```
## Water Quality Analysis
```

Drinking water potability

Context

Access to safe drinking-water is essential to health, a basic human right and a component of effective policy for health protection. This is important as a health and development issue at a national, regional and local level. In some regions, it has been shown that investments in water supply and sanitation can yield a net economic benefit, since the reductions in adverse health effects and health care costs outweigh the costs of undertaking the interventions.

Content

The water_potability.csv file contains water quality metrics for 3276 different water bodies.

1. pH value: PH is an important parameter in evaluating the acid–base balance of water. It is also the indicator of acidic or alkaline condition of water status. WHO has recommended maximum permissible limit of pH from 6.5 to 8.5. The current investigation ranges were 6.52–6.83 which are in the range of WHO standards.

2. Hardness: Hardness is mainly caused by calcium and magnesium salts. These salts are dissolved from geologic deposits through which water travels. The length of time water is in contact with hardness producing material helps determine how much hardness there is in raw water. Hardness was originally defined as the capacity of water to precipitate soap caused by Calcium and Magnesium.

3. Solids (Total dissolved solids - TDS): Water has the ability to dissolve a wide range of inorganic and some organic minerals or salts such as potassium, calcium, sodium, bicarbonates, chlorides, magnesium, sulfates etc. These minerals produced un-wanted taste and diluted color in appearance of water. This is the important parameter for the use of water. The water with high TDS value indicates that water is highly mineralized. Desirable limit for TDS is 500 mg/l and maximum limit is 1000 mg/l which prescribed for drinking purpose.

4. Chloramines: Chlorine and chloramine are the major disinfectants used in public water systems. Chloramines are most commonly formed when ammonia is added to chlorine to treat drinking water. Chlorine levels up to 4 milligrams per liter (mg/L or 4 parts per million (ppm)) are considered safe in drinking water.

5. Sulfate: Sulfates are naturally occurring substances that are found in minerals, soil, and rocks. They are present in ambient air, groundwater, plants, and food. The principal commercial use of sulfate is in the chemical industry. Sulfate concentration in seawater is about 2,700 milligrams per liter (mg/L). It ranges from 3 to 30 mg/L in most freshwater supplies, although much higher concentrations (1000 mg/L) are found in some geographic locations.

6. Conductivity: Pure water is not a good conductor of electric current rather’s a good insulator. Increase in ions concentration enhances the electrical conductivity of water. Generally, the amount of dissolved solids in water determines the electrical conductivity. Electrical conductivity (EC) actually measures the ionic process of a solution that enables it to transmit current. According to WHO standards, EC value should not exceeded 400 μS/cm.

7. Organic_carbon: Total Organic Carbon (TOC) in source waters comes from decaying natural organic matter (NOM) as well as synthetic sources. TOC is a measure of the total amount of carbon in organic compounds in pure water. According to US EPA < 2 mg/L as TOC in treated / drinking water, and < 4 mg/Lit in source water which is use for treatment.

8. Trihalomethanes: THMs are chemicals which may be found in water treated with chlorine. The concentration of THMs in drinking water varies according to the level of organic material in the water, the amount of chlorine required to treat the water, and the temperature of the water that is being treated. THM levels up to 80 ppm is considered safe in drinking water.

9. Turbidity: The turbidity of water depends on the quantity of solid matter present in the suspended state. It is a measure of light emitting properties of water and the test is used to indicate the quality of waste discharge with respect to colloidal matter. The mean turbidity value obtained for Wondo Genet Campus (0.98 NTU) is lower than the WHO recommended value of 5.00 NTU.

10. Potability: Indicates if water is safe for human consumption where 1 means Potable and 0 means Not potable.

```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
```

```
###!pip install -r https://raw.githubusercontent.com/bentoml/BentoML/main/examples/quickstart/requirements.txt
```

```
# Suppressing Warnings
import warnings
warnings.filterwarnings('ignore')
```

```
# Importing Pandas and NumPy
import pandas as pd, numpy as np
```

```
# Importing all datasets
water_portability = pd.read_csv("/content/water_potability.csv")
water_portability.head(4)
```

