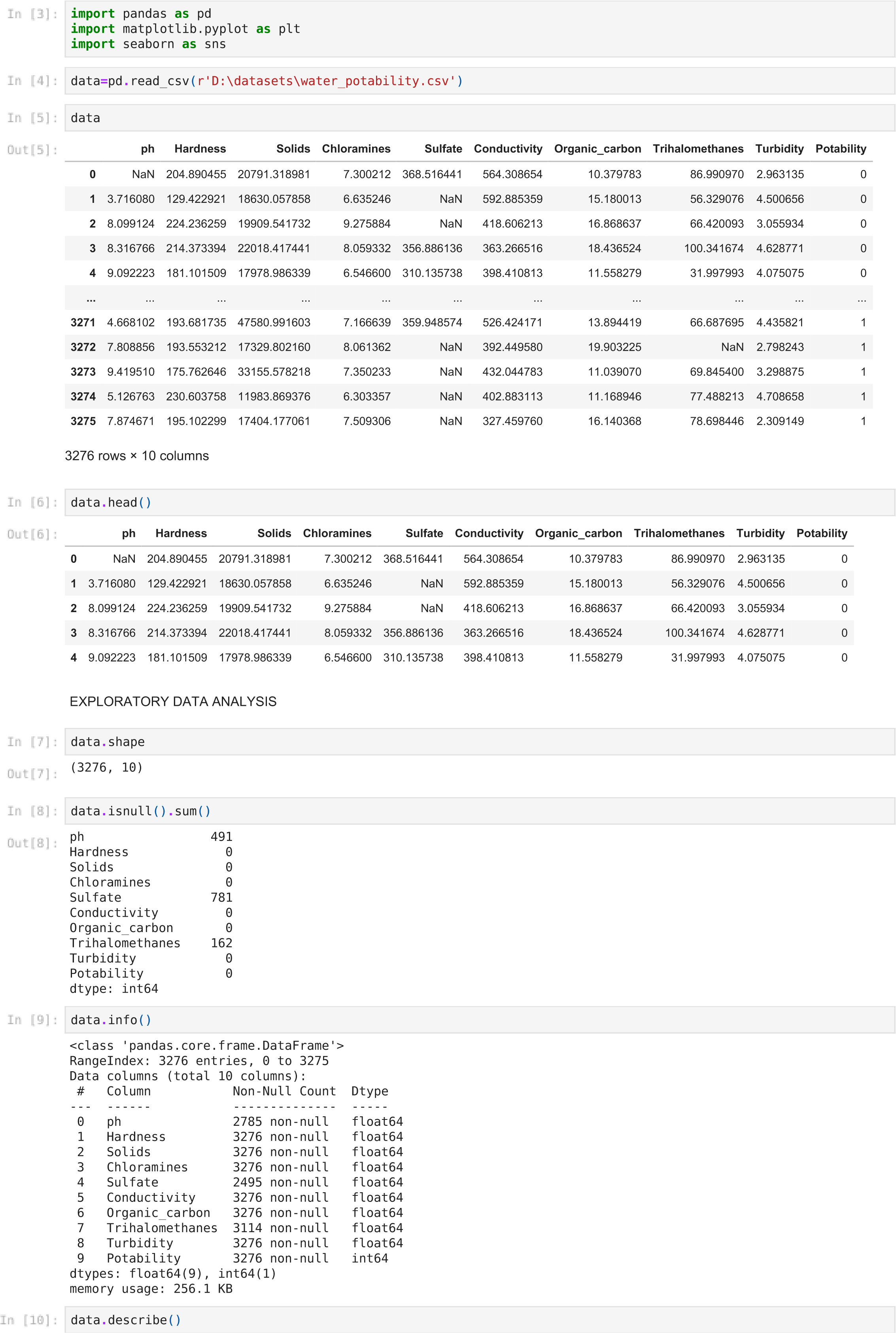
Hyperparameter tuning is the process of finding the optimal values for the hyperparameters of a machine learning model. Hyperparameters are parameters that control the learning process of a machine learning model, but are not directly learned from the data.

