IBM Coursera Advanced Data Science Capstone Project

Water Quality :

Drinking water potability

BY : Hrishikesh Kini

Outlines

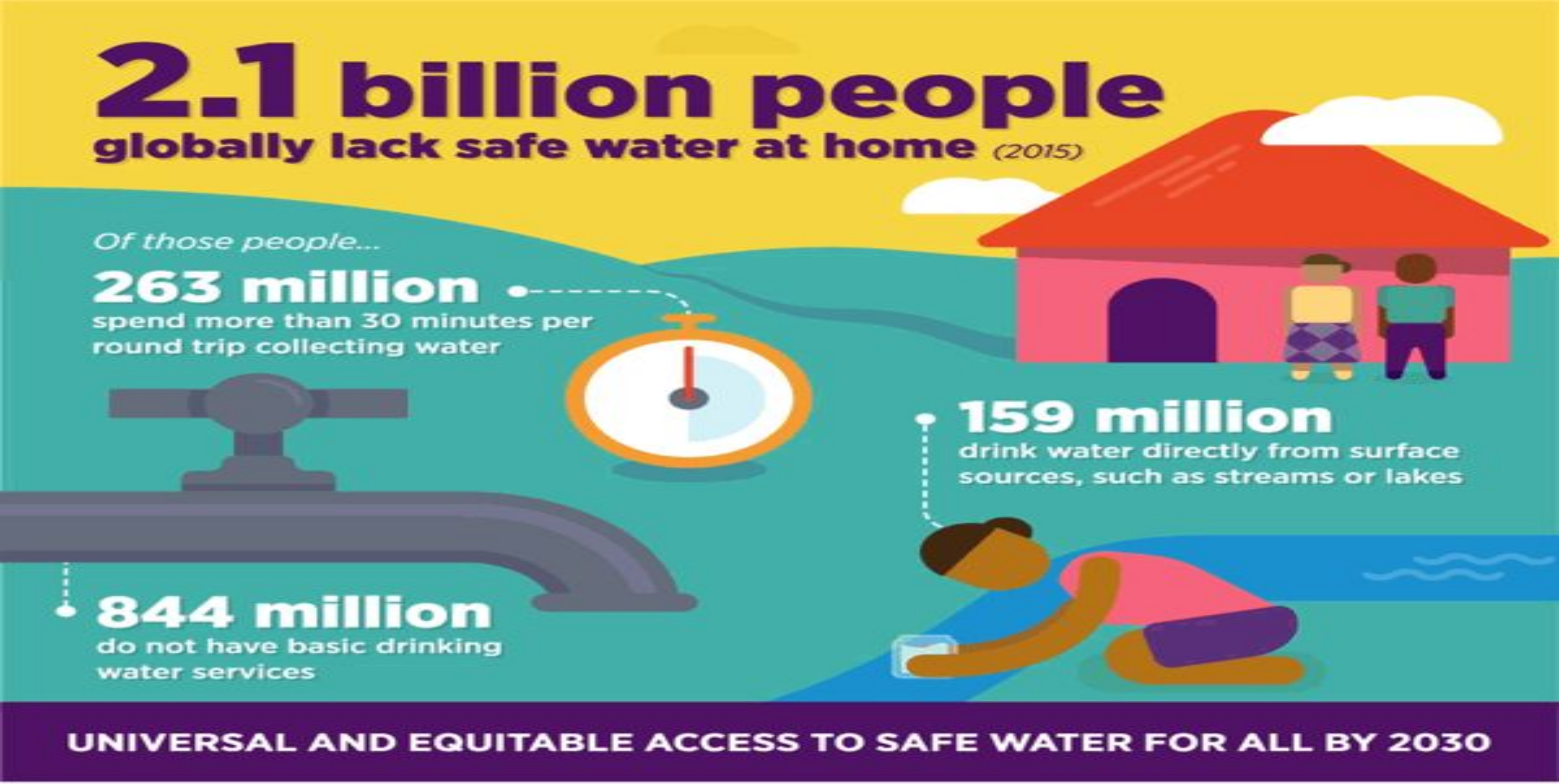
- Introduction
- Dataset - Use case
- Data Exploration
- Data Preprocessing
- Model selection
- Model definition and training
- Model evaluation
- Hyperparameter Tuning
- Model deployment

Safe and readily available water is important for public health, whether it is used for drinking, domestic use, food production or recreational purposes. Improved water supply and sanitation, and better management of water resources, can boost countries' economic growth and can contribute greatly to poverty reduction.

Contaminated water and poor sanitation are linked to transmission of diseases such as **cholera, diarrhoea, dysentery, hepatitis A, typhoid, and polio**. Absent, inadequate, or inappropriately managed water and sanitation services expose individuals to preventable health risks.

