

WATER QUALITY

ANALYSIS

PHASE-4



Creating visualizations and building a predictive model involves several steps, and it can be a complex process. I'll provide a high-level overview of the steps involved. Keep in mind that the specific tools and techniques you use will depend on the nature of your data and your objectives.

Data Preparation

- Start by cleaning and preprocessing your data. This may involve handling missing values, encoding categorical variables, and scaling numerical features. Ensure your data is in a suitable format for analysis.

Exploratory Data Analysis (EDA)

- Conduct EDA to gain insights into your data. Visualize the data to understand its distribution, relationships between variables, and potential patterns. You can use tools like matplotlib, seaborn, or Tableau for this.

Feature Engineering

- Create new features or transform existing ones to improve the performance of your predictive model. This step often involves domain knowledge and creativity. Feature engineering can have a significant impact on model accuracy.

Data Splitting

- Divide your data into training, validation, and test sets. The training set is used to train the model, the validation set is used for hyperparameter tuning, and the test set is used to evaluate the model's performance.

Model Selection

- Choose an appropriate machine learning or statistical model for your predictive task. The choice of model depends on the nature of the data (classification, regression, time series, etc.) and your specific goals.

Model Training

- Train your chosen model on the training data. Tune hyperparameters as needed to optimize performance. You can use libraries like scikit-learn, TensorFlow, or PyTorch for model training.

Model Evaluation

- Assess the performance of your model using the validation set. Common evaluation metrics include accuracy, precision, recall, F1 score, RMSE, MAE, etc., depending on the problem type.

Hyperparameter Tuning

- Fine-tune your model's hyperparameters to optimize its performance. You can use techniques like grid search, random search, or Bayesian optimization.

Visualization for Model Interpretability

- Visualize the model's predictions and decision boundaries. Tools like SHAP values, Partial Dependence Plots, and LIME can help you interpret and explain your model's predictions.

Final Model Selection and Testing

- After selecting the best-performing model and fine-tuning its hyperparameters, evaluate it on the test set to obtain a final performance estimate.

Deployment

- If the model meets your expectations, deploy it in your application or workflow. Ensure that it's properly integrated and maintained.

Monitoring and Maintenance

- Continuously monitor the model's performance in production. Retrain or update the model as needed to account for changing data distributions or requirements.

Visualization tools and libraries you can use include Matplotlib, Seaborn, Plotly, or interactive dashboards created with tools like Streamlit or Dash.

Remember that the choice of tools and techniques depends on your specific dataset and predictive task. The above steps provide a general framework, and you may need to adapt them to your unique circumstances. Additionally, the choice of machine learning algorithms and visualization techniques should align with the problem you are trying to solve (e.g., regression, classification, clustering, time series forecasting, etc.)

DATA SET LINK:

<https://www.kaggle.com/datasets/adityakadiwal/water-potability>


Installation code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
plt.style.use('fivethirtyeight')
plt.style.use('dark_background')
import seaborn as sns
color = sns.color_palette()
```









```
import plotly.express      as ex
import plotly.graph_objs   as go
import plotly.offline      as pyo
import scipy.stats         as stats

# Histogram of a water quality parameter

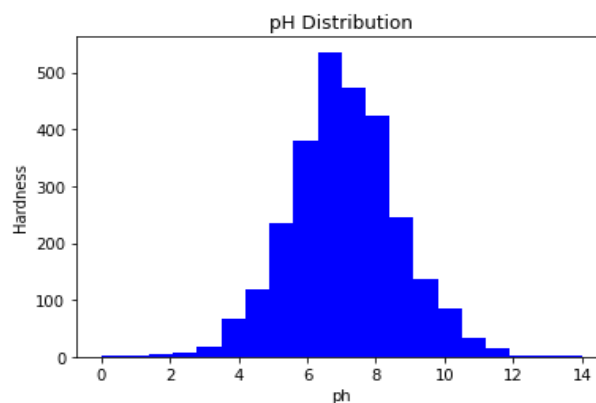
plt.hist(df['pH'], bins=20, color='blue')
plt.xlabel('pH')
plt.ylabel('Frequency')
plt.title('pH Distribution')
plt.show()
```

 jupyter Untitled18 Last Checkpoint: 12 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

       Run    Code 

