Water Quality Analysis

Project Description:

Phase 4: Development Part 2 - Data Visualization and Predictive Modeling

In the fourth phase of the "Water Quality Analysis" project, we will continue building upon the foundation established in the earlier phases. We've already collected and preprocessed water quality data in Phase 1, performed exploratory data analysis in Phase 2, and in Phase 3, we laid the groundwork for data visualization and predictive modeling. Phase 4 will encompass two major components: data visualization and the development of a predictive model.

1. Data Visualization:

Visualizations with Matplotlib and Seaborn: We will utilize powerful Python libraries such as Matplotlib and Seaborn to create a variety of visualizations that offer insights into the water quality dataset. These visualizations will include:

Histograms: Visualize the distribution of each water quality parameter, helping us understand their frequency and range.

Scatter Plots: Explore relationships between pairs of parameters, uncovering potential correlations or patterns.

Correlation Matrices: Create correlation matrices to quantify the relationships between different water quality parameters.

Insights from Visualizations: Our goal is to gain a deeper understanding of the dataset and to identify any interesting trends, anomalies, or patterns that may exist. These insights will guide our subsequent work in building a predictive model.

2. Predictive Modeling for Water Potability

Selecting Machine Learning Algorithms We will employ machine learning techniques to build a predictive model. Potential algorithms may include:

Logistic Regression: A fundamental algorithm for binary classification, we will evaluate its effectiveness in predicting water potability.

Random Forest: A more complex ensemble learning method, known for its robustness and ability to handle complex relationships in the data.

Feature Engineering: We will consider feature engineering techniques to improve the model's performance. This may involve selecting relevant features, transforming data, or creating new features that can enhance the predictive power of the model.

Model Evaluation: To assess the predictive model, we will employ various evaluation metrics such as accuracy, precision, recall, and F1-score. Additionally, we will use techniques like cross-validation to ensure the model's generalizability.

Hyperparameter Tuning: If using algorithms with hyperparameters, we will fine-tune these parameters to optimize model performance.

Interpreting Results: Once the model is developed, we will interpret the results and understand which water quality parameters are most influential in determining water potability. This insight can be crucial for future decision-making.

Project Milestones for Phase 4:

Create a diverse set of data visualizations to gain insights into the dataset.

- Build, train, and evaluate predictive models for water potability.

- Fine-tune models for optimal performance.

- Document findings and insights from the analysis and modeling.

The completion of Phase 4 marks a significant step forward in our water quality analysis project, as it equips us with the tools to predict water probability based on water quality parameters. This predictive capability can be of great value for ensuring safe and clean water sources. Throughout this phase, we emphasize the importance of thorough documentation to facilitate the sharing of results and insights with stakeholders and peers.

```
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import numpy as np

data = pd.read_csv("/content/water_potability.csv")
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

```
data = data.dropna()
data.isnull().sum()
```

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

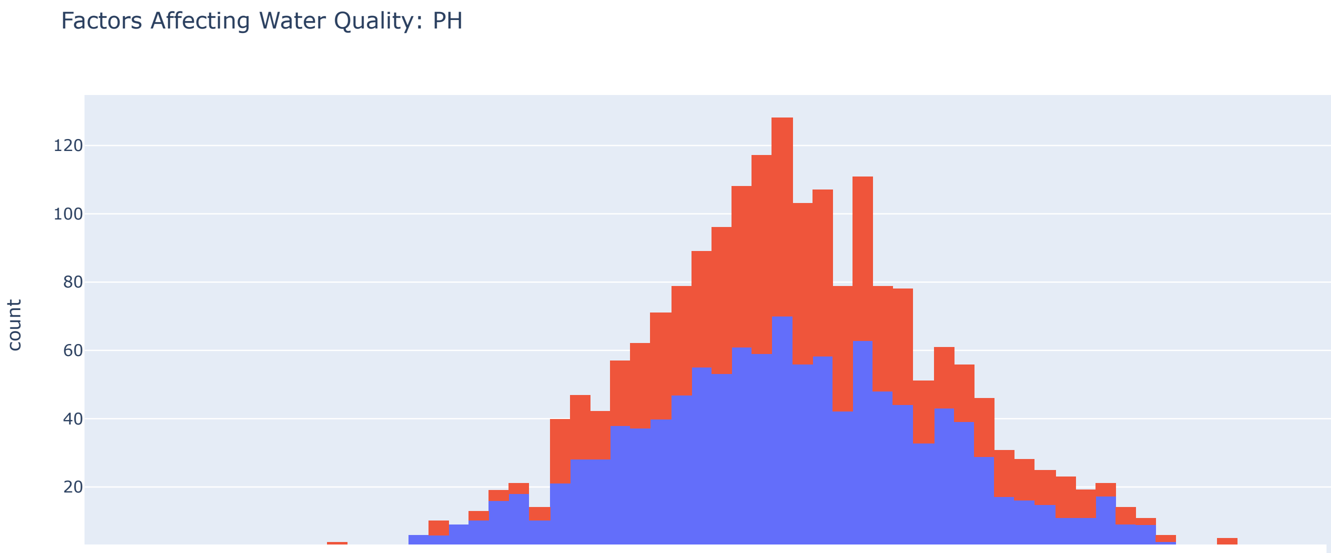
```
plt.figure(figsize=(15, 10))
sns.countplot(data.Potability)
plt.title("Distribution of Unsafe and Safe Water")
plt.show()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   3801             try:
-> 3802                 return self._engine.get_loc(casted_key)
   3803             except KeyError as err:

8 frames

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.Int64HashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.Int64HashTable.get_item()

KeyError: 0

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self, key, method, tolerance)
   3802                 return self._engine.get_loc(casted_key)
   3803             except KeyError as err:
-> 3804                 raise KeyError(key) from err
   3805             except TypeError:
   3806                 # If we have a listlike key, _check_indexing_error will raise

KeyError: 0
```

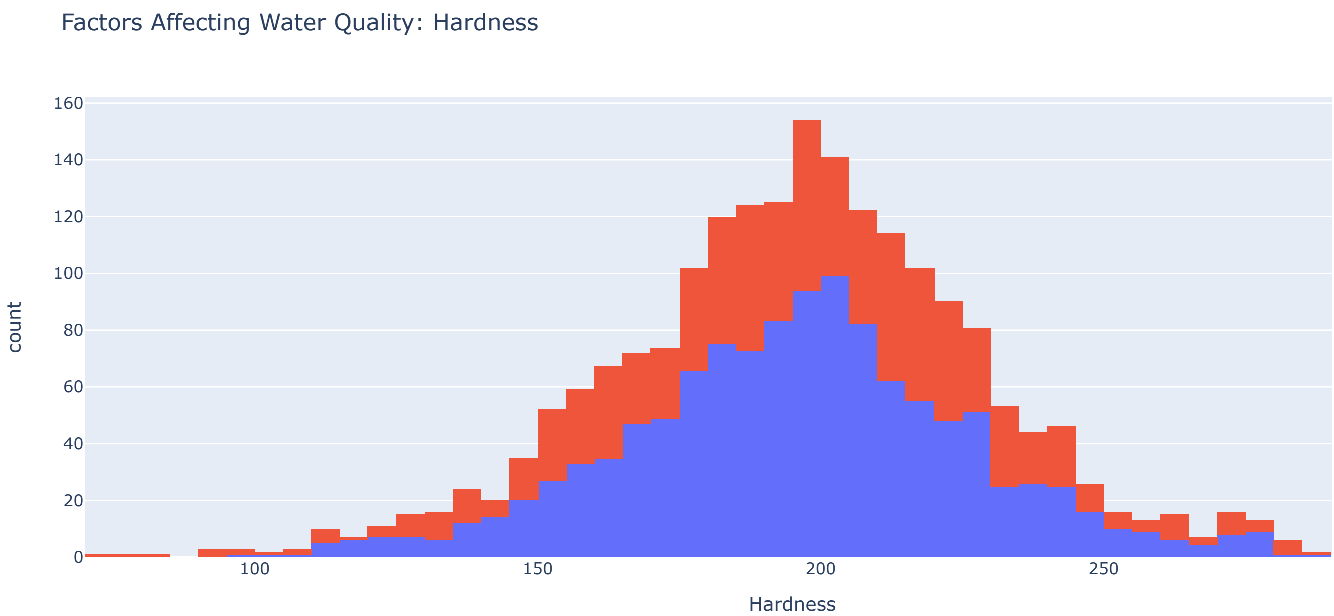
SEARCH STACK OVERFLOW

```
<Figure size 1500x1000 with 0 Axes>
```

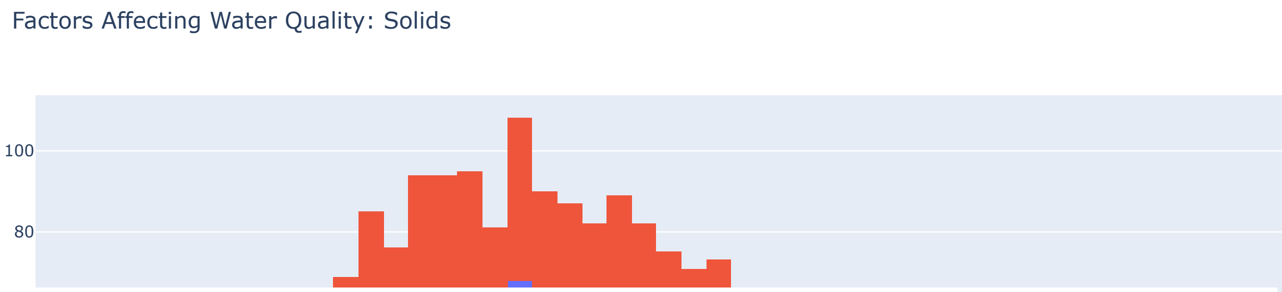
```
import plotly.express as px
data = data
figure = px.histogram(data, x = "ph",
                      color = "Potability",
                      title= "Factors Affecting Water Quality: PH")
figure.show()
```



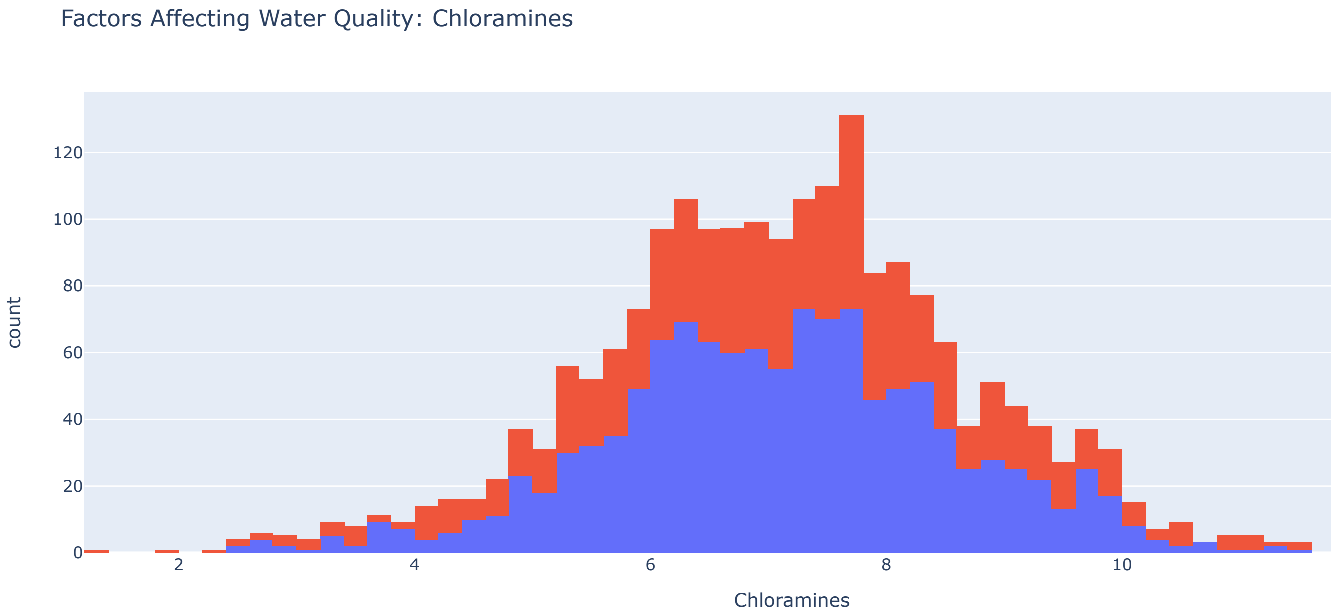