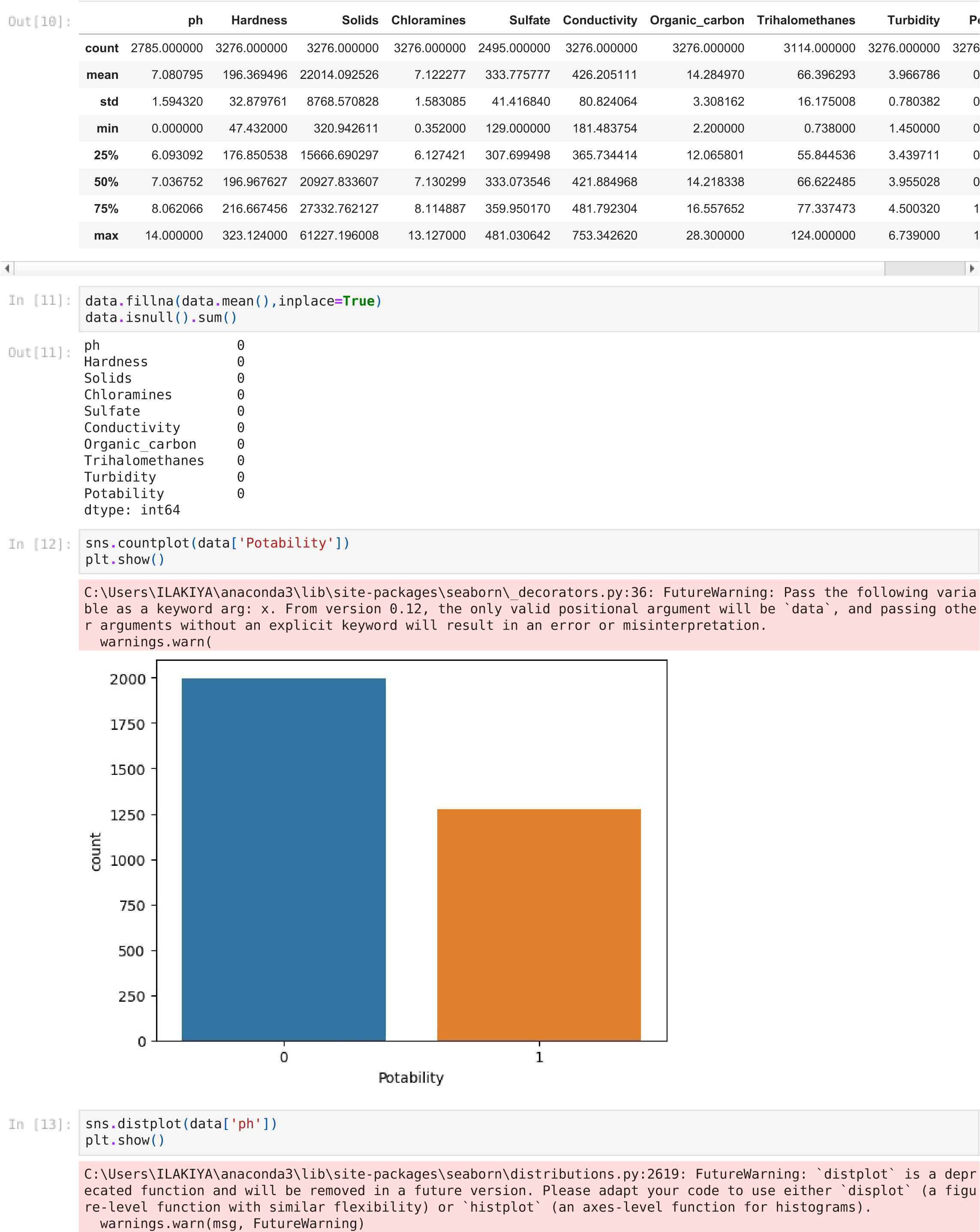
Hyperparameter tuning is important for water quality analysis because it can help to improve the performance of machine learning models. For example, hyperparameter tuning can be used to find the optimal values for the following parameters:

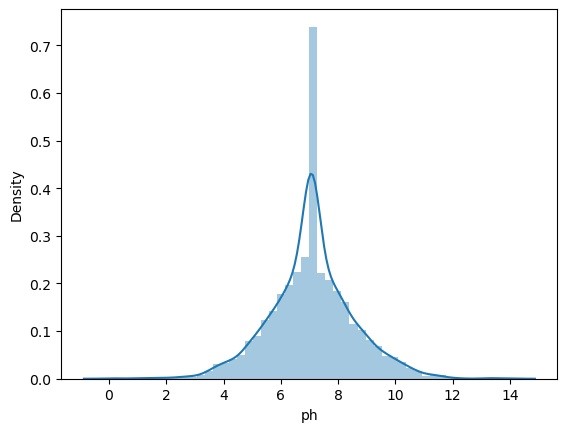
* Learning rate: This parameter controls the speed at which the model learns from the data.
* Number of trees: This parameter controls the number of trees in a decision tree forest.
* Depth of trees: This parameter controls the maximum depth of the trees in a decision tree forest.
* Regularization parameter: This parameter helps to prevent overfitting.