```
In [10]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('water_quality_data.csv')
plt.hist(df['ph'], bins=20, color='blue')
plt.xlabel('ph')
plt.ylabel('Hardness')
plt.title('pH Distribution')
plt.show()
```



Box plot to identify outliers

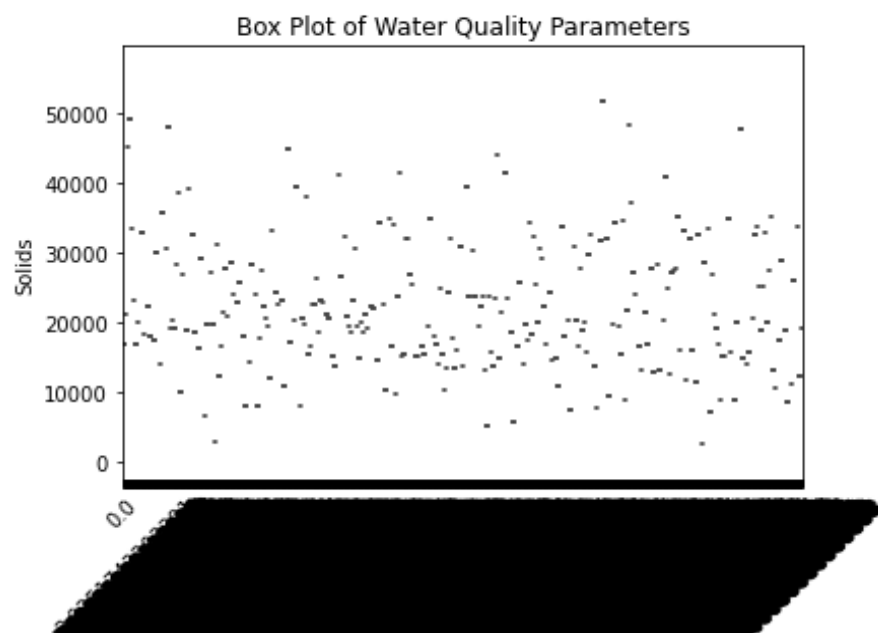
```
sns.boxplot(x='Parameter', y='Value', data=df)
plt.xlabel('Water Quality Parameter')
plt.ylabel('Value')
plt.title('Box Plot of Water Quality Parameters')
plt.xticks(rotation=45)
plt.show()
```

jupyter Untitled20 Last Checkpoint: 10 minutes ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help

Run

```
In [2]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_csv('water_quality_data.csv')
sns.boxplot(x='ph', y='Solids', data=df)
plt.xlabel('ph')
plt.ylabel('Solids')
plt.title('Box Plot of Water Quality Parameters')
plt.xticks(rotation=45)
plt.show()
```



```

print('Boxplot and density distribution of different features by Potability\n')
fig, ax = plt.subplots(ncols=2, nrows=9, figsize=(14, 28))
features = list(df.columns.drop('Potability'))
i=0
for cols in features:
    sns.kdeplot(df[cols], fill=True, alpha=0.4, hue = df.Potability,
        palette=('indianred', 'steelblue'), multiple='stack', ax=ax[i,0])
    sns.boxplot(data= df, y=cols, x='Potability', ax=ax[i, 1],
        palette=('indianred', 'steelblue'))
    ax[i,0].set_xlabel(' ')
    ax[i,1].set_xlabel(' ')
    ax[i,1].set_ylabel(' ')
    ax[i,1].xaxis.set_tick_params(labelsize=14)
    ax[i,0].tick_params(left=False, labelleft=False)
    ax[i,0].set_ylabel(cols, fontsize=16)
    i=i+1
plt.show()

```

```

In [ ]: print('Boxplot and density distribution of different features by Potability\n')