```
import plotly.express as px
data = data
figure = px.histogram(data, x = "Hardness",
                      color = "Potability",
                      title= "Factors Affecting Water Quality: Hardness")
figure.show()
```




```
import plotly.express as px
data = data
figure = px.histogram(data, x = "Solids",
                      color = "Potability",
                      title= "Factors Affecting Water Quality: Solids")
figure.show()
```




```
import plotly.express as px
data = data
figure = px.histogram(data, x = "Chloramines",
                      color = "Potability",
                      title= "Factors Affecting Water Quality: Chloramines")
figure.show()
```

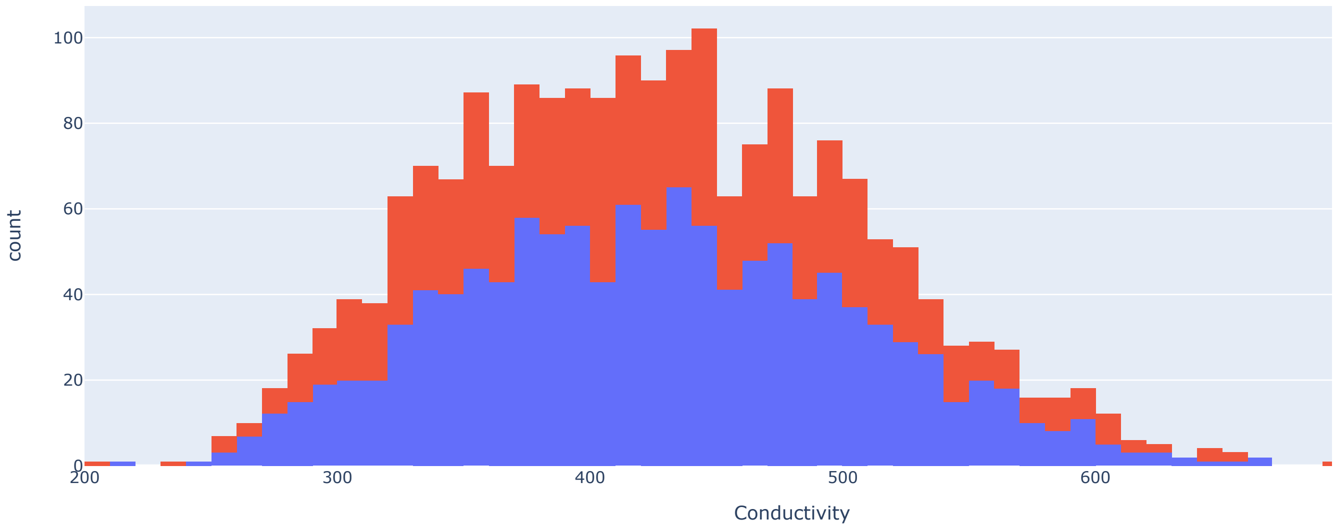