This is particularly the case in health care facilities where both patients and staff are placed at additional risk of infection and disease when water, sanitation, and hygiene services are lacking. Globally, **15%** of patients develop an infection during a hospital stay, with the proportion much greater in low-income countries.

So, I took some inspiration from this to use this **Water Quality** dataset to understand what constitutes to safe, Potable water and apply machine learning to it to distinguish between Potable and Non-Potable water.



World Health Organization unicef

Search

Dataset

Water Quality

Drinking water potability

Aditya Kadiwal • updated 7 months ago (Version 3)

808

Data Tasks (2) Code (284) Discussion (18) Activity Metadata

Download (525 kB) New Notebook

Usability 10.0

License CC0: Public Domain

Tags earth and nature, beginner, energy, public health, environment and 2 more

Description

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

1. **ph**: pH of 1. water (0 to 14).
2. **Hardness**: Capacity of water to precipitate soap in mg/L.
3. **Solids**: Total dissolved solids in ppm.
4. **Chloramines**: Amount of Chloramines in ppm.
5. **Sulfate**: Amount of Sulfates dissolved in mg/L.
6. **Conductivity**: Electrical conductivity of water in μS/cm.
7. **Organic_carbon**: Amount of organic carbon in ppm.
8. **Trihalomethanes**: Amount of Trihalomethanes in μg/L.
9. **Turbidity**: Measure of light emitting property of water in NTU.
10. **Potability**: Indicates if water is safe for human consumption. Potable - 1 and Not potable - 0

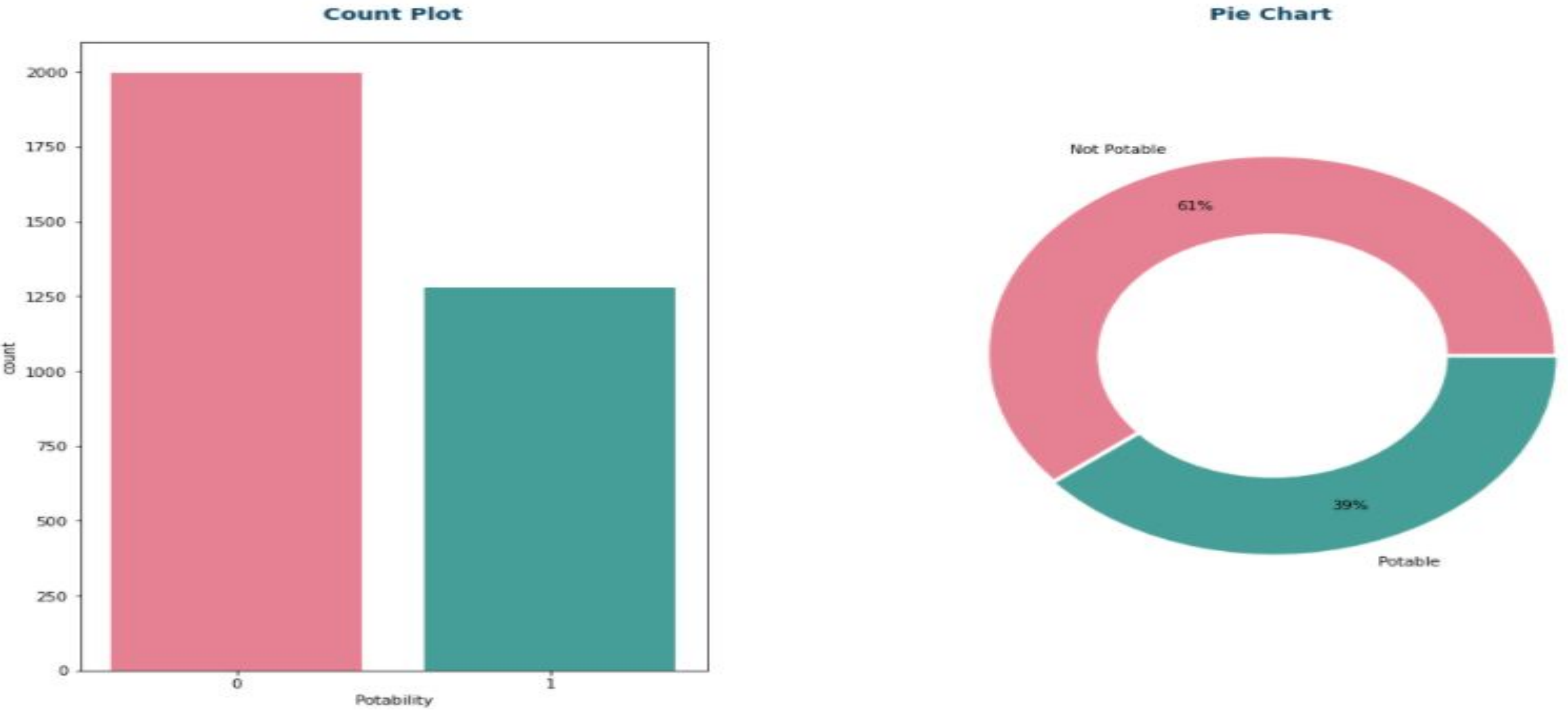
```
raw_data = pd.read_csv(body)
raw_data.head()
```