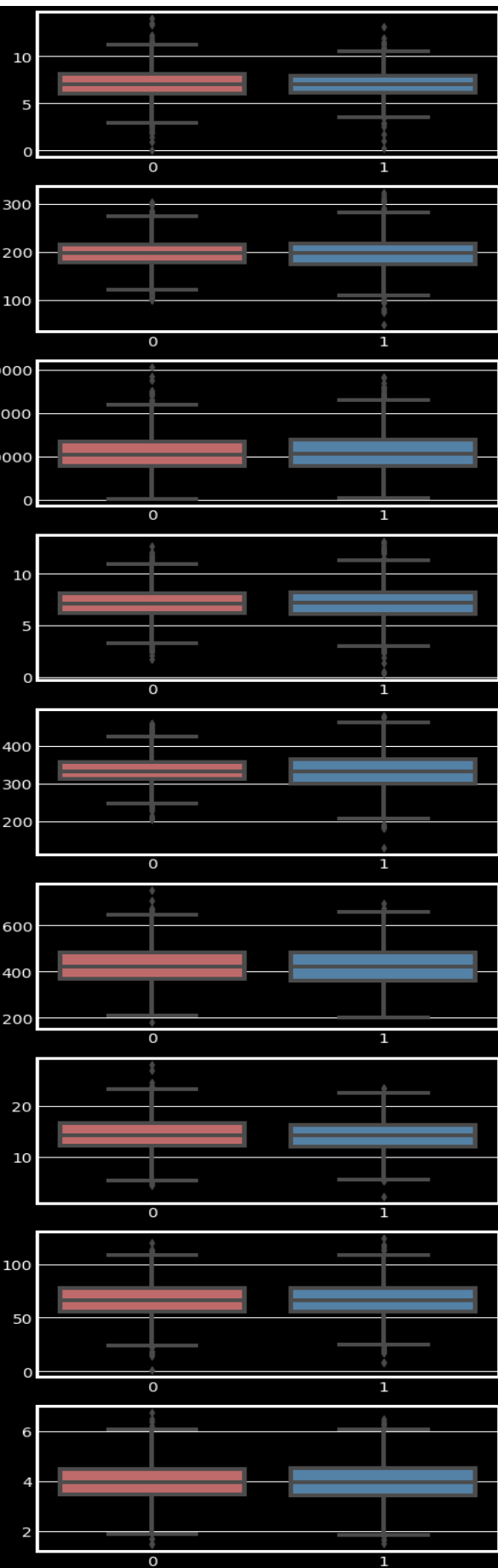
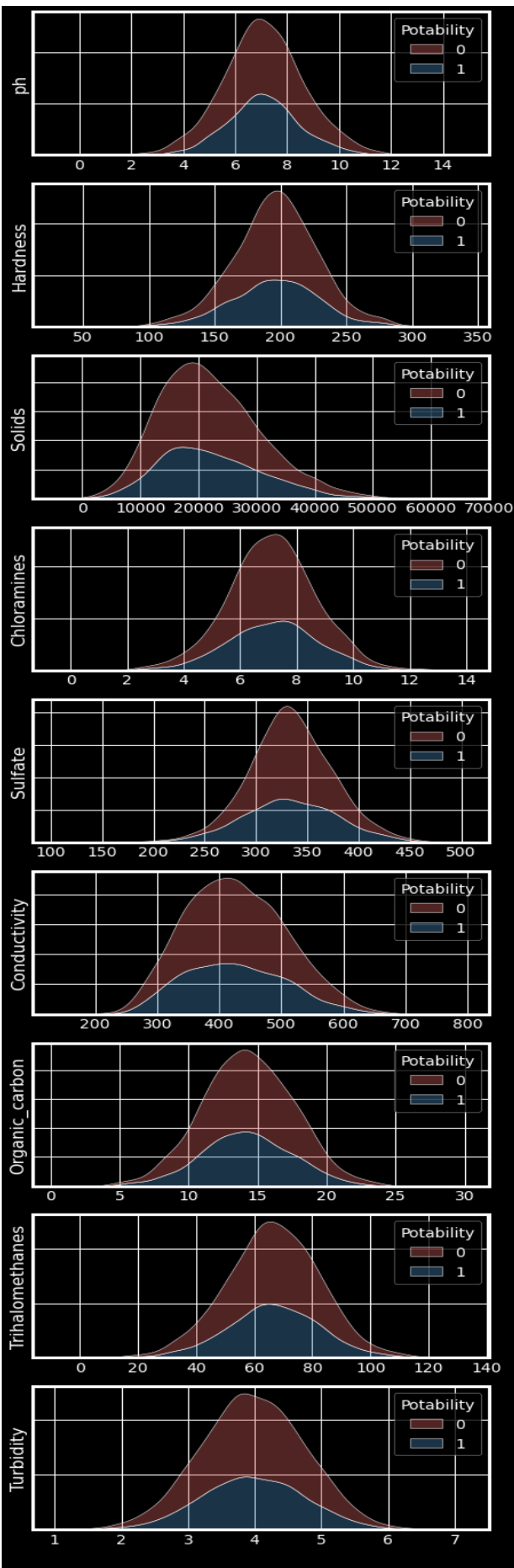
fig, ax = plt.subplots(ncols=2, nrows=9, figsize=(14, 28))