```
import plotly.express as px
data = data
figure = px.histogram(data, x = "Sulfate",
                      color = "Potability",
                      title= "Factors Affecting Water Quality: Sulfate")
figure.show()
```

Factors Affecting Water Quality: Sulfate

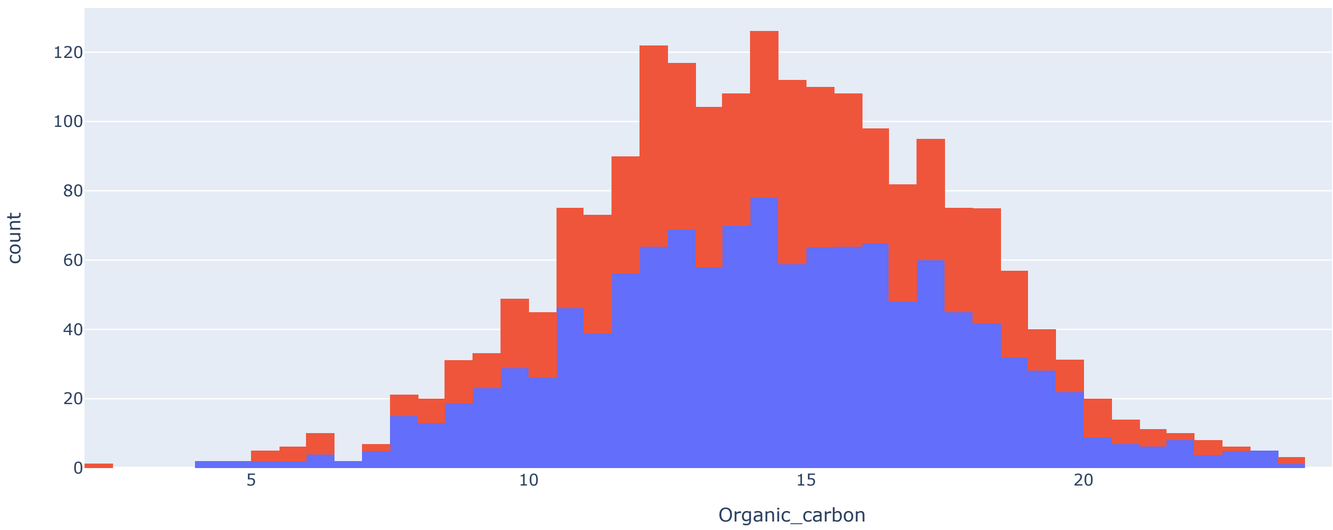
```
import plotly.express as px
data = data
figure = px.histogram(data, x = "Conductivity",
                      color = "Potability",
                      title= "Factors Affecting Water Quality: Conductivity")
figure.show()
```

Factors Affecting Water Quality: Conductivity

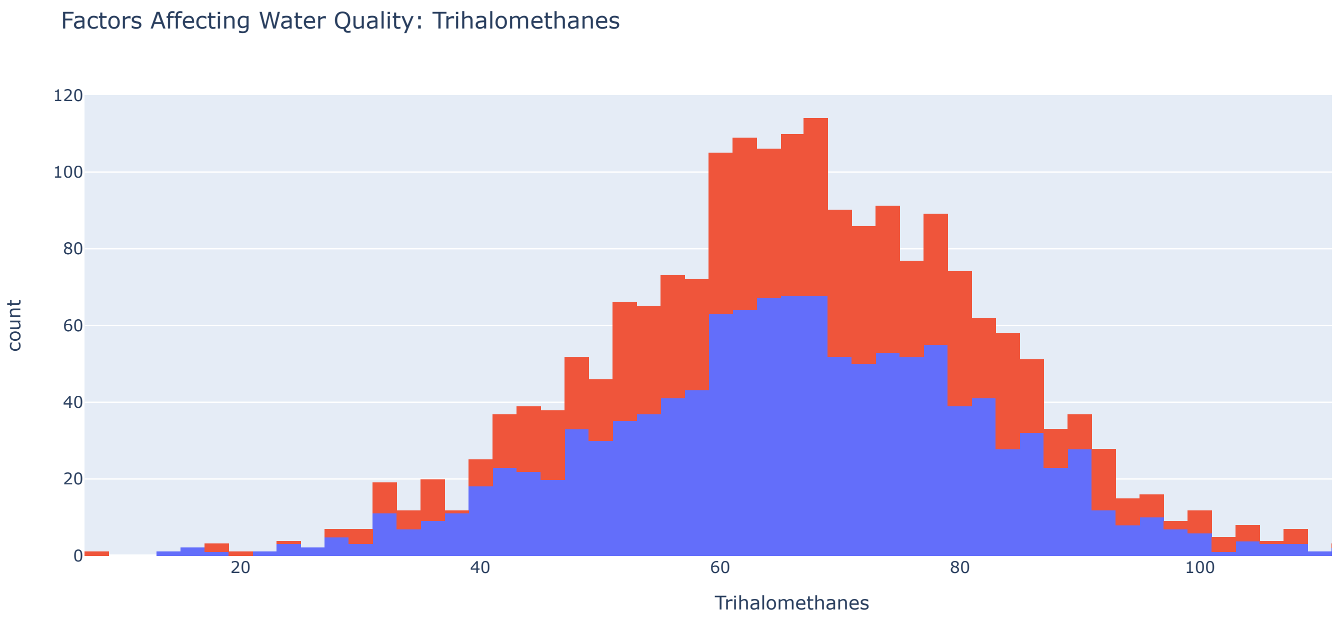