There are a variety of methods that can be used for hyperparameter tuning.

* Grid search
* Random search
* Bayesian optimization

data set.





data

**.**

hist

(

figsize

**=**

(

14

,

14

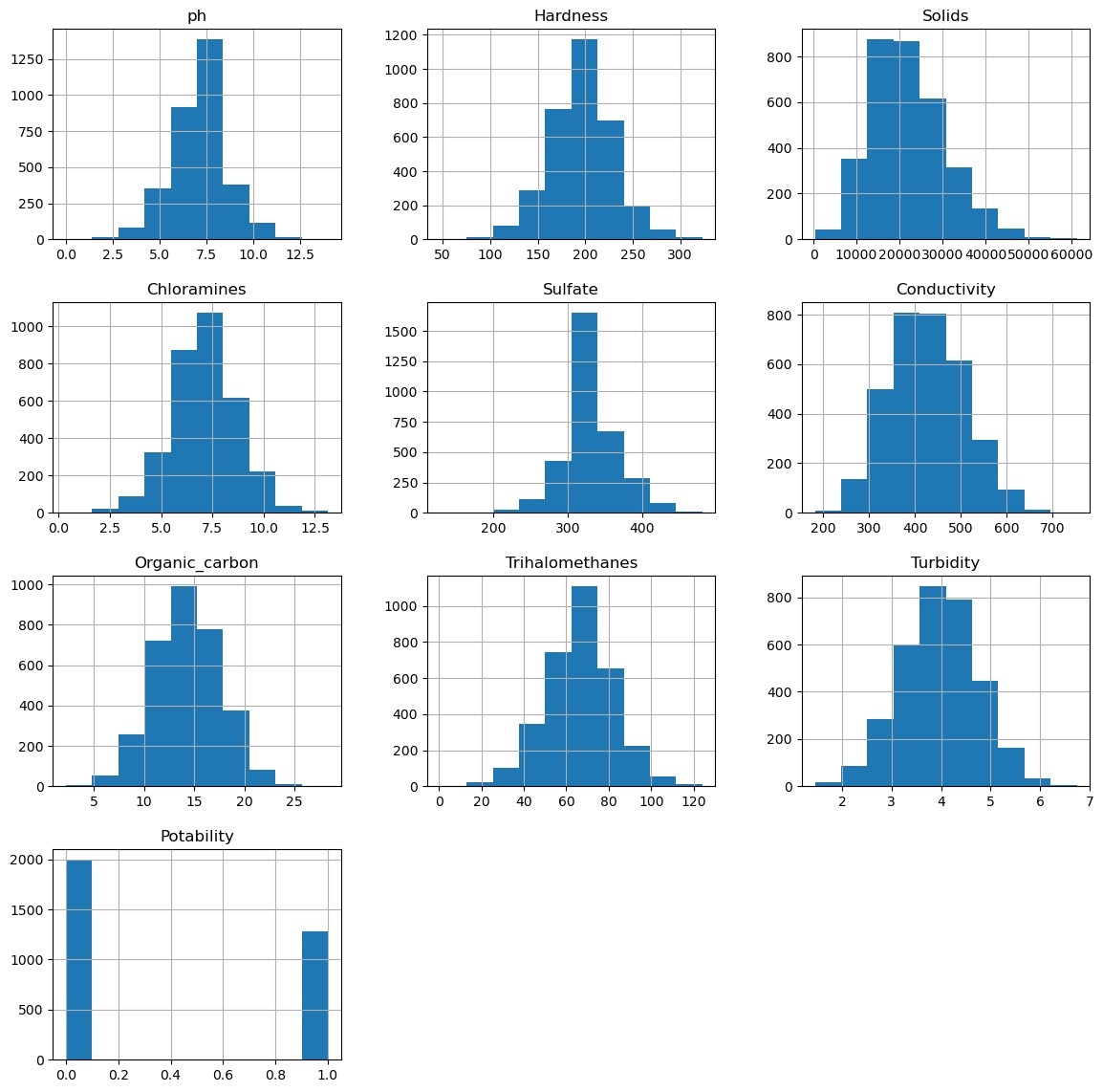
))

plt

**.**

show

()



plt

**.**

figure

(

figsize

**=**

(

13

,

8

))

sns

**.**

heatmap

(

data

**.**

corr

()