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	NaN	204.890455	20791.318981	7.300212	368.516441	564.308654	10.379783	86.990970	2.963135	0
1	3.716080	129.422921	18630.057858	6.635246	NaN	592.885359	15.180013	56.329076	4.500656	0
2	8.099124	224.236259	19909.541732	9.275884	NaN	418.606213	16.868637	66.420093	3.055934	0
3	8.316766	214.373394	22018.417441	8.059332	356.886136	363.266516	18.436524	100.341674	4.628771	0

```
water_portability.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3276 entries, 0 to 3275
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ph               2785 non-null   float64
 1   Hardness         3276 non-null   float64
 2   Solids           3276 non-null   float64
 3   Chloramines      3276 non-null   float64
 4   Sulfate          2495 non-null   float64
 5   Conductivity     3276 non-null   float64
 6   Organic_carbon   3276 non-null   float64
 7   Trihalomethanes  3114 non-null   float64
 8   Turbidity        3276 non-null   float64
 9   Potability       3276 non-null   int64  
dtypes: float64(9), int64(1)
memory usage: 256.1 KB
```

```
water_portability.shape
```

```
(3276, 10)
```

```
water_portability.isnull().sum()
```

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

```
def impute_nan(df,variable,median):
    df[variable+"_median"]=df[variable].fillna(median)
    df[variable+"_random"]=df[variable]
    ##It will have the random sample to fill the na
    random_sample=df[variable].dropna().sample(df[variable].isnull().sum(),random_state=0)
    ##pandas need to have same index in order to merge the dataset
    random_sample.index=df[df[variable].isnull()].index
    df.loc[df[variable].isnull(),variable+'_random']=random_sample
```

```
median=water_portability.ph.median()
```

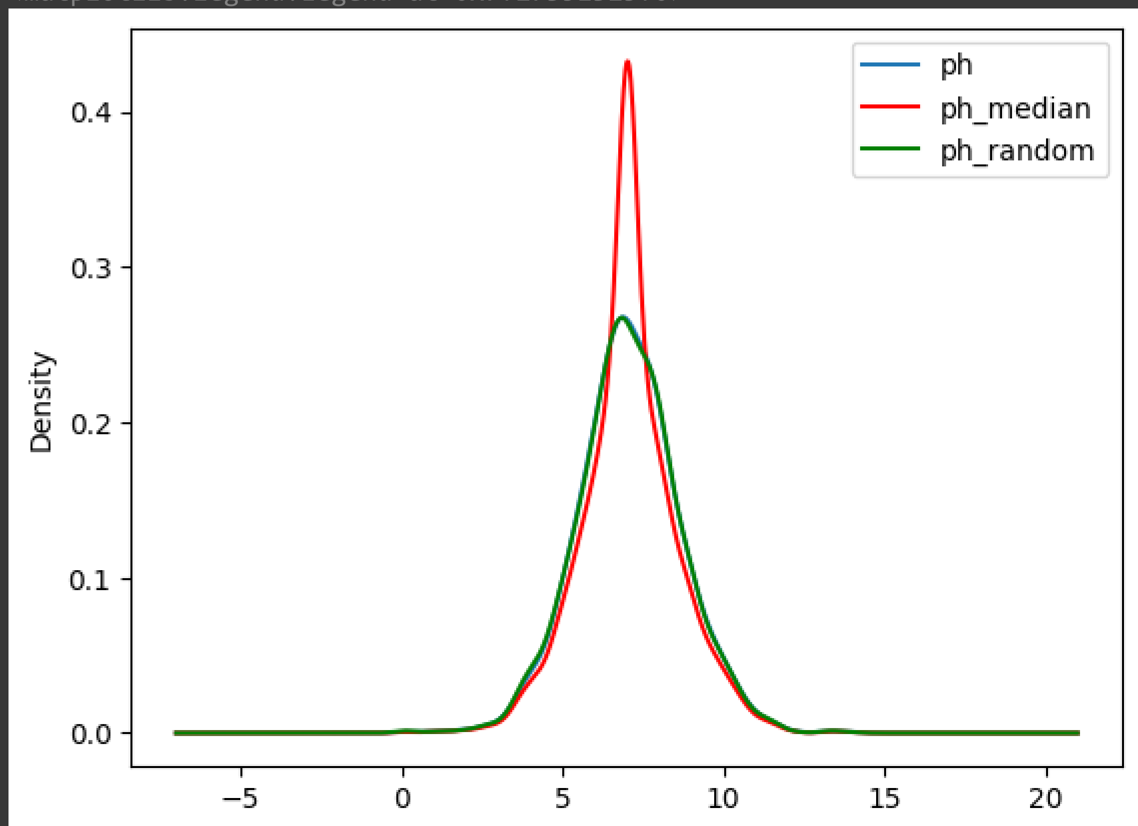
```
impute_nan(water_portability,"ph",median)
```

```
import matplotlib.pyplot as plt
%matplotlib inline
```

```
fig = plt.figure()
ax = fig.add_subplot(111)

water_portability.ph.plot(kind='kde', ax=ax)
water_portability.ph_median.plot(kind='kde', ax=ax, color='red')
water_portability.ph_random.plot(kind='kde', ax=ax, color='green')
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best')
```

```
<matplotlib.legend.Legend at 0x7f27351325f0>
```




```
water_portability = water_portability.drop(columns=["ph","ph_median"])
```

```
water_portability = water_portability.rename(columns={"ph_random": "ph"})
```

```
median=water_portability.Sulfate.median()
```

```
impute_nan(water_portability,"Sulfate",median)
```

```
water_portability = water_portability.drop(columns=["Sulfate","Sulfate_median"])
```

```
water_portability = water_portability.rename(columns={"Sulfate_random": "Sulfate"})
```

