Project Tasks Outline

1. Dataset Download and Description

Download the dataset and explain each feature briefly, highlighting its importance for water quality analysis.

```
df = pd.read_csv("dataset.csv")
df.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

```
df.describe()
```

Python

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
count	2785.000000	3276.000000	3276.000000	3276.000000	2495.000000	3276.000000	3276.000000	3114.000000	3276.000000	3276.000000
mean	7.080795	196.369496	22014.092526	7.122277	333.775777	426.205111	14.284970	66.396293	3.966786	0.390110
std	1.594320	32.879761	8768.570828	1.583085	41.416840	80.824064	3.308162	16.175008	0.780382	0.487849
min	0.000000	47.432000	320.942611	0.352000	129.000000	181.483754	2.200000	0.738000	1.450000	0.000000
25%	6.093092	176.850538	15666.690297	6.127421	307.699498	365.734414	12.065801	55.844536	3.439711	0.000000
50%	7.036752	196.967627	20927.833607	7.130299	333.073546	421.884968	14.218338	66.622485	3.955028	0.000000
75%	8.062066	216.667456	27332.762127	8.114887	359.950170	481.792304	16.557652	77.337473	4.500320	1.000000
max	14.000000	323.124000	61227.196008	13.127000	481.030642	753.342620	28.300000	124.000000	6.739000	1.000000