features = list(df.columns.drop('Potability'))
i=0
for cols in features:
    sns.kdeplot(df[cols], fill=True, alpha=0.4, hue = df.Potability,
        palette=('indianred', 'steelblue'), multiple='stack', ax=ax[i,0])

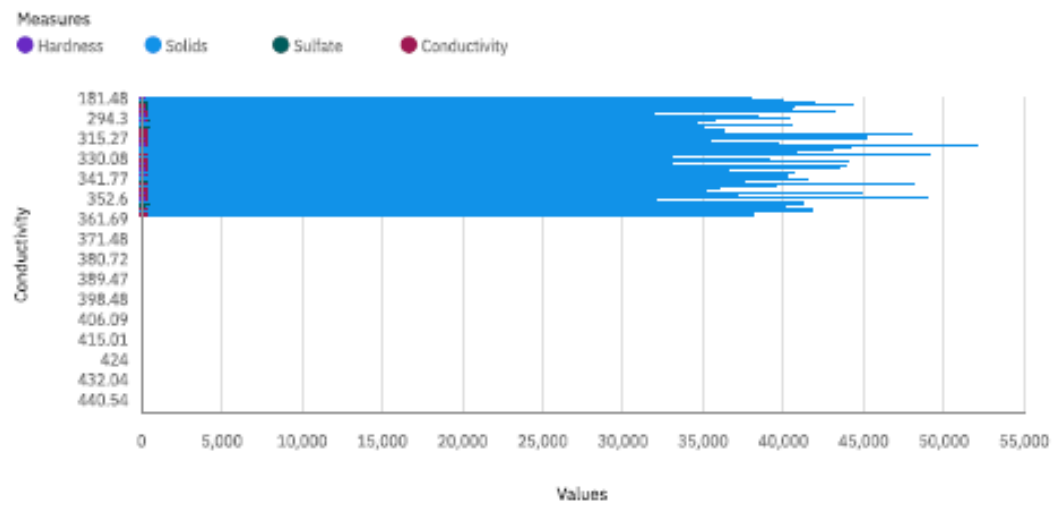
    sns.boxplot(data= df, y=cols, x='Potability', ax=ax[i, 1],
        palette=('indianred', 'steelblue'))
    ax[i,0].set_xlabel(' ')
    ax[i,1].set_xlabel(' ')
    ax[i,1].set_ylabel(' ')
    ax[i,1].xaxis.set_tick_params(labelsize=14)
    ax[i,0].tick_params(left=False, labelleft=False)
    ax[i,0].set_ylabel(cols, fontsize=16)
    i=i+1

plt.show()