```
import plotly.express as px
data = data
figure = px.histogram(data, x = "Organic_carbon",
                      color = "Potability",
                      title= "Factors Affecting Water Quality: Organic Carbon")
figure.show()
```

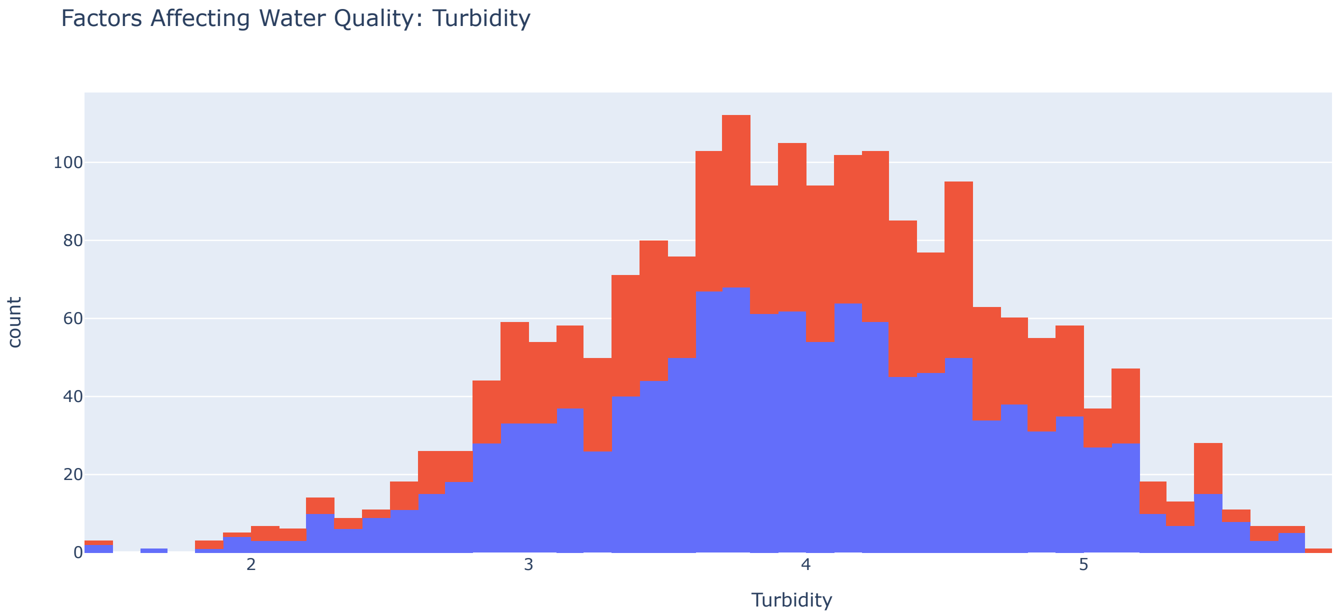
Factors Affecting Water Quality: Organic Carbon




```
import plotly.express as px
data = data
figure = px.histogram(data, x = "Trihalomethanes",
                      color = "Potability",
                      title= "Factors Affecting Water Quality: Trihalomethanes")
figure.show()
```




```
import plotly.express as px
data = data
figure = px.histogram(data, x = "Turbidity",
                      color = "Potability",
                      title= "Factors Affecting Water Quality: Turbidity")
figure.show()
```




```
pip install pycaret
```

```
Collecting pycaret
  Downloading pycaret-3.1.0-py3-none-any.whl (483 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 483.9/483.9 kB 7.0 MB/s eta 0:00:00
Requirement already satisfied: ipython>=5.5.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (7.34.0)
Requirement already satisfied: ipywidgets>=7.6.5 in /usr/local/lib/python3.10/dist-packages (from pycaret) (7.7.1)
Requirement already satisfied: tqdm>=4.62.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (4.66.1)
Requirement already satisfied: numpy<1.24,>=1.21 in /usr/local/lib/python3.10/dist-packages (from pycaret) (1.23.5)
Requirement already satisfied: pandas<2.0.0,>=1.3.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (1.5.3)
Requirement already satisfied: jinja2>=1.2 in /usr/local/lib/python3.10/dist-packages (from pycaret) (3.1.2)
Collecting scipy~=1.10.1 (from pycaret)
  Downloading scipy-1.10.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (34.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 34.4/34.4 MB 44.9 MB/s eta 0:00:00
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (1.3.2)
Requirement already satisfied: scikit-learn<1.3.0,>=1.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (1.2.2)
Collecting pyod>=1.0.8 (from pycaret)
  Downloading pyod-1.1.0.tar.gz (153 kB)
```