```
df.info()
```

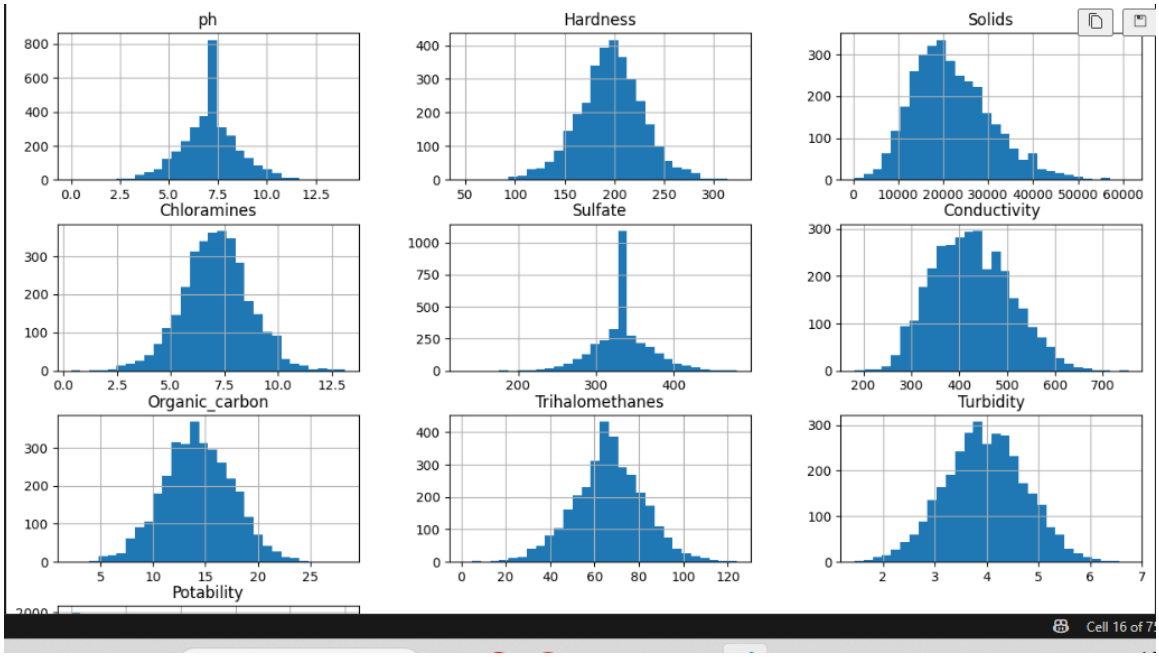
✓ 0.0s

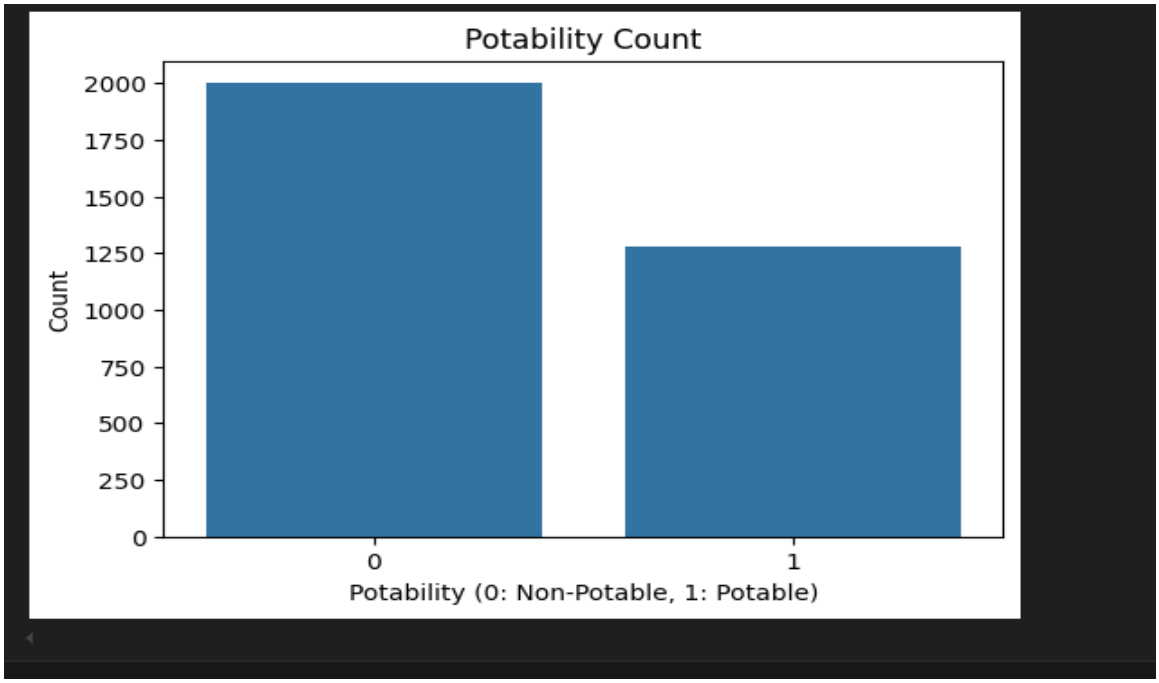
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3276 entries, 0 to 3275
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ph               2785 non-null   float64
 1   Hardness         3276 non-null   float64
 2   Solids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