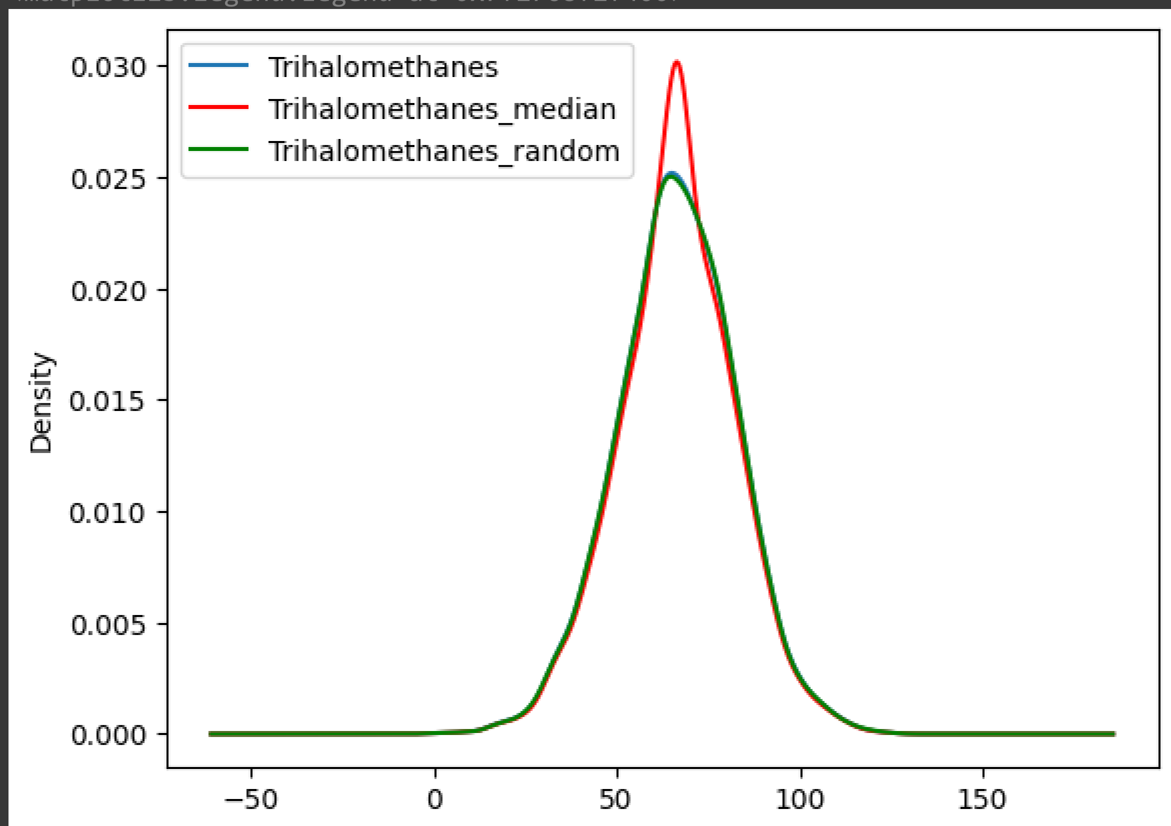
```
median=water_portability.Trihalomethanes.median()
```

```
impute_nan(water_portability,"Trihalomethanes",median)
```

```
fig = plt.figure()
ax = fig.add_subplot(111)

water_portability.Trihalomethanes.plot(kind='kde', ax=ax)
water_portability.Trihalomethanes_median.plot(kind='kde', ax=ax, color='red')
water_portability.Trihalomethanes_random.plot(kind='kde', ax=ax, color='green')
lines, labels = ax.get_legend_handles_labels()
ax.legend(lines, labels, loc='best')
```

```
<matplotlib.legend.Legend at 0x7f2768f2f400>
```




```
water_portability = water_portability.drop(columns=["Trihalomethanes","Trihalomethanes_median"])
```

```
water_portability = water_portability.rename(columns={"Trihalomethanes_random": "Trihalomethanes"})
```

```
water_portability.isnull().sum()
```

```
Hardness          0
Solids            0
Chloramines       0
Conductivity      0
Organic_carbon    0
Turbidity         0
Potability        0
```

```
ph                 0
Sulfate            0
Trihalomethanes    0
dtype: int64
```

```
water_portability.Potability.value_counts()
```

```
0    1998
1    1278
Name: Potability, dtype: int64
```