	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
4	9.092223	181.101509	17978.986339	6.546600	310.135738	398.410813	11.558279	31.997993	4.075075	0

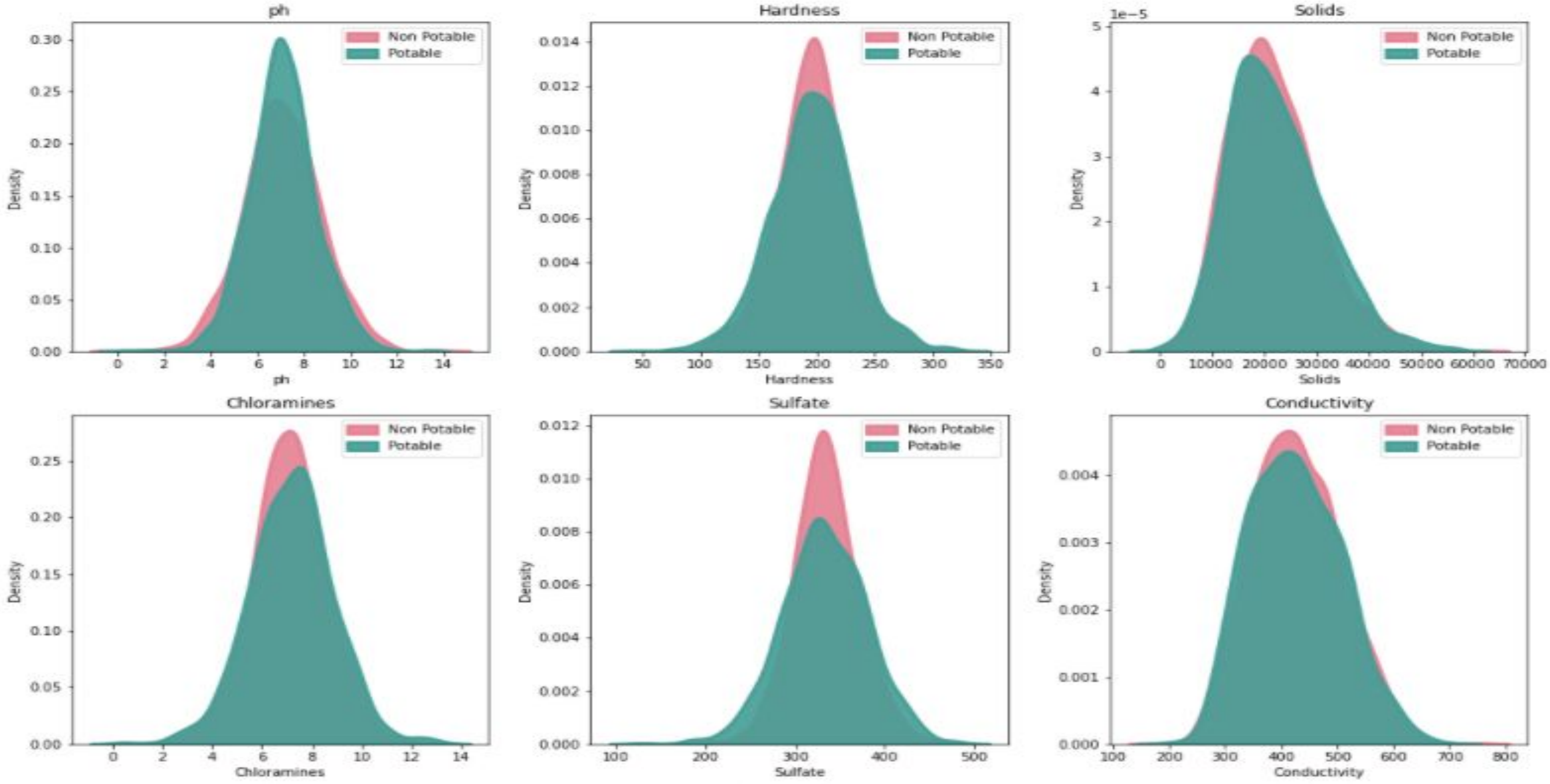
```
raw_data.info()
```

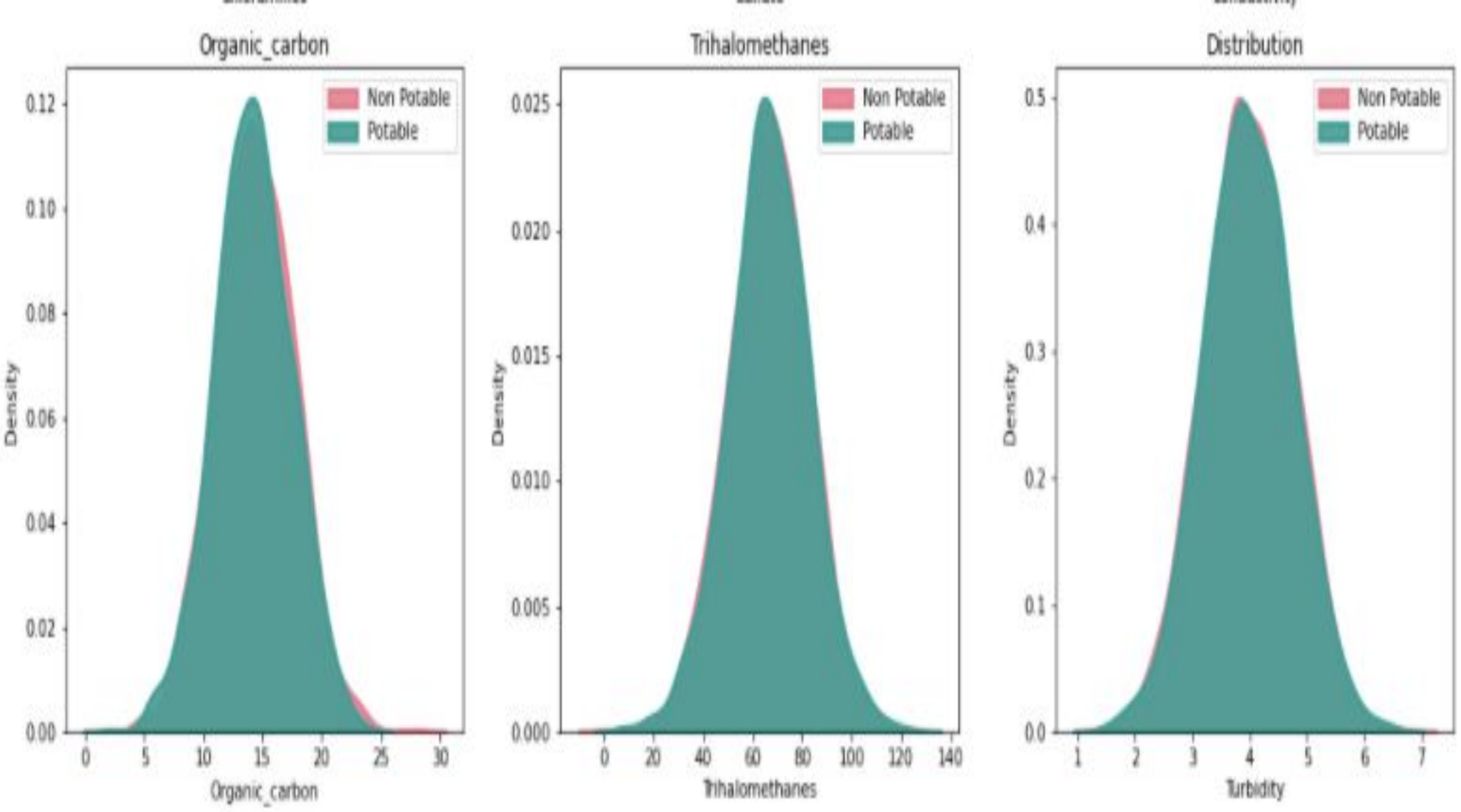
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3276 entries, 0 to 3275
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ph               2785 non-null   float64
 1   Hardness         3276 non-null   float64
 2   Solids           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
```

Potablity of Water Quality