2. Exploratory Data Analysis (EDA)

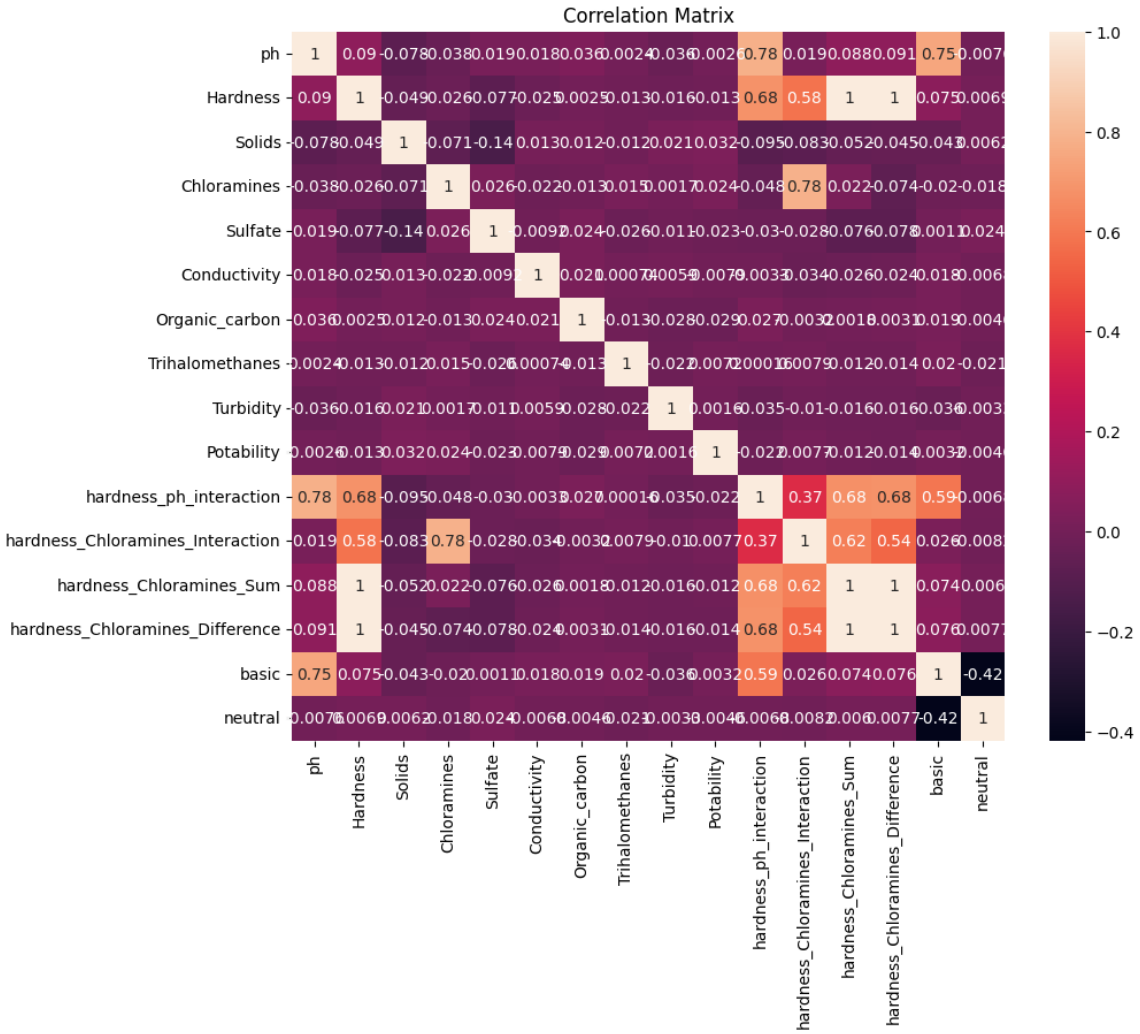
Perform visualizations, check distributions, and analyze correlations between features to uncover data patterns.

Distribution Of Each Column :





Correlation Matrix :



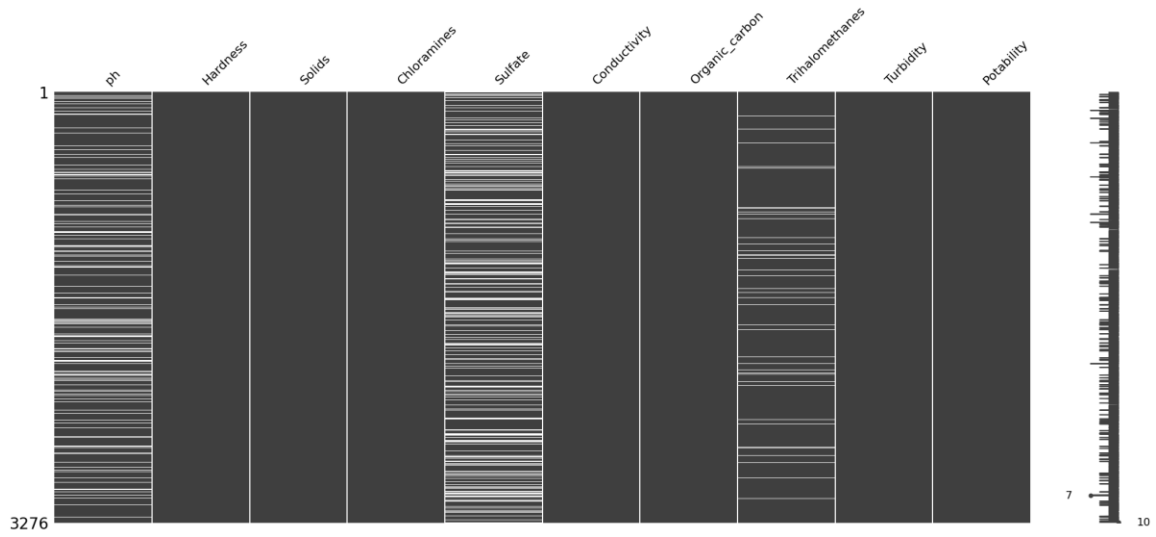
3. Data Preprocessing

Handling The Null Values:

```
1  df.isnull().sum()
```

✓ 0.0s

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



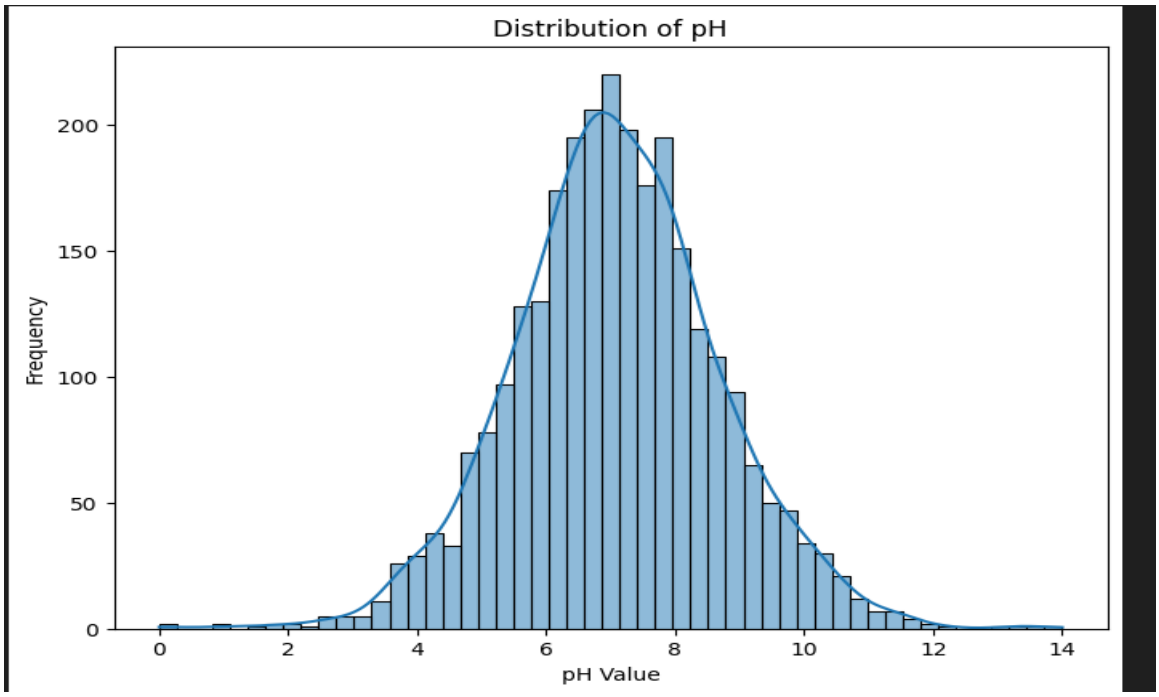
Filling The Null Values:

Fill The NUll Values By On Basis Of Potability (1 - 0) :

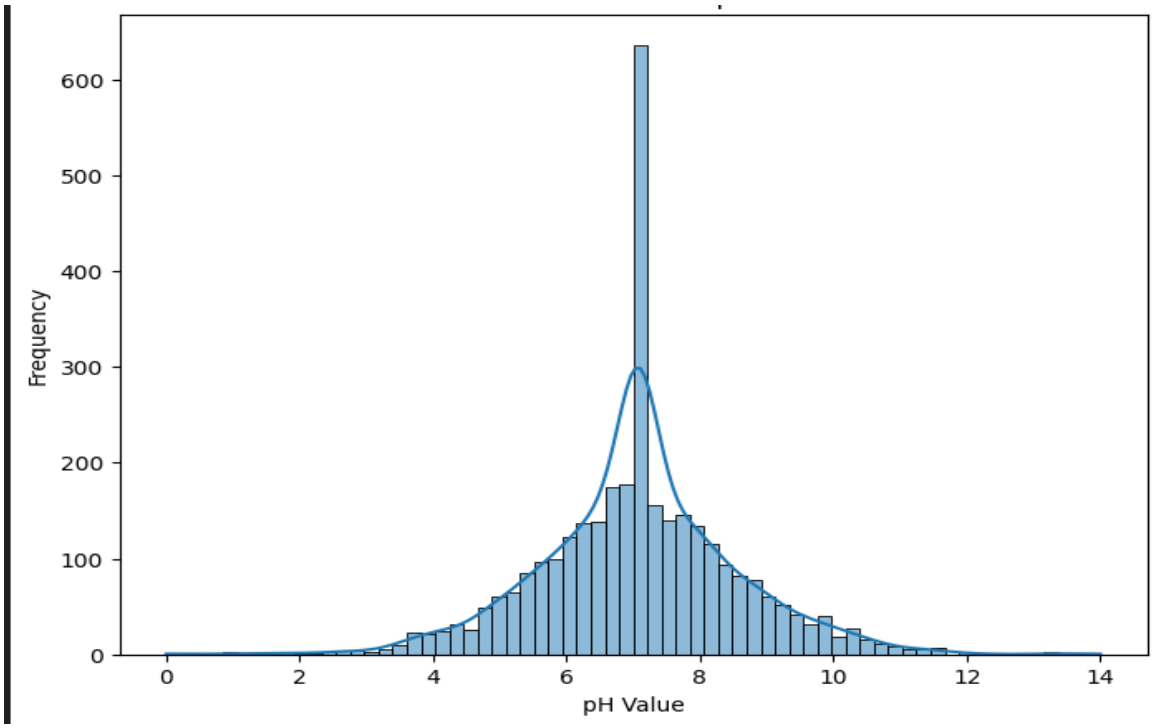
```
1  df['ph'] = df.groupby('Potability')['ph'].apply(lambda x: x.fillna(x.mean())).reset_index(level=0, drop=True)
2  df['Sulfate'] = df.groupby('Potability')['Sulfate'].apply(lambda x: x.fillna(x.mean())).reset_index(level=0, drop=True)
3  df['Trihalomethanes'] = df.groupby('Potability')['Trihalomethanes'].apply(lambda x: x.fillna(x.mean())).reset_index(level=0, drop=True)