,

annot

**=**

**True**

,

cmap

**=**

'terrain'

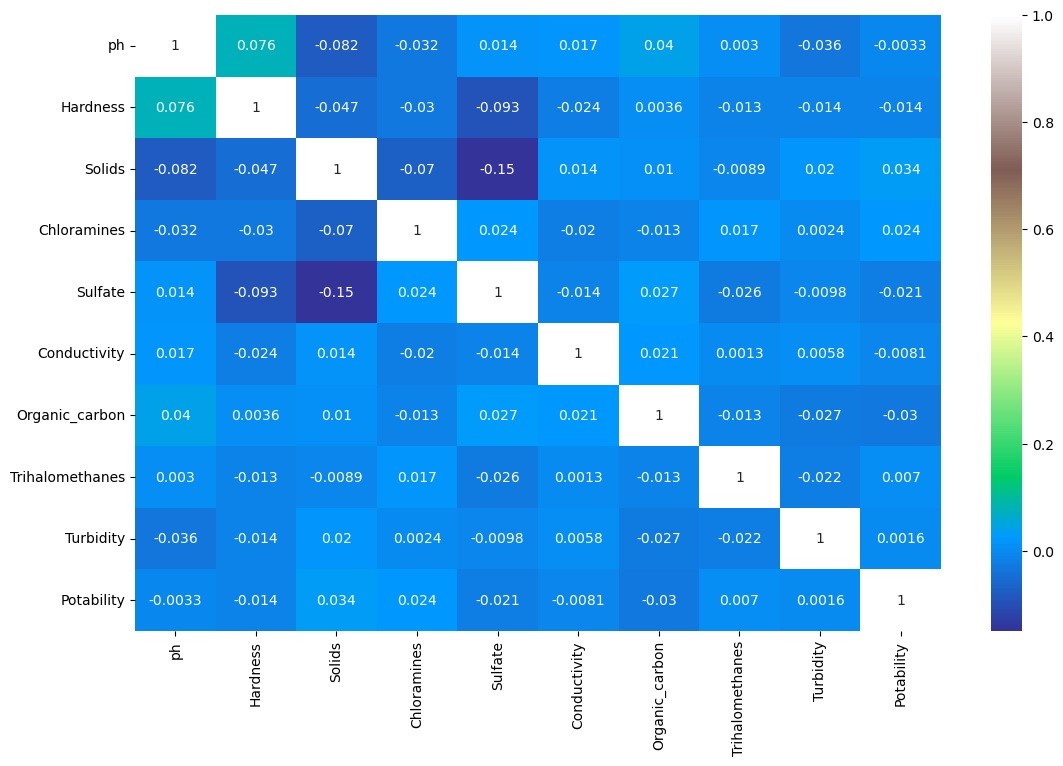
)

plt

**.**

show

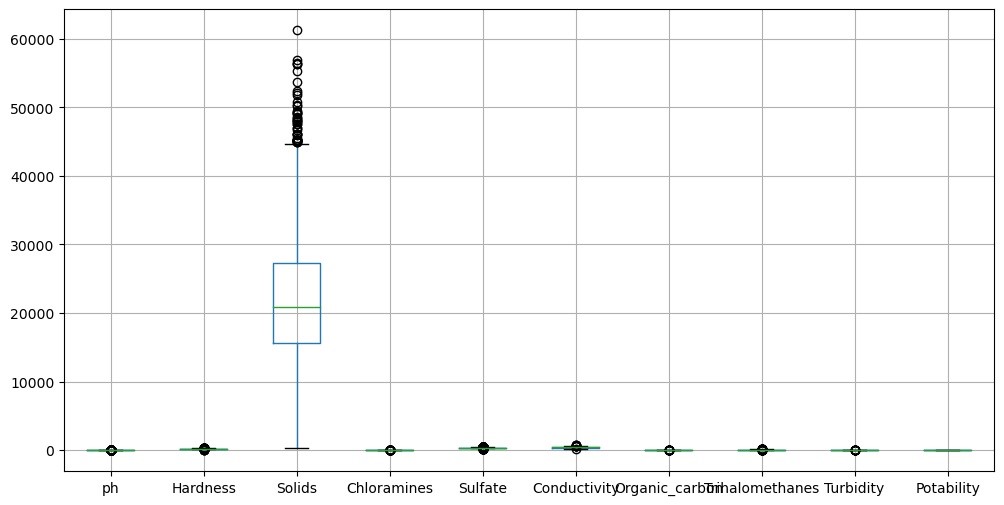
()