BENTO ML

BentoML is designed for teams working to bring **machine learning (ML)** models into production in a reliable, scalable, and cost-efficient way. In particular, AI application developers can leverage **BentoML** to easily integrate state-of-the-art pre-trained models into their applications. By seamlessly bridging the gap between model creation and production deployment, BentoML promotes collaboration between **developers** and in-house **data science teams**.

```
import bentoml
import pandas as pd
from bentoml.io import NumpyNdarray, PandasDataFrame, JSON
import numpy as np
from pydantic import BaseModel
```

```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
```

```
water_portability.columns
```

```
Index(['Hardness', 'Solids', 'Chloramines', 'Conductivity', 'Organic_carbon',
       'Turbidity', 'Potability', 'ph', 'Sulfate', 'Trihalomethanes'],
      dtype='object')
```

```
water_portability.head(1)
```

	Hardness	Solids	Chloramines	Conductivity	Organic_carbon	Turbidity	Potability	ph	Sulfate	Trihalomethanes
0	204.890455	20791.318981	7.300212	564.308654	10.379783	2.963135	0	9.074923	368.516441	86.99097

```
X = water_portability.drop(columns = "Potability")
```

```
Y = water_portability["Potability"]
```

```
scaler = StandardScaler()
scaler.fit(X)
X = pd.DataFrame(scaler.transform(X), columns=X.columns)
```

```
bentoml.sklearn.save("scaler", scaler)
```

```
WARNING:bentoml.sklearn:The "bentoml.sklearn.save" method is deprecated. Use "bentoml.sklearn.save_model" instead.
Model(tag="scaler:vdtzxdvkeon6oasc", path="/root/bentoml/models/scaler/vdtzxdvkeon6oasc/")
```

```
###! bentoml models list
```

```
water_portability.head(4)
```

	Hardness	Solids	Chloramines	Conductivity	Organic_carbon	Turbidity	Potability	ph	Sulfate	Trihalomethanes
0	204.890455	20791.318981	7.300212	564.308654	10.379783	2.963135	0	9.074923	368.516441	86.990970
1	129.422921	18630.057858	6.635246	592.885359	15.180013	4.500656	0	3.716080	298.082462	56.329076
2	224.236259	19909.541732	9.275884	418.606213	16.868637	3.055934	0	8.099124	367.224297	66.420093
3	214.373394	22018.417441	8.059332	363.266516	18.436524	4.628771	0	8.316766	356.886136	100.341674

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=10)
```

```
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
```

```
(2293, 9) (983, 9) (2293,) (983,)
```

```
from sklearn import svm
```

```
clf_svm = svm.SVC(gamma='scale')
```

```
bentoml.sklearn.save("clf_svm", clf_svm)
clf_svm.fit(X_train, y_train)
```

```
WARNING:bentoml.sklearn:The "bentoml.sklearn.save" method is deprecated. Use "bentoml.sklearn.save_model" instead.
```

▾ SVC

SVC()

```
clf_rf = RandomForestClassifier()
clf_rf.fit(X_train, y_train)
```

```
▾ RandomForestClassifier
RandomForestClassifier()
```

```
bentoml.sklearn.save("clf_rf", clf_rf)
```

```
WARNING:bentoml.sklearn:The "bentoml.sklearn.save" method is deprecated. Use "bentoml.sklearn.save_model" instead.
Model(tag="clf_rf:vhhiamfkeon6oasc", path="/root/bentoml/models/clf_rf/vhhiamfkeon6oasc/")
```

```
clf_dt = DecisionTreeClassifier()
clf_dt.fit(X_train, y_train)
```

```
▾ DecisionTreeClassifier
DecisionTreeClassifier()
```

```
bentoml.sklearn.save("clf_dt", clf_dt)
```

```
WARNING:bentoml.sklearn:The "bentoml.sklearn.save" method is deprecated. Use "bentoml.sklearn.save_model" instead.
Model(tag="clf_dt:vh3qpffkeon6oasc", path="/root/bentoml/models/clf_dt/vh3qpffkeon6oasc/")
```

