Import pandas as pd

Import numpy as np

Import matplotlib.pyplot as plt

Import plotly

Import plotly.express as px

Import plotly.graph_objs as go

Import plotly.offline as py

From plotly.offline import iplot

From plotly.subplots import make_subplots

Import plotly.figure_factory as ff

Import warnings

Warnings.filterwarnings(“ignore”)

Add Codeadd Markdown

Exploratory Data Analysis (EDA)

Add Codeadd Markdown

Import the CSV file

Df = pd.read_csv(‘../input/water-potability/water_potability.csv’)

Df.head()

Add Codeadd Markdown

Dimensions of the Dataset

Df.shape

Add Codeadd Markdown

Data Info

Df.info()

Add Codeadd Markdown

Data Description

Df.describe()

Add Codeadd Markdown

Skewness

Df.drop(['Potability'], axis=1).skew()

Add Codeadd Markdown

Plot Histogram for each feature

Df.drop(['Potability'], axis=1).hist(figsize=(18,10));

Add Codeadd Markdown

Correlation between each feature

Df.corr()

Add Codeadd Markdown

Heat Map of correlation between each feature

Fig = go.Figure(go.Heatmap(x=df.corr().columns.tolist(), y=df.corr().columns.tolist(), z=df.corr(), colorscale='bluered'))

Fig.show()

Expand

Add Codeadd Markdown

Target Variable

Df['Potability'].value_counts(normalize=True)

Expand

Add Codeadd Markdown

```
Fig = px.bar(df, x=['Not Potable','Potable'], y=df['Potability'].value_counts(normalize=True), title="Target Variable Value Count")

Fig.show()
```

Expand

Add Codeadd Markdown

```
# Missing Values


Df.isnull().sum()
```

Expand

Add Codeadd Markdown

```
Df.isnull().mean().plot.bar(figsize=(12,6))

Plt.ylabel('Percentage of missing values')

Plt.xlabel('Features')

Plt.title('Missing Data in Percentages');
```

Expand

Add Codeadd Markdown

```
# Use KNN Imputer to impute NaN Values


From sklearn.impute import KNNImputer


Imputer = KNNImputer()

Df[['ph','Sulfate','Trihalomethanes']] = imputer.fit_transform(df[['ph','Sulfate','Trihalomethanes']])
```

Add Codeadd Markdown

```
# Checking for Missing Values after Imputing


Df.isnull().sum()
```

Expand

Add Codeadd Markdown

Df.head()

Add Codeadd Markdown

Modelling

Add Codeadd Markdown

Split the Dataset into Training and Test Datasets

```
From sklearn.model_selection import train_test_split

X = df.drop(['Potability'],axis=1)
Y = df['Potability']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state =42)
```

Add Codeadd Markdown

Machine Learning Algorithms Params

Add Codeadd Markdown

Random Forest Params

Add Codeadd Markdown

```
From sklearn.ensemble import RandomForestClassifier

Rf_params = {'n_estimators':[100, 200, 300],
        'criterion':['gini','entropy'],
        'min_samples_split':[3,5,10],
        'min_samples_leaf':[1,3,5],
        'max_features':['auto','sqrt','log2'],
        'bootstrap':[False]
       }
```

Add Codeadd Markdown

Gradient Boosting Classifier Params

Add Codeadd Markdown

```
From sklearn.ensemble import GradientBoostingClassifier


Gb_params = {'loss':['deviance', 'exponential'],
        'learning_rate':[0.001,0.01,0.1,0.5],
        'n_estimators':[100,300,500],
        'min_samples_split':[3,5,10],
        'max_depth':[3,5,10],
        'max_features':['auto','sqrt','log2']
       }
```

Add Codeadd Markdown

Multi-layer Perceptron (MLP) Classifier Params

Add Codeadd Markdown

```
From sklearn.neural_network import MLPClassifier


Mlp_params = {'hidden_layer_sizes':[(100,50,10),(100,100,100),(100,100),(5,5),(5,3,2)],
         'activation':['identity', 'logistic', 'tanh', 'relu'],
         'solver':['lbfgs','sgd','adam'],
         'alpha':[0.1,0.01,0.005,0.0001,0.00001],
         'learning_rate':['constant','invscaling','adaptive']
        }
```

Add Codeadd Markdown

KNeighbors Classifier Params

Add Codeadd Markdown

```
From sklearn.neighbors import KNeighborsClassifier


Kn_params = {'n_neighbors':[3,5,10],
        'weights':['uniform','distance'],
        'algorithm':['auto','ball_tree','kd_tree','brute'],
```

```
        'p':[1,2,5]
       }
```

Add Codeadd Markdown

XGBoost Params

Add Codeadd Markdown

```
From xgboost import XGBClassifier
Xg_params = {'learning_rate':[0.0001,0.001,0.01,0.1,0.3],
        'min_split_loss':[2,5,10],
        'max_depth':[3,5,10],
        'n_estimators':[100,300,500],
        'tree_method':['auto','exact','approx','hist','gpu_hist']
       }
```

Add Codeadd Markdown

C-Support Vector Classifier (SVC) Params

Add Codeadd Markdown

```
From sklearn.svm import SVC

Svc_params = {'C':[0.001,0.01,0.1,0.3],
         'kernel':['linear','poly','rbf','sigmoid'],
         'degree':[3,5,10],
         'gamma':['auto','scale']
        }
```