4
```

✓ 0.0s

Distribution Of pH Before Computing Null Values:

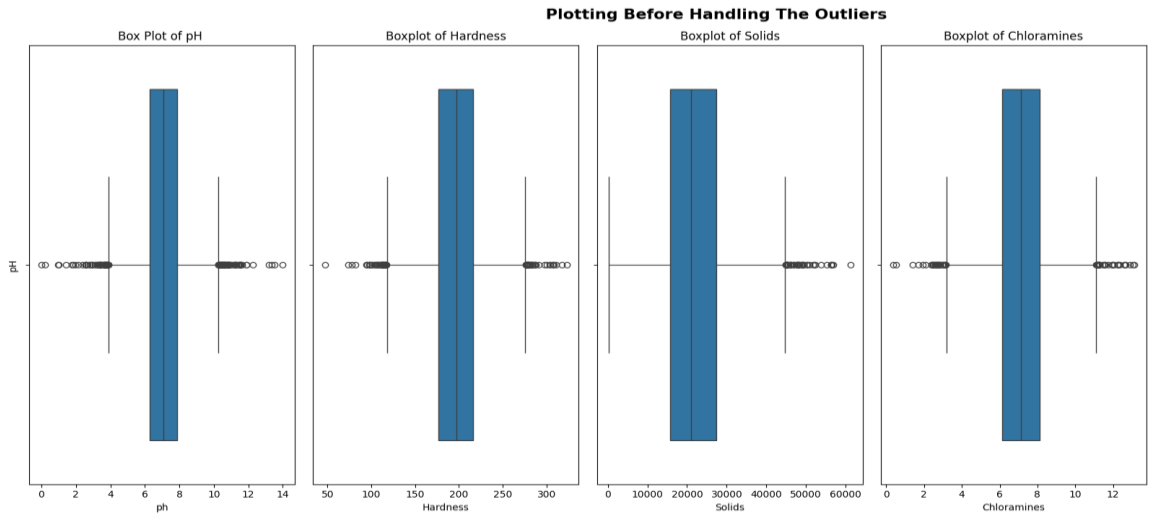


Distribution Of pH After Computing Null Values:



4. Outlier Detection

Identify and treat outliers using statistical method boxplots to ensure clean data for modeling.



Handling The Outliers

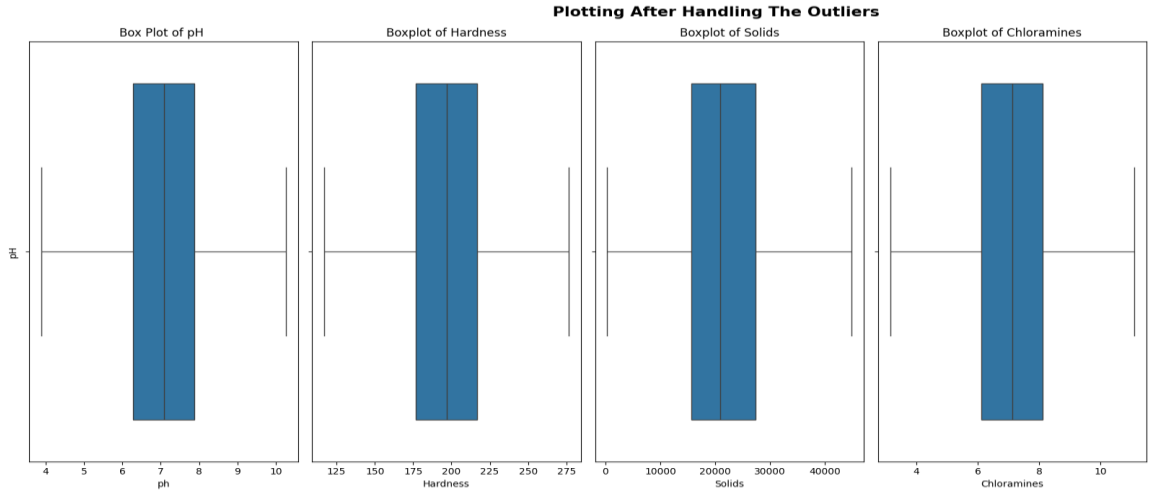
```python
def capOutliers(data, column):
    Q1 = data[column].quantile(0.25)
    Q3 = data[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_limit = Q1 - 1.5 * IQR
    upper_limit = Q3 + 1.5 * IQR