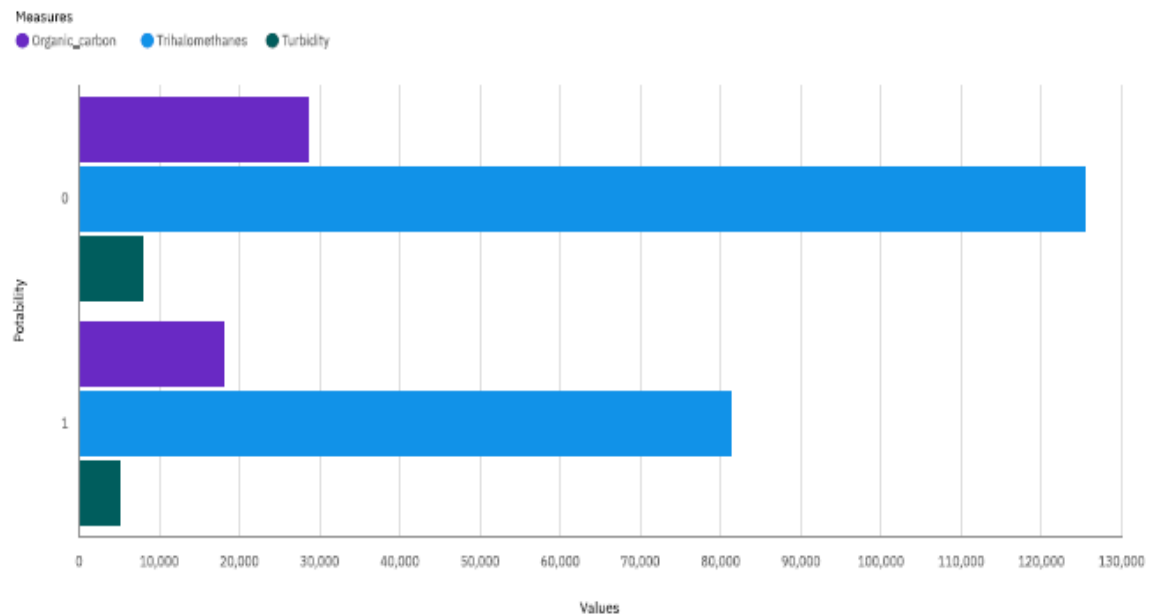
```



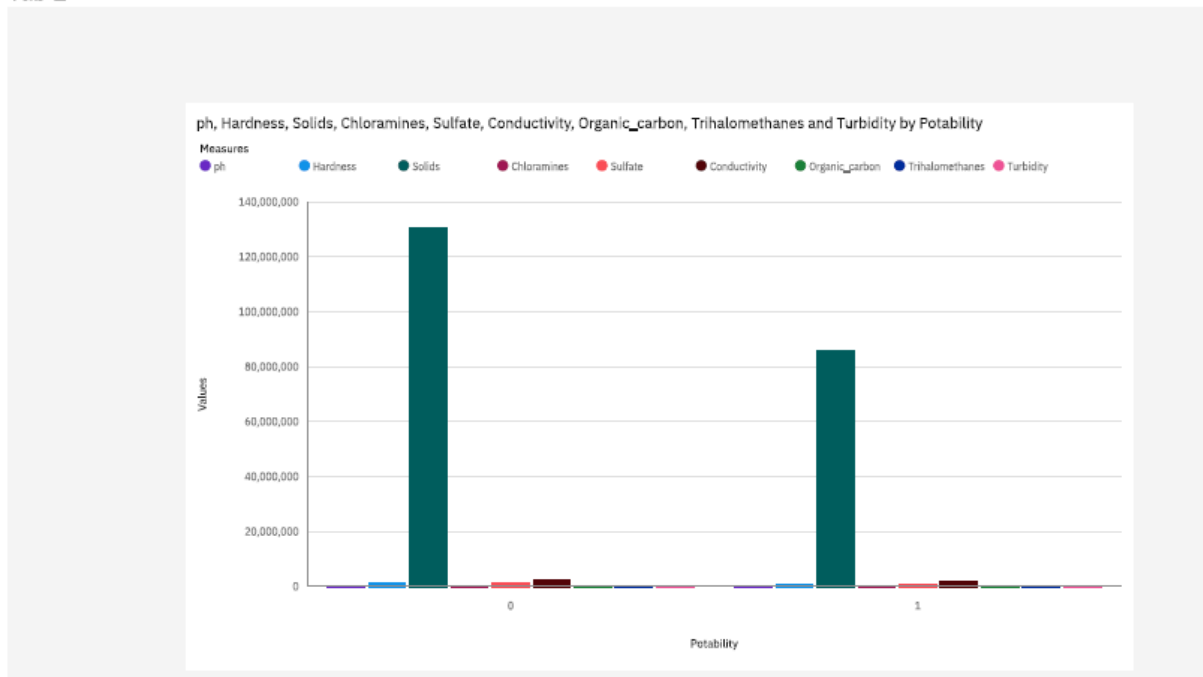
Hardness, Solids, Sulfate and Conductivity by Conductivity



Organic_carbon, Trihalomethanes and Turbidity by Potability

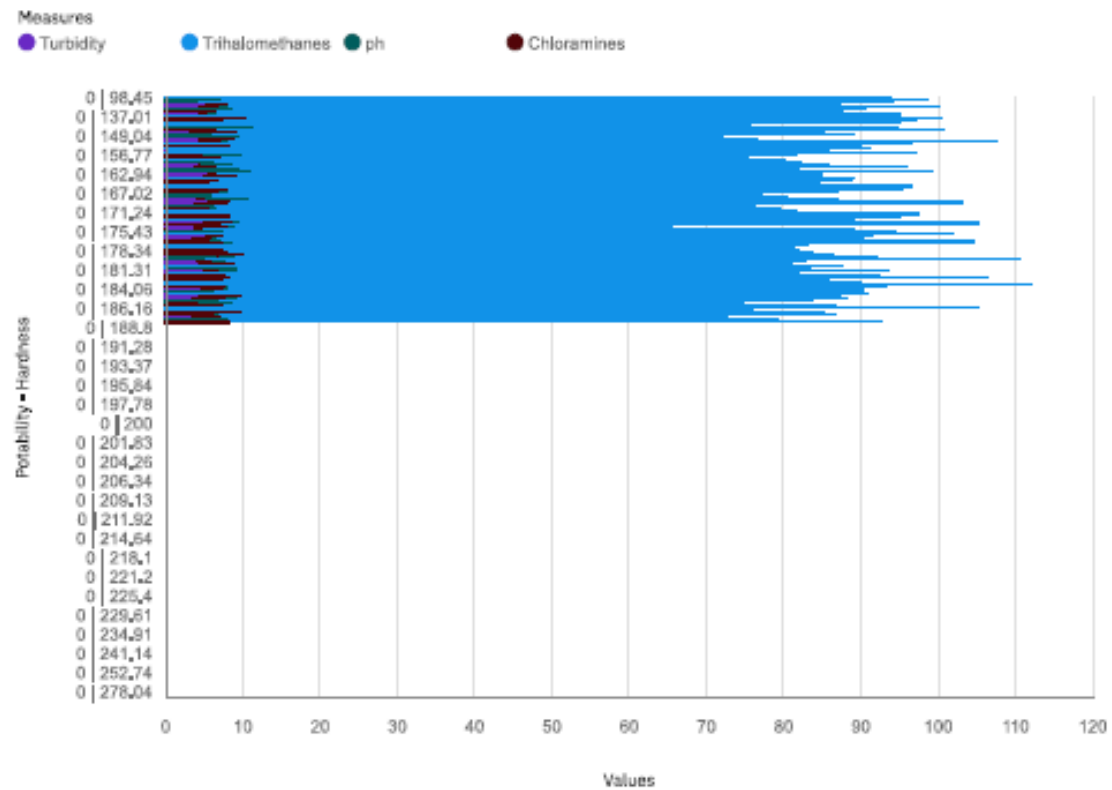


Tab 1



Tab 1

Turbidity, Trihalomethanes, ph and Chloramines by Potability and Hardness



Conclusion

- The Solid levels seem to contain some discrepancy since its values are on an average 40 folds more than the upper limit for safe drinking water.(Desirable limit for TDS is 500 mg/l and maximum limit is 1000 mg/l which prescribed for drinking purpose.)
- The data contains almost equal number of acidic and basic pH level water samples.
- The correlation coefficients between the features were very low.
- Random Forest and XGBoost worked the best to train the model, both gives us f1 score (Balanced with precision & recall) as around 76%.