Add Codeadd Markdown

AdaBoost Classifier Params

Add Codeadd Markdown

```
From sklearn.ensemble import AdaBoostClassifier

Ada_params = {'n_estimators':[50,100,200,300],
         'learning_rate':[0.0001,0.001,0.01,0.1,0.3]
```

```
        }
```

Add Codeadd Markdown

Logistic Regression Params

Add Codeadd Markdown

```
From sklearn.linear_model import LogisticRegression


Lg_params = {'penalty':['l1','l2','elasticnet'],
        'dual':[True, False],
        'C':[0.001,0.01,0.1,0.3],
        'solver':['newton-cg','lbfgs','sag','saga'],
        'max_iter':[100,200,300],
        'l1_ratio':[0.001,0.01,0.1,0.3,0.8]
        }
```

Add Codeadd Markdown

Decision Tree Classifier Params

Add Codeadd Markdown

```
From sklearn.tree import DecisionTreeClassifier


Dt_params = {'criterion':['gini','entropy'],
        'splitter':['best','random'],
        'max_depth':[3,5,10],
        'min_samples_split':[2,5,10],
        'min_samples_leaf':[1,3,5,10],
        'max_features':['auto','sqrt','log2']
       }
```

Add Codeadd Markdown

```
# Grid Search CV to find Optimized model


Models = [RandomForestClassifier(),
```

```
        GradientBoostingClassifier(),
        MLPClassifier(),
        KNeighborsClassifier(),
        XGBClassifier(),
        SVC(),
        AdaBoostClassifier(),
        LogisticRegression(),
        DecisionTreeClassifier()
       ]

Model_params = [rf_params,
                Gb_params,
                Mlp_params,
                Kn_params,
                Xg_params,
                Svc_params,
                Ada_params,
                Lg_params,
                Dt_params
               ]
```

Add Codeadd Markdown

Standard Scaling

Add Codeadd Markdown

```
From sklearn.preprocessing import StandardScaler

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

Add Codeadd Markdown

Grid Search CV

Add Codeadd Markdown

```
From sklearn.model_selection import StratifiedKFold, GridSearchCV


#cross validation results
Cv_results = []


#to use in ensemble modeling
Best_estimators = []


For i in range(len(models)):
   Clf = GridSearchCV(models[i],
                      Param_grid = model_params[i],
                      Cv = StratifiedKFold(n_splits=2),
                      Scoring = 'roc_auc',
                      N_jobs = -1,
                      Verbose=1)


   Clf.fit(X_train_scaled,y_train)
   Cv_results.append(clf.best_score_)
   Best_estimators.append(clf.best_estimator_)
   Print('Method: {}  Score: {}'.format(models[i],cv_results[i]))




Results = pd.DataFrame({'Cross Validation Means':cv_results,
                      'ML Models':[
                       'Random Forest',
                       'Gradient Boosting',
```

```
                'MLP Classifier',
                'KNeighbors Classifier',
                'XGBoost',
                'SVC',
                'AdaBoost',
                'Logistic Regression',
                'Decision Tree'
               ]})
```

Add Codeadd Markdown

Results

Add Codeadd Markdown

Ensemble Model

Add Codeadd Markdown

```
From sklearn.ensemble import VotingClassifier

Voting = VotingClassifier(estimators=[('rf',best_estimators[0]),
                                      ('gb',best_estimators[1]),
                                      ('mlp',best_estimators[2]),
                                      ('kn',best_estimators[3]),
('xgb',best_estimators[4]),
                                      ('svc',best_estimators[5]),
                                      ('adb',best_estimators[6]),
                                      ('lg',best_estimators[7]),
                                      ('dt',best_estimators[8]),
                                     ],
                          Voting='hard',
                          N_jobs= -1)
```

Add Codeadd Markdown

From sklearn.metrics import accuracy_score

Voting = voting.fit(X_train_scaled,y_train)

My_score = accuracy_score(voting.predict(X_test_scaled),y_test)

Print(my_score)

Add Codeadd Markdown

Output:

STATION CODE LOCATIONS STATE Temp D.O. (mg/l) PH CONDUCTIVITY (µmhos/cm) B.O.D. (mg/l) NITRATENAN N+ NITRITENANN (mg/l) FECAL COLIFORM (MPN/100ml) TOTAL COLIFORM (MPN/100ml)Mean year

0 1393 DAMANGANGA AT D/S OF MADHUBAN, DAMAN DAMAN & DIU 30.6 6.7 7.5 203 NAN 0.1 11 27 2014

1 1 1399 ZUARI AT D/S OF PT. WHERE KUMBARJRIA CANAL JOI... GOA 29.8 5.7 7.2 189 2 0.2 4953 8391 2014

2 1475 ZUARI AT PANCHAWADI GOA 29.5 6.3 6.9 179 1.7 0.1 3243 5330 2014

3 3 3181 RIVER ZUARI AT BORIM BRIDGE GOA 29.7 5.8 6.9 64 3.8 0.5 5382 8443 2014

4 3182 RIVER ZUARI AT MARCAIM JETTY GOA 29.5 5.8 7.3 83 1.9 0.4 3428 5500 2014