    #capp the outlier value
    data[column] = np.where(data[column] < lower_limit, lower_limit, data[column])
    data[column] = np.where(data[column] > upper_limit, upper_limit, data[column])

    return data

```

✓ 0.0s



5. Train/Test Split

Split the dataset into training and testing subsets to properly evaluate the model's performance.

Splitting The Data Into Train Test :

```
X = df.drop(columns=['Potability'])
y = df['Potability']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

✓ 0.0s

Scaling The Data :

Generate + Code + Markdown

```
scaler = StandardScaler()


X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

✓ 0.0s

```
model = RandomForestClassifier(random_state=42)
model.fit(X_train_scaled, y_train)
```

✓ 1.7s

RandomForestClassifier

RandomForestClassifier(random_state=42)

6. Model Building

Train different classification models, tune hyperparameters, and select the best-performing model.

I trained two models random-forest model and logiststic regression.

Random Forest:

```
Random Forest model Accuracy :  81.25
[[382  30]
 [ 93 151]]
              precision    recall  f1-score   support

           0       0.80      0.93      0.86       412
           1       0.83      0.62      0.71       244

    accuracy                           0.81       656
   macro avg       0.82      0.77      0.79       656
weighted avg       0.82      0.81      0.81       656
```

Logistic Regression:

```
Logistic Regression Accuracy: 63.41463414634146
Confusion Matrix:
[[408   4]
 [236   8]]
Classification Report:
              precision    recall  f1-score   support

           0       0.63      0.99      0.77       412
           1       0.67      0.03      0.06       244