```
clf_lg = LogisticRegression()
clf_lg.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
bentoml.sklearn.save("clf_lg", clf_lg)
```

```
WARNING:bentoml.sklearn:The "bentoml.sklearn.save" method is deprecated. Use "bentoml.sklearn.save_model" instead.
Model(tag="clf_lg:vijgr7vkeon6oasc", path="/root/bentoml/models/clf_lg/vijgr7vkeon6oasc/")
```

```
scaler = bentoml.sklearn.get("scaler:latest").to_runner()
clf_svm = bentoml.sklearn.get("clf_svm:latest").to_runner()
clf_rf = bentoml.sklearn.get("clf_rf:latest").to_runner()
clf_dt = bentoml.sklearn.get("clf_dt:latest").to_runner()
clf_lg = bentoml.sklearn.get("clf_lg:latest").to_runner()
```

```
scaler.init_local()
clf_svm.init_local()
clf_rf.init_local()
clf_dt.init_local()
clf_lg.init_local()
```

```
WARNING:bentoml._internal.runner.runner:'Runner.init_local' is for debugging and testing only. Make sure to remove it before deploying to production.
WARNING:bentoml._internal.runner.runner:'Runner.init_local' is for debugging and testing only. Make sure to remove it before deploying to production.
```

```
WARNING:bentoml._internal.runner.runner:'Runner.init_local' is for debugging and testing only. Make sure to remove it before deploying to production.
WARNING:bentoml._internal.runner.runner:'Runner.init_local' is for debugging and testing only. Make sure to remove it before deploying to production.
WARNING:bentoml._internal.runner.runner:'Runner.init_local' is for debugging and testing only. Make sure to remove it before deploying to production.
```

```
###!bentoml models list
```

```
##!bentoml models get clf_svm:latest
```

```
###!bentoml models get clf_rf:latest
```

```
###!bentoml models get clf_dt:latest
```

```
###!bentoml models get clf_lg:latest
```

```
###!bentoml models get scaler:latest
```

```
print(X_train.columns)
print(X_test.columns)
```

```
X_train = pd.DataFrame(X_train)
```

```
X_test = pd.DataFrame(X_test)
```

```
y_train = pd.DataFrame(y_train)
```

```
y_test = pd.DataFrame(y_test)
```

```
y_train.rename(columns = {0:"Potability"}, inplace=True)
y_test.rename(columns = {0:"Potability"}, inplace=True)
```

```
# Create service with the model
service = bentoml.Service(
    "water_quality_prediction", runners=[scaler, clf_svm, clf_rf, clf_dt, clf_lg]
)
```

```
clf_rf = bentoml.sklearn.get("clf_rf:latest").to_runner()
```

```
svc = bentoml.Service("rf_classifier", runners=[clf_rf])
```

```
@svc.api(input=NumpyNdarray(), output=NumpyNdarray())
def classify(input_series: np.ndarray) -> np.ndarray:
    result = clf_rf.predict.run(input_series)
    return result
```

```
%%writefile service.py
import numpy as np
import bentoml
from bentoml.io import NumpyNdarray

clf_rf = bentoml.sklearn.get("clf_rf:latest").to_runner()

svc = bentoml.Service("clf_rf", runners=[clf_rf])

@svc.api(input=NumpyNdarray(), output=NumpyNdarray())
def classify(input_series: np.ndarray) -> np.ndarray:
    return clf_rf.predict.run(input_series)
```

```
Overwriting service.py
```

```
##!bentoml serve service:svc
```

```
2024-01-03T11:03:28+0000 [INFO] [cli] Environ for worker 0: set CPU thread count to 2
2024-01-03T11:03:28+0000 [INFO] [cli] Prometheus metrics for HTTP BentoServer from "service:svc" can be accessed at http://localhost:3000/metrics.
2024-01-03T11:03:29+0000 [INFO] [cli] Starting production HTTP BentoServer from "service:svc" listening on http://0.0.0.0:3000 (Press CTRL+C to quit)
^C
```

```
import bentoml
bento_model: bentoml.Model = bentoml.models.get("clf_rf:latest")

print(bento_model.path)
print(bento_model.info.metadata)
print(bento_model.info.labels)
```

```
/root/bentoml/models/clf_rf/vhhiamfkeon6oasc
{}
{}
```

```
Start coding or generate with AI.
```