<

AxesSubplot

:>



data

**.**

boxplot

(

figsize

**=**

(

12

,

6

))



x

**=**

data

**.**

drop

(

'Potability'

,

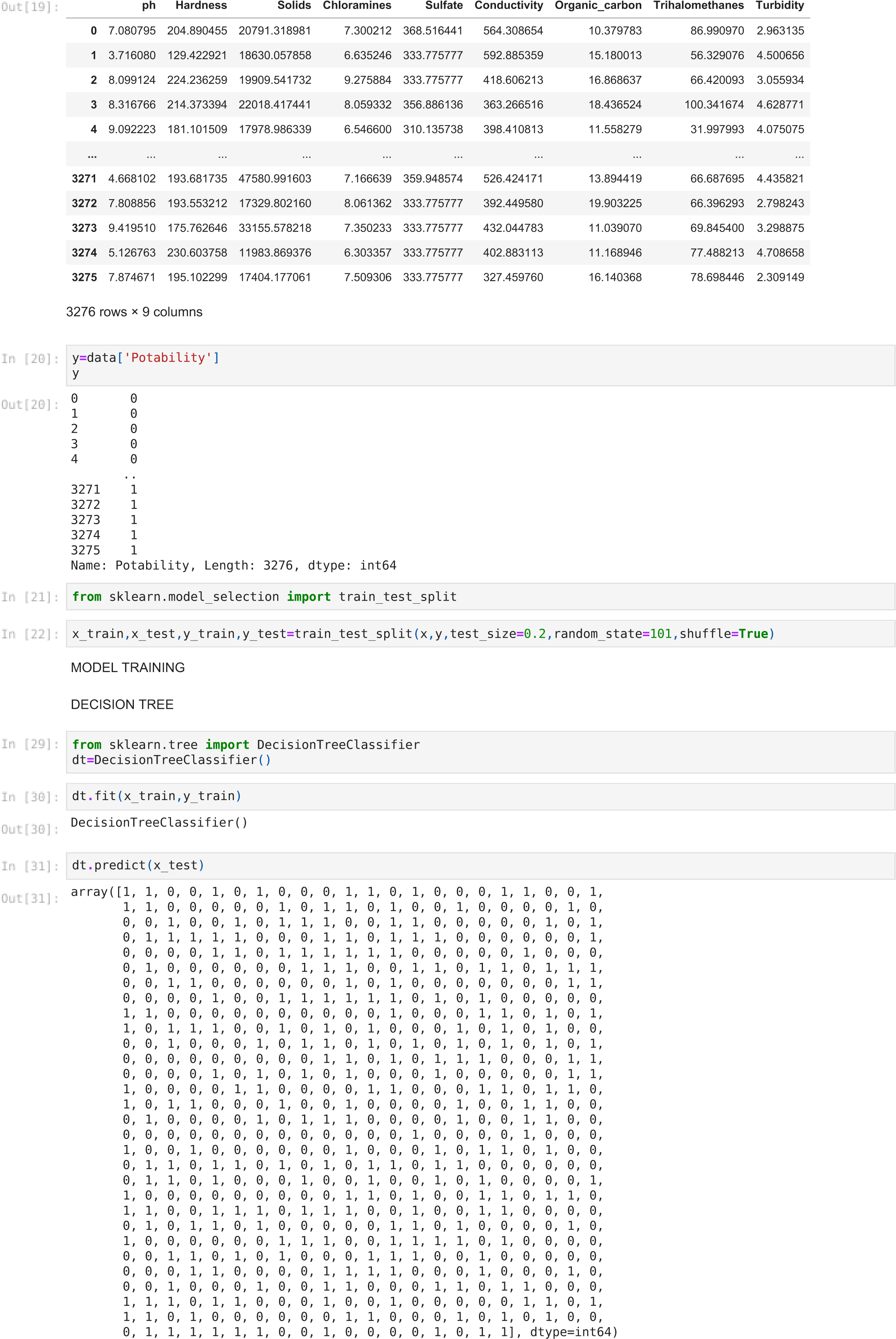
axis

**=**

1

)

x



0.5777439024390244

|  |
| --- |
| confusion\_matrix(y\_prediction,y\_test) |

array([[263, 138],

[139, 116]], dtype=int64)

