TRAIN THE MODEL

TEAM ID:	PNT2022TMID32788
TEAM MEMBERS:	ROSHANA V S SNEKA M S SRAVANI SOWMYA SHRI J SNEHA K
PROJECT:	EFFICIENT WATER QUALITY ANALYSIS AND PREDICTION USING MACHINE LEARNING

Importing libraries

In [1]:

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

In [2]:

```
!pip install ibm_watson_machine_learning
```

```
Requirement already satisfied: ibm_watson_machine_learning in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (1.0.257)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (1.26.7)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (0.8.9)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (0.3.3)
Requirement already satisfied: packaging in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (21.3)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (2.26.0)
Requirement already satisfied: ibm-cos-sdk==2.11.* in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (2.11.
0)
Requirement already satisfied: importlib-metadata in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (4.8.2)
Requirement already satisfied: pandas<1.5.0,>=0.24.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (1.3.
4)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm_watson_machine_learning) (2022.9.24)
Requirement already satisfied: ibm-cos-sdk-s3transfer==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm_
watson_machine_learning) (2.11.0)
Requirement already satisfied: ibm-cos-sdk-core==2.11.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm_watson
_machine_learning) (2.11.0)
Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk==2.11.*->ibm_watson_m
achine_learning) (0.10.0)
Requirement already satisfied: python-dateutil<3.0.0,>=2.1 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from ibm-cos-sdk-core==2.11.0->ib
m-cos-sdk==2.11.*->ibm_watson_machine_learning) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas<1.5.0,>=0.24.2->ibm_watson_machine_l
earning) (2021.3)
Requirement already satisfied: numpy>=1.17.3 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from pandas<1.5.0,>=0.24.2->ibm_watson_machine_
learning) (1.20.3)
Requirement already satisfied: six>=1.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from python-dateutil<3.0.0,>=2.1->ibm-cos-sdk-core==
2.11.0->ibm-cos-sdk==2.11.*->ibm_watson_machine_learning) (1.15.0)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->ibm_watson_machine_learning) (3.
3)
Requirement already satisfied: charset-normalizer~=2.0.0 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from requests->ibm_watson_machine_l
earning) (2.0.4)
Requirement already satisfied: zipp>=0.5 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from importlib-metadata->ibm_watson_machine_learnin
g) (3.6.0)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /opt/conda/envs/Python-3.9/lib/python3.9/site-packages (from packaging->ibm_watson_machine_l
earning) (3.0.4)
```

Reading Dataset

In [3]:

```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.
cos_client = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='g1IDqsEFOzpgPkIIAaKTQN61e3iSsW6bmIlqLruc1oE4',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

bucket = 'customermodeldeployment-donotdelete-pr-95uwaji0d4dfmd'
object_key = 'water_data1.txt'

streaming_body_1 = cos_client.get_object(Bucket=bucket, Key=object_key)['Body']
data = pd.read_csv(streaming_body_1)

# Your data file was loaded into a botocore.response.StreamingBody object.
# Please read the documentation of ibm_boto3 and pandas to learn more about the possibilities to load the data.
# ibm_boto3 documentation: https://ibm.github.io/ibm-cos-sdk-python/
# pandas documentation: http://pandas.pydata.org/
```

Analyse the data

In [4]:
```
data.head()
```

Out[4]:

	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	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	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

In [5]:
```
data.describe()
```

Out[5]:

	year
count	1991.000000
mean	2010.038172
std	3.057333
min	2003.000000
25%	2008.000000
50%	2011.000000
75%	2013.000000
max	2014.000000

In [6]:
```
data.info()
```

```
RangeIndex: 1991 entries, 0 to 1990
Data columns (total 12 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   STATION CODE                    1991 non-null   object
 1   LOCATIONS                       1991 non-null   object
 2   STATE                           1991 non-null   object
 3   Temp                            1991 non-null   object
 4   D.O. (mg/l)                     1991 non-null   object
 5   PH                              1991 non-null   object
 6   CONDUCTIVITY (µmhos/cm)         1991 non-null   object
 7   B.O.D. (mg/l)                   1991 non-null   object
 8   NITRATENAN N+ NITRITENANN (mg/l)  1991 non-null   object
 9   FECAL COLIFORM (MPN/100ml)      1991 non-null   object
 10  TOTAL COLIFORM (MPN/100ml)Mean  1991 non-null   object
 11  year                            1991 non-null   int64
dtypes: int64(1), object(11)
memory usage: 186.8+ KB
```

In [7]:
```
data.shape
```

Out[7]: (1991, 12)

Handling Missing Values_1

In [8]:
```
data.isnull().any()
```

Out[8]:
```
STATION CODE                      False
LOCATIONS                         False
STATE                             False
Temp                              False
D.O. (mg/l)                       False
PH                                False
CONDUCTIVITY (µmhos/cm)           False
B.O.D. (mg/l)                     False
NITRATENAN N+ NITRITENANN (mg/l)  False
FECAL COLIFORM (MPN/100ml)        False
TOTAL COLIFORM (MPN/100ml)Mean    False
year                              False
dtype: bool
```

In [9]:
```
data.isnull().sum()
```

Out[9]:
```
STATION CODE                      0
LOCATIONS                         0
STATE                             0
Temp                              0
D.O. (mg/l)                       0
PH                                0
CONDUCTIVITY (µmhos/cm)           0
B.O.D. (mg/l)                     0
NITRATENAN N+ NITRITENANN (mg/l)  0
FECAL COLIFORM (MPN/100ml)        0
TOTAL COLIFORM (MPN/100ml)Mean    0
year                              0
dtype: int64
```