```
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 153.4/153.4 kB 20.4 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
Requirement already satisfied: imbalanced-learn>=0.8.1 in /usr/local/lib/python3.10/dist-packages (from pycaret) (0.10.1)
Collecting category-encoders>=2.4.0 (from pycaret)
  Downloading category_encoders-2.6.2-py2.py3-none-any.whl (81 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 81.8/81.8 kB 12.0 MB/s eta 0:00:00
Requirement already satisfied: lightgbm>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (4.0.0)
Requirement already satisfied: numba>=0.55.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (0.56.4)
Requirement already satisfied: requests>=2.27.1 in /usr/local/lib/python3.10/dist-packages (from pycaret) (2.31.0)
Requirement already satisfied: psutil>=5.9.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (5.9.5)
Requirement already satisfied: markupsafe>=2.0.1 in /usr/local/lib/python3.10/dist-packages (from pycaret) (2.1.3)
Requirement already satisfied: importlib-metadata>=4.12.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (6.8.0)
Requirement already satisfied: nbformat>=4.2.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (5.9.2)
Requirement already satisfied: cloudpickle in /usr/local/lib/python3.10/dist-packages (from pycaret) (2.2.1)
Collecting deprecation>=2.1.0 (from pycaret)
  Downloading deprecation-2.1.0-py2.py3-none-any.whl (11 kB)
Requirement already satisfied: xxhash in /usr/local/lib/python3.10/dist-packages (from pycaret) (3.4.1)
Requirement already satisfied: matplotlib>=3.3.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (3.7.1)
Collecting scikit-plot>=0.3.7 (from pycaret)
  Downloading scikit_plot-0.3.7-py3-none-any.whl (33 kB)
Requirement already satisfied: yellowbrick>=1.4 in /usr/local/lib/python3.10/dist-packages (from pycaret) (1.5)
Requirement already satisfied: plotly>=5.0.0 in /usr/local/lib/python3.10/dist-packages (from pycaret) (5.15.0)
Collecting kaleido>=0.2.1 (from pycaret)
  Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl (79.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.9/79.9 MB 9.6 MB/s eta 0:00:00
Collecting schemdraw==0.15 (from pycaret)
  Downloading schemdraw-0.15-py3-none-any.whl (106 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 106.8/106.8 kB 13.7 MB/s eta 0:00:00
Collecting plotly-resampler>=0.8.3.1 (from pycaret)
  Downloading plotly_resampler-0.9.1-py3-none-any.whl (73 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 73.4/73.4 kB 8.7 MB/s eta 0:00:00
Requirement already satisfied: statsmodels>=0.12.1 in /usr/local/lib/python3.10/dist-packages (from pycaret) (0.14.0)
Collecting sktime!=0.17.1,!=0.17.2,!=0.18.0,<0.22.0,>=0.16.1 (from pycaret)
  Downloading sktime-0.21.1-py3-none-any.whl (17.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17.1/17.1 MB 83.6 MB/s eta 0:00:00
Collecting tbats>=1.1.3 (from pycaret)
  Downloading tbats-1.1.3-py3-none-any.whl (44 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44.0/44.0 kB 4.6 MB/s eta 0:00:00
Collecting pmdarima!=1.8.1,<3.0.0,>=1.8.0 (from pycaret)
  Downloading pmdarima-2.0.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl (2.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 85.1 MB/s eta 0:00:00
```

```
import pycaret
```

```
correlation = data.corr()
correlation["ph"].sort_values(ascending=False)
```

```
ph                 1.000000
Hardness           0.108948
Organic_carbon     0.028375
Trihalomethanes    0.018278
Potability         0.014530
Conductivity       0.014128
Sulfate            0.010524
Chloramines       -0.024768
Turbidity         -0.035849
Solids            -0.087615
Name: ph, dtype: float64
```