|  |
| --- |
| y\_test**.**shape |

(656,)

|  |  |  |
| --- | --- | --- |
|  |  | |
| Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js | |  |

|  |
| --- |
| **from** sklearn.metrics **import** accuracy\_score,confusion\_matrix accuracy\_score(y\_prediction,y\_test) |

|  |
| --- |
| y\_prediction**=**dt**.**predict(x\_test) |

# HYPER PARAMETER TUNING

|  |
| --- |
| **from** sklearn.model\_selection **import** GridSearchCV  **from** sklearn.model\_selection **import** RepeatedStratifiedKFold dt**=**DecisionTreeClassifier()  criterion**=**["gini","entropy"] splitter**=**["best","random"] min\_samples\_split**=**range(1,10)  parameters**=**dict(criterion**=**criterion,splitter**=**splitter,min\_samples\_split**=**min\_samples\_split) cv**=** RepeatedStratifiedKFold(n\_splits**=**5,random\_state**=**101)  grid\_search\_cv\_dt**=**GridSearchCV(estimator**=**dt,param\_grid**=**parameters,scoring**=**'accuracy',cv**=**cv) |
| grid\_search\_cv\_dt**.**fit(x\_train,y\_train) |
| C:\Users\ILAKIYA\anaconda3\lib\site-packages\sklearn\model\_selection\\_validation.py:372: FitFailedWarning:  200 fits failed out of a total of 1800.  The score on these train-test partitions for these parameters will be set to nan. If these failures are not expected, you can try to debug them by setting error\_score='raise'.  Below are more details about the failures:  --------------------------------------------------------------------------------  200 fits failed with the following error:  Traceback (most recent call last):  File "C:\Users\ILAKIYA\anaconda3\lib\site-packages\sklearn\model\_selection\\_validation.py", line 680, in \_fit  \_and\_score  estimator.fit(X\_train, y\_train, \*\*fit\_params)  File "C:\Users\ILAKIYA\anaconda3\lib\site-packages\sklearn\tree\\_classes.py", line 937, in fit super().fit(  File "C:\Users\ILAKIYA\anaconda3\lib\site-packages\sklearn\tree\\_classes.py", line 250, in fit raise ValueError(  ValueError: min\_samples\_split must be an integer greater than 1 or a float in (0.0, 1.0]; got the integer 1  warnings.warn(some\_fits\_failed\_message, FitFailedWarning)  C:\Users\ILAKIYA\anaconda3\lib\site-packages\sklearn\model\_selection\\_search.py:969: UserWarning: One or more o f the test scores are non-finite: [ nan nan 0.57946565 0.56641221 0.58019084 0.58145038  0.57820611 0.57923664 0.58137405 0.57912214 0.58133588 0.57744275 0.58183206 0.58091603 0.58076336 0.58538168 0.58141221 0.58122137 nan nan 0.58435115 0.57221374 0.58419847 0.57671756 0.58591603 0.57167939 0.58374046 0.57652672 0.58446565 0.58175573  0.5879771 0.58343511 0.58603053 0.58332061 0.58667939 0.58110687] warnings.warn( |

GridSearchCV(cv=RepeatedStratifiedKFold(n\_repeats=10, n\_splits=5, random\_state=101), estimator=DecisionTreeClassifier(), param\_grid={'criterion': ['gini', 'entropy'], 'min\_samples\_split': range(1, 10),

'splitter': ['best', 'random']}, scoring='accuracy')

|  |
| --- |
| print(grid\_search\_cv\_dt**.**best\_params\_) |

{'criterion': 'gini', 'min\_samples\_split': 9, 'splitter': 'random'}

|  |
| --- |
| prediction\_grid**=**grid\_search\_cv\_dt**.**predict(x\_test) |
| **from** sklearn.metrics **import** accuracy\_score,confusion\_matrix accuracy\_score(y\_prediction,y\_test)**\***100 |

57.77439024390244

**CONCLUSION**

In conclusion, the successful completion of the model training for water quality analysis marks a significant milestone in our ongoing efforts to monitor and manage water resources. This trained model holds the potential to enhance our understanding of water quality, predict potential issues, and ultimately contribute to the preservation of this invaluable resource. As we move forward, it is crucial to continue refining and validating the model's performance to ensure its reliability in real-world applications. With the continued collaboration of experts in the field, this technology can be a valuable tool for safeguarding water quality and supporting sustainable environmental management.