In [10]:
```
data.dtypes
```

Out[10]:
```
STATION CODE                        object
LOCATIONS                           object
STATE                               object
Temp                                object
D.O. (mg/l)                         object
PH                                  object
CONDUCTIVITY (µmhos/cm)             object
B.O.D. (mg/l)                       object
NITRATENAN N+ NITRITENANN (mg/l)    object
FECAL COLIFORM (MPN/100ml)          object
TOTAL COLIFORM (MPN/100ml)Mean      object
year                                 int64
dtype: object
```

In [11]:
```
data['Temp']=pd.to_numeric(data['Temp'],errors='coerce')
data['D.O. (mg/l)']=pd.to_numeric(data['D.O. (mg/l)'],errors='coerce')
data['PH']=pd.to_numeric(data['PH'],errors='coerce')
data['B.O.D. (mg/l)']=pd.to_numeric(data['B.O.D. (mg/l)'],errors='coerce')
data['CONDUCTIVITY (µmhos/cm)']=pd.to_numeric(data['CONDUCTIVITY (µmhos/cm)'],errors='coerce')
data['NITRATENAN N+ NITRITENANN (mg/l)']=pd.to_numeric(data['NITRATENAN N+ NITRITENANN (mg/l)'],errors='coerce')
data['TOTAL COLIFORM (MPN/100ml)Mean']=pd.to_numeric(data['TOTAL COLIFORM (MPN/100ml)Mean'],errors='coerce')
data.dtypes
```

Out[11]:
```
STATION CODE                         object
LOCATIONS                            object
STATE                                object
Temp                                float64
D.O. (mg/l)                         float64
PH                                  float64
CONDUCTIVITY (µmhos/cm)             float64
B.O.D. (mg/l)                       float64
NITRATENAN N+ NITRITENANN (mg/l)    float64
FECAL COLIFORM (MPN/100ml)           object
TOTAL COLIFORM (MPN/100ml)Mean      float64
year                                  int64
dtype: object
```

In [12]:
```
data.isnull().sum()
```

Out[12]:
```
STATION CODE                          0
LOCATIONS                             0
STATE                                 0
Temp                                 92
D.O. (mg/l)                          31
PH                                    8
CONDUCTIVITY (µmhos/cm)              25
B.O.D. (mg/l)                        43
NITRATENAN N+ NITRITENANN (mg/l)    225
FECAL COLIFORM (MPN/100ml)            0
TOTAL COLIFORM (MPN/100ml)Mean      132
year                                  0
dtype: int64
```

In [13]:
```
data['Temp'].fillna(data['Temp'].mean(),inplace=True)
data['D.O. (mg/l)'].fillna(data['D.O. (mg/l)'].mean(),inplace=True)
data['PH'].fillna(data['PH'].mean(),inplace=True)
data['CONDUCTIVITY (µmhos/cm)'].fillna(data['CONDUCTIVITY (µmhos/cm)'].mean(),inplace=True)
data['B.O.D. (mg/l)'].fillna(data['B.O.D. (mg/l)'].mean(),inplace=True)
data['NITRATENAN N+ NITRITENANN (mg/l)'].fillna(data['NITRATENAN N+ NITRITENANN (mg/l)'].mean(),inplace=True)
data['TOTAL COLIFORM (MPN/100ml)Mean'].fillna(data['TOTAL COLIFORM (MPN/100ml)Mean'].mean(),inplace=True)
```

In [14]:
```
data.drop(["FECAL COLIFORM (MPN/100ml)"],axis=1,inplace=True)
```

In [15]:
```
data=data.rename(columns = {'D.O. (mg/l)': 'do'})
data=data.rename(columns = {'CONDUCTIVITY (µmhos/cm)': 'co'})
data=data.rename(columns = {'B.O.D. (mg/l)': 'bod'})
data=data.rename(columns = {'NITRATENAN N+ NITRITENANN (mg/l)': 'na'})
data=data.rename(columns = {'TOTAL COLIFORM (MPN/100ml)Mean': 'tc'})
data=data.rename(columns = {'STATION CODE': 'station'})
data=data.rename(columns = {'LOCATIONS': 'location'})
data=data.rename(columns = {'STATE': 'state'})
data=data.rename(columns = {'PH': 'ph'})
```

Water Quality Index (WQI) Calculation

In [16]:
```
#calculation of pH
data['npH']=data.ph.apply(lambda x: (100 if(8.5>=x>=7)
                                  else(80 if(8.6>=x>=8.5) or (6.9>=x>=6.8)
                                    else (60 if(8.8>=x>=8.6) or (6.8>=x>=6.7)
                                        else(40 if(9>=x>=8.8) or (6.7>=x>=6.5)
                                            else 0)))))
```

In [17]:
```
#calculation of dissolved oxygen
data['ndo']=data.do.apply(lambda x: (100 if(x>=6)
                                  else(80 if(6>=x>=5.1)
                                    else (60 if(5>=x>=4.1)
                                        else(40 if(4>=x>=3)
                                            else 0)))))
```

In [18]:
```
#calculation of total coliform
data['nco']=data.tc.apply(lambda x: (100 if(5>=x>=0)
                                  else(80 if(50>=x>=5)
                                    else (60 if(500>=x>=50)
                                        else(40 if(10000>=x>=500)
                                            else 0)))))
```

In [19]:
```
#calculation of B.D.O
data['nbdo']=data.bod.apply(lambda x