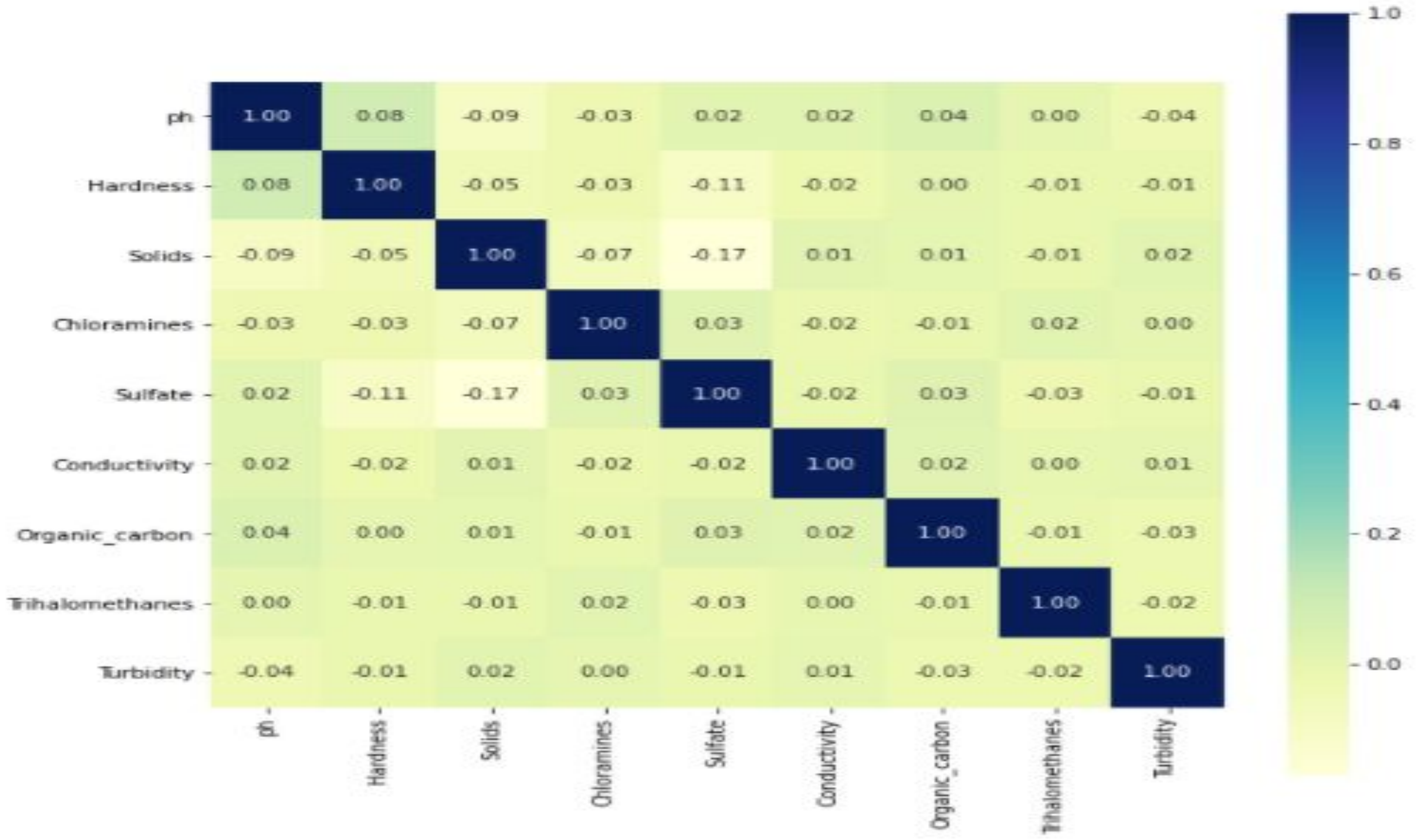


Distribution of features





Organic_carbon
Trihalomethanes
Distribution
Non Potable
Potable
Density
Organic_carbon
Trihalomethanes
Turbidity


ph
Hardness
Solids
Chloramines
Sulfate
Conductivity
Organic_carbon
Trihalomethanes
Turbidity
1.00 0.08 -0.09 -0.03 0.02 0.02 0.04 0.00 -0.04
0.08 1.00 -0.05 -0.03 -0.11 -0.02 0.00 -0.01 -0.01
-0.09 -0.05 1.00 -0.07 -0.17 0.01 0.01 -0.01 0.02
-0.03 -0.03 -0.07 1.00 0.03 -0.02 -0.01 0.02 0.00
0.02 -0.11 -0.17 0.03 1.00 -0.02 0.03 -0.03 -0.01
0.02 -0.02 0.01 -0.02 -0.02 1.00 0.02 0.00 0.01
0.04 0.00 0.01 -0.01 0.03 0.02 1.00 -0.01 -0.03
0.00 -0.01 -0.01 0.02 -0.03 0.00 -0.01 1.00 -0.02
-0.04 -0.01 0.02 0.00 -0.01 0.01 -0.03 -0.02 1.00
1.0
0.8
0.6
0.4
0.2
0.0


Missing Values Per Feature
Sulfate
ph
Trihalomethanes
Hardness
Solids
Chloramines
Conductivity
Organic_carbon
Turbidity
Potability
781
491
162
0
0
0
0
0
0
0
0
700
600
500
400
300
200
100
0

```
data.Potability.value_counts()
```

```
0    1200
1     811
Name: Potability, dtype: int64
```

Not potable is much more potable(1200 > 811) so we need to balance the data to prevent bias.