    accuracy                           0.63       656
   macro avg       0.65      0.51      0.42       656
weighted avg       0.65      0.63      0.51       656
```

Using random forest model I get an accuracy of 81.25 but with logistic regression I get an accuracy of 69.4
so I choose random forest model and apply backend on my model.

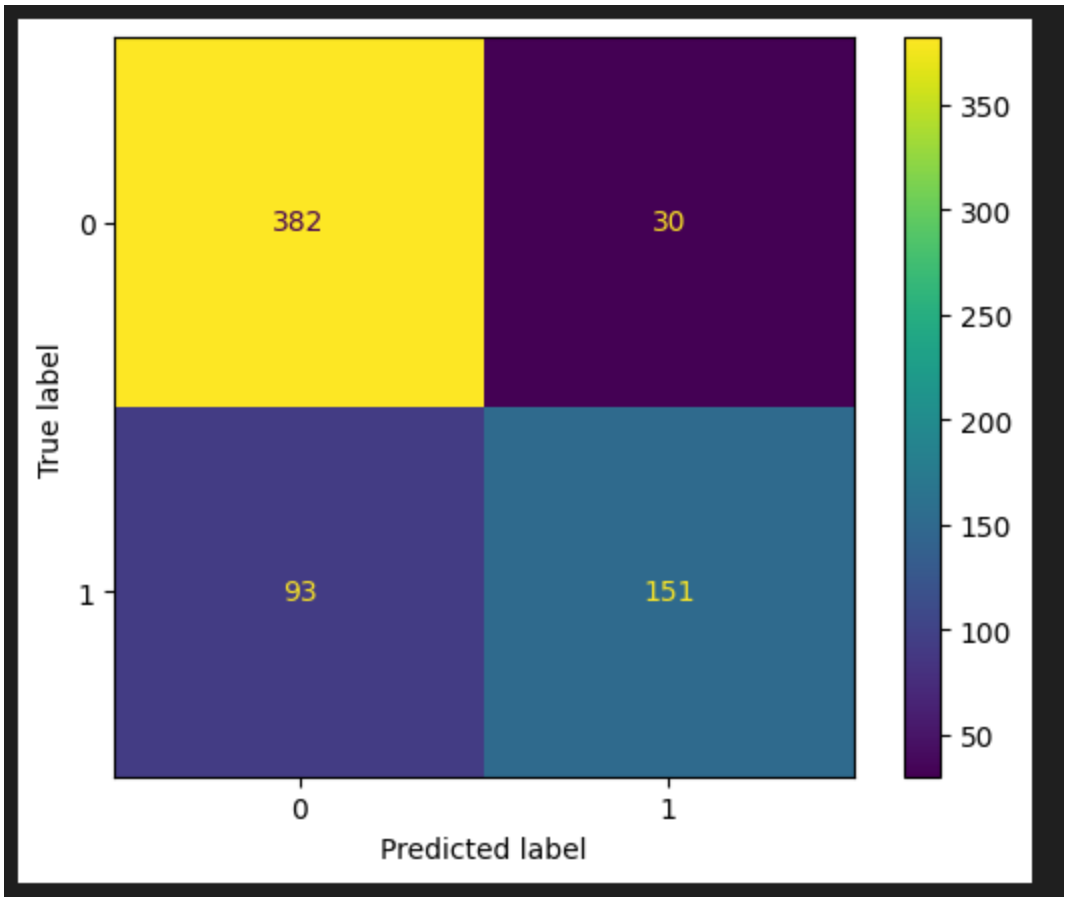
7. Model Evaluation

Evaluate models using Accuracy, F1-score, and Confusion Matrix to assess predictive performance.

```
Random Forest model Accuracy :  81.25
[[382  30]
 [ 93 151]]
              precision    recall  f1-score   support

           0       0.80      0.93      0.86       412
           1       0.83      0.62      0.71       244

    accuracy                           0.81       656
   macro avg       0.82      0.77      0.79       656
weighted avg       0.82      0.81      0.81       656
```



8. Saving the Model

Save the trained machine learning model into a .pkl file for later use in the backend.

Saving The MOdel Using Joblib:

```
import joblib

joblib.dump(model, 'water_potability_model.pkl')
```

✓ 0.1s

```
['water_potability_model.pkl']
```

```
joblib.dump(scaler, 'scaler.pkl')
```

✓ 0.0s

```
['scaler.pkl']
```

9. Backend Development (FastAPI)

Create a FastAPI server that loads the trained model and exposes a prediction API endpoint.

```
PS E:\mlthon\Competition> uvicorn api_app:app --reload
INFO:     Will watch for changes in these directories: ['E:\\mlthon\\Competition']
INFO:     Uvicorn running on http://127.0.0.1:8000 (Press CTRL+C to quit)
INFO:     Started reloader process [10876] using StatReload
INFO:     Started server process [1104]
INFO:     Waiting for application startup.
INFO:     Application startup complete.
```

```
PROBLEMS 1  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  QUERY RESULTS (PREVIEW)  JUPYTER  SQL HISTORY  TASK MONITOR
PS E:\mlthon\Competition> streamlit run api_app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8502
  Network URL: http://192.168.18.68:8502
```

10. Frontend Development (Streamlit)

Develop a simple web interface where users can input features and receive water potability predictions.

```
PS E:\mlthon\Competition> streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8504
  Network URL: http://192.168.18.68:8504
```

11. GitHub Repository Setup

Organize and upload all project files to GitHub, including notebook, backend, frontend, and model files.

https://github.com/SELENO-HAIDER/WaterPotabilityPrediction/blob/main/hello.py

12. Demo Video Recording

Record a short video explaining the project flow, model usage, frontend interaction, and API working.