```
from pycaret.classification import *
clf = setup(data, target = "Potability",  session_id = 786)
compare_models()
```

	Description	Value
0	Session id	786
1	Target	Potability
2	Target type	Binary
3	Original data shape	(2011, 10)
4	Transformed data shape	(2011, 10)
5	Transformed train set shape	(1407, 10)
6	Transformed test set shape	(604, 10)
7	Numeric features	9
8	Preprocess	True
9	Imputation type	simple
10	Numeric imputation	mean
11	Categorical imputation	mode
12	Fold Generator	StratifiedKFold
13	Fold Number	10
14	CPU Jobs	-1
15	Use GPU	False
16	Log Experiment	False
17	Experiment Name	clf-default-name
18	USI	f0c1

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
et	Extra Trees Classifier	0.6802	0.6956	0.3952	0.6778	0.4977	0.2870	0.3100	0.4080
rf	Random Forest Classifier	0.6780	0.6844	0.4040	0.6696	0.5024	0.2854	0.3063	0.6920
qda	Quadratic Discriminant Analysis	0.6745	0.7091	0.3866	0.6795	0.4879	0.2746	0.3013	0.0270
gbc	Gradient Boosting Classifier	0.6489	0.6554	0.3581	0.6232	0.4505	0.2186	0.2397	0.3920
lightgbm	Light Gradient Boosting Machine	0.6432	0.6658	0.4869	0.5719	0.5232	0.2416	0.2453	0.4140
xgboost	Extreme Gradient Boosting	0.6333	0.6677	0.4729	0.5540	0.5074	0.2193	0.2224	0.3190
nb	Naive Bayes	0.6212	0.6280	0.2506	0.5728	0.3474	0.1344	0.1581	0.0270
ridge	Ridge Classifier	0.5984	0.0000	0.0282	0.6267	0.0534	0.0137	0.0499	0.0450
lda	Linear Discriminant Analysis	0.5970	0.5189	0.0299	0.5867	0.0564	0.0115	0.0421	0.0270
dummy	Dummy Classifier	0.5970	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0220
dt	Decision Tree Classifier	0.5956	0.5784	0.4902	0.4981	0.4927	0.1570	0.1576	0.0350
lr	Logistic Regression	0.5949	0.4964	0.0053	0.1500	0.0102	-0.0022	-0.0138	1.0540
ada	Ada Boost Classifier	0.5949	0.5823	0.3087	0.4993	0.3796	0.1034	0.1109	0.1740
knn	K Neighbors Classifier	0.5423	0.5226	0.3262	0.4122	0.3625	0.0145	0.0145	0.0420
svm	SVM - Linear Kernel	0.4789	0.0000	0.5982	0.2408	0.3434	-0.0014	-0.0104	0.0430

ExtraTreesClassifier

```
model = create_model("rf")
predict = predict_model(model, data=data)
predict.head()
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.6596	0.6720	0.3684	0.6364	0.4667	0.2419	0.2614
1	0.6809	0.7256	0.3684	0.7000	0.4828	0.2828	0.3133
2	0.7163	0.6705	0.4211	0.7742	0.5455	0.3644	0.4002
3	0.7021	0.6919	0.4386	0.7143	0.5435	0.3407	0.3630
4	0.6383	0.6312	0.4035	0.5750	0.4742	0.2113	0.2190
5	0.6454	0.6917	0.3509	0.6061	0.4444	0.2103	0.2273
6	0.7092	0.7448	0.4035	0.7667	0.5287	0.3466	0.3839
7	0.6500	0.6197	0.3750	0.6000	0.4615	0.2222	0.2357
8	0.7000	0.7027	0.5000	0.6667	0.5714	0.3478	0.3563
9	0.6786	0.6937	0.4107	0.6571	0.5055	0.2857	0.3030
Mean	0.6780	0.6844	0.4040	0.6696	0.5024	0.2854	0.3063
Std	0.0270	0.0364	0.0412	0.0651	0.0406	0.0583	0.0644

0	Random Forest Classifier	0.8951	0.9681	0.8089	0.9213	0.8615	0.7776	0.7819

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_
3	8.316766	214.373398	22018.417969	8.059333	356.886139	363.266510	18
4	9.092223	181.101517	17978.986328	6.546600	310.135742	398.410828	11
5	5.584086	188.313324	28748.687500	7.544869	326.678375	280.467926	8
6	10.223862	248.071732	28749.716797	7.513409	393.663391	283.651642	13
7	8.635849	203.361526	13672.091797	4.563009	303.309784	474.607635	12