```
notpotable  = data[data['Potability']==0]
potable = data[data['Potability']==1]

from sklearn.utils import resample
df_minority_upsampled = resample(potable, replace = True, n_samples = 1200)

from sklearn.utils import shuffle
data = pd.concat([notpotable, df_minority_upsampled])
data = shuffle(data)
```

```
data.shape
```

```
(2400, 10)
```

```
data.Potability.value_counts()
```

```
0    1200
1    1200
Name: Potability, dtype: int64
```

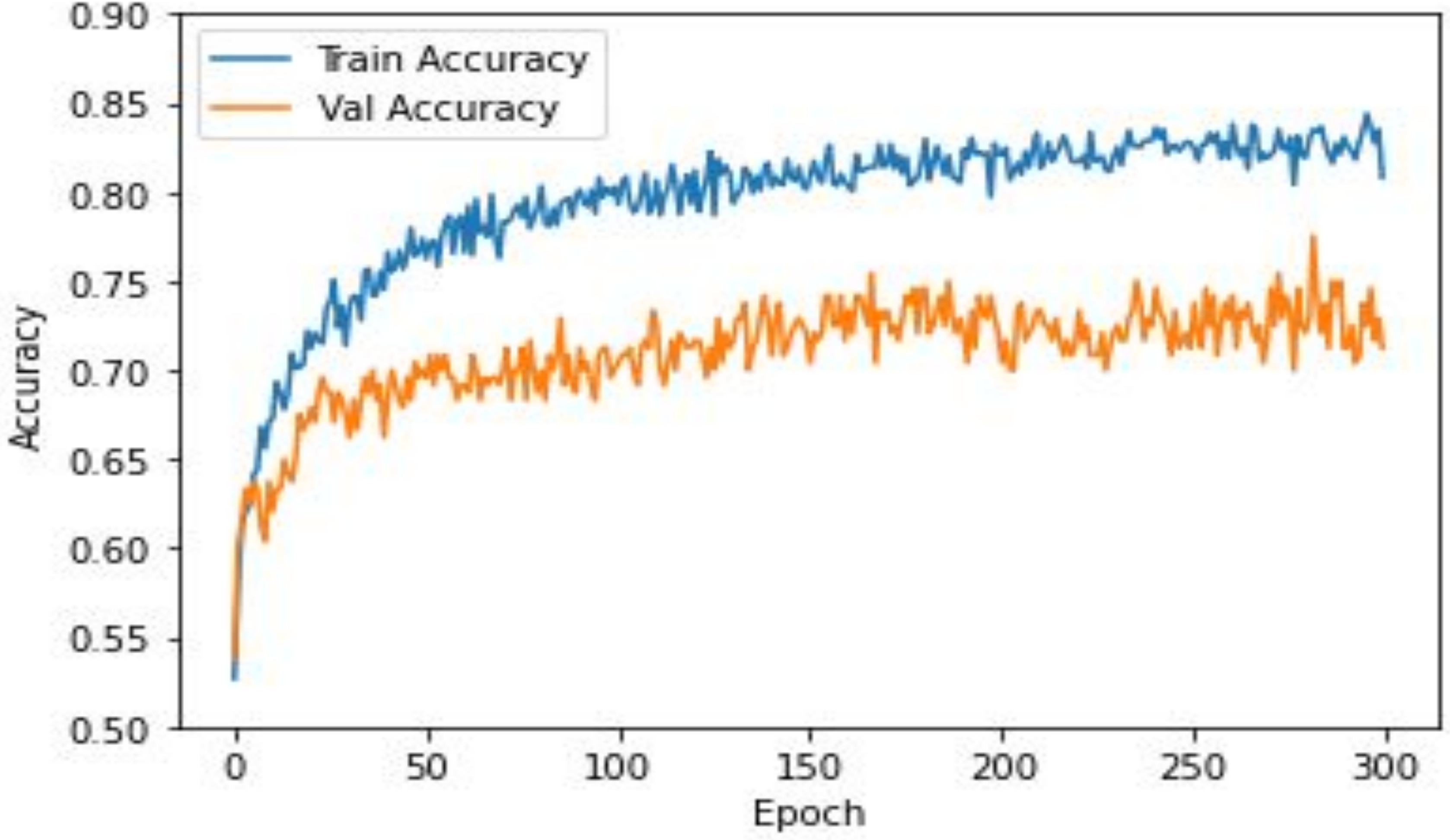
```python
classifiers = [('Logistic Regression', lr), ('K Nearest Neighbours', knn),
                ('Decision Tree', dt), ('Random Forest', rf), ('AdaBoost', ada),
               ('Bagging Classifier', bagging), ('XGBoost', xgb)]
```

```python
from sklearn.metrics import accuracy_score

for classifier_name, classifier in classifiers:

    # Fit clf to the training set
    classifier.fit(X_train, y_train)

    # Predict y_pred
    y_pred = classifier.predict(X_test)
    accuracy = accuracy_score(y_test,y_pred)



    # Evaluate clf's accuracy on the test set
    print('{:s} : {:.2f}'.format(classifier_name, accuracy))
```

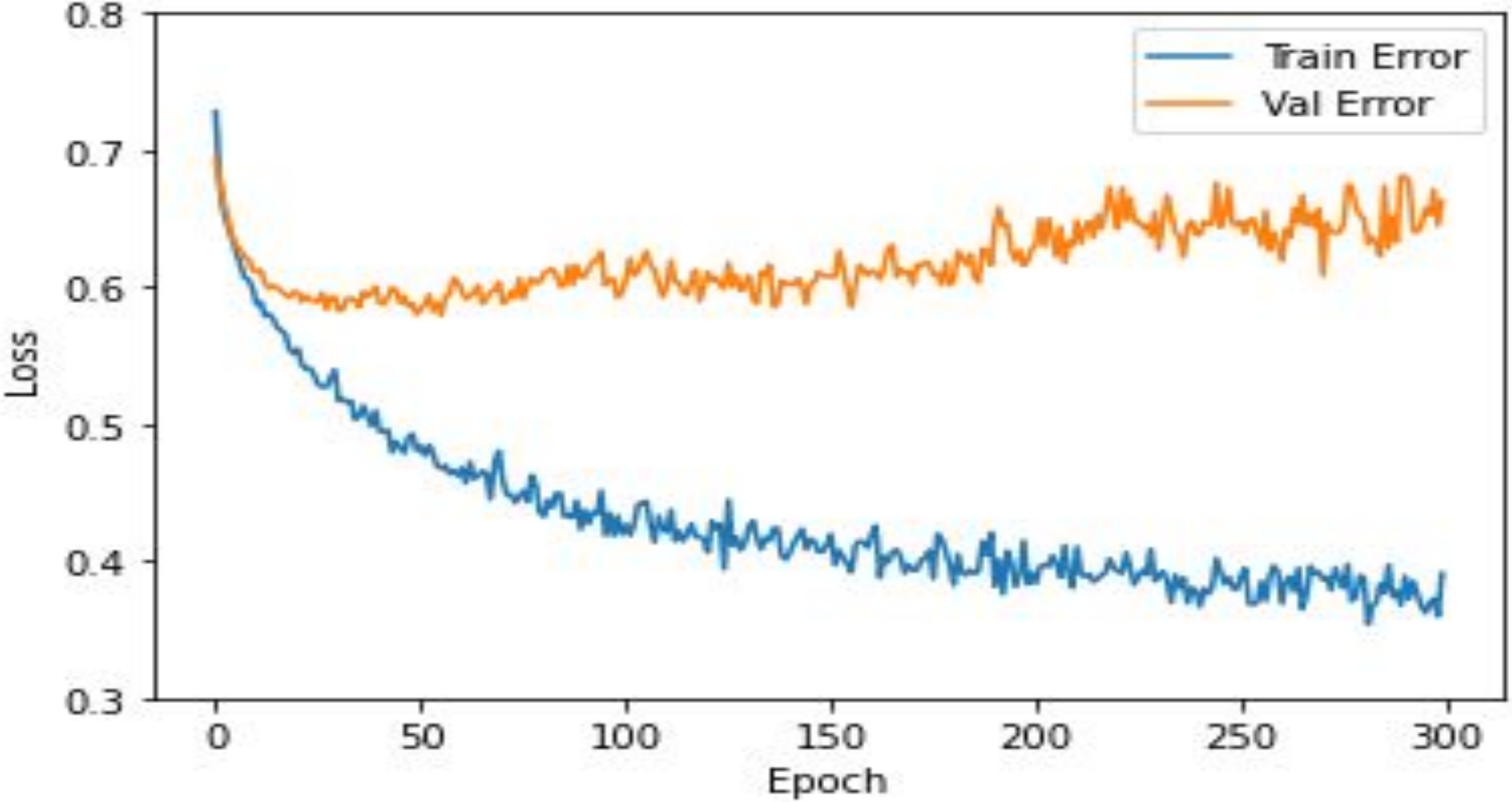
```
Logistic Regression : 0.53
K Nearest Neighbours : 0.76
Decision Tree : 0.77
Random Forest : 0.88
AdaBoost : 0.65
Bagging Classifier : 0.85
```

```
model = models.Sequential()

model.add(layers.Dense(16, input_shape=(9,)))
model.add(BatchNormalization())
model.add(Activation("relu"))

model.add(layers.Dense(32))
model.add(BatchNormalization())
model.add(Activation("relu"))

model.add(layers.Dense(16))
model.add(BatchNormalization())
model.add(Activation("relu"))

model.add(layers.Dense(1))
model.add(Activation("sigmoid"))
```

```
opt = Adam(learning_rate=0.001)

model.compile(loss="binary_crossentropy",
              optimizer=opt,
              metrics=['accuracy'])
```



Train Accuracy
Val Accuracy
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
Accuracy
0
50
100
150
200
250
300
Epoch



Train Error
Val Error
Loss
Epoch
0.8
0.7
0.6
0.5
0.4
0.3
0
50
100
150
200
250
300

Applying StandardScaler befor fitting ML model to normalize the features.

```
from sklearn.preprocessing import StandardScaler
st = StandardScaler()
col= x.columns
x[col] = st.fit_transform(x[col])
x[col]
```

```python
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(x,y, test_size = 0.1)
```

```python
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier()
classifier.fit(X_train,Y_train)
y_pred=classifier.predict(X_test)
```

```python
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(Y_test, y_pred))
```

```
Accuracy: 0.8583333333333333
```

```python
random_grid = {'bootstrap': [True, False],
               'max_depth': [10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 110, None],
               'max_features': ['auto', 'sqrt'],
               'min_samples_leaf': [1, 2, 4],
               'min_samples_split': [2, 5, 10],
               'n_estimators': [130, 180, 230]}
```

```python
from sklearn.model_selection import RandomizedSearchCV
# Use the random grid to search for best hyperparameters
# First create the base model to tune
from sklearn.ensemble import RandomForestRegressor
rf = RandomForestRegressor()
# Random search of parameters, using 3 fold cross validation,
# search across 100 different combinations, and use all available cores
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid, n_iter = 100, cv = 3, verbose=2, random_state=
7, n_jobs = -1)
# Fit the random search model
rf_random.fit(X_train,Y_train)
```

```
rf_random.best_params_
```

```
{'n_estimators': 230,
 'min_samples_split': 2,
 'min_samples_leaf': 1,
 'max_features': 'sqrt',
 'max_depth': 60,
 'bootstrap': False}
```

```
rf_random.best_estimator_
```

```
RandomForestRegressor(bootstrap=False, max_depth=60, max_features='sqrt',
                      n_estimators=230)
```

```
y_pred_r = rf_random.predict(X_test)
y_pred_r = y_pred_r.round()
```

```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(Y_test, y_pred_r))
```

```
Accuracy: 0.8791666666666667
```

	precision	recall	f1-score	support
0	0.87	0.88	0.88	118
1	0.88	0.88	0.88	122
accuracy			0.88	240
macro avg	0.88	0.88	0.88	240
weighted avg	0.88	0.88	0.88	240

Accuracy is 88%

```
scoring_endpoint = client.deployments.get_scoring_href(created_deployment)
print(scoring_endpoint)
```

```
https://us-south.ml.cloud.ibm.com/ml/v4/deployments/5a40960c-2088-47cd-96ab-97baf5e729da/predictions
```

```
# use our WML client to score our model
# add some test data
scoring_payload = {"input_data": [
    {'fields': ['ph', 'Hardness', 'Solids', 'Chloramines', 'Sulfate', 'Conductivity',
                'Organic_carbon', 'Trihalomethanes', 'Turbidity'],
     'values': [[6, 191, 6200, 7.5, 333, 425, 20.8, 98.3, 3.7]]
    }]}
```

```
predictions = client.deployments.score(deployment_uid, scoring_payload)
print('prediction',json.dumps(predictions, indent=2))
```

```
prediction {
  "predictions": [
    {
      "fields": [
        "prediction"
      ],
      "values": [
        [
          0.45217391304347826
        ]
      ]
    }
  ]
}
```

drinkingwaterpotability.herokuapp.com

Water Quality : Drinking water potability

Made by Hrishikesh Kini

Enter The Values

ph

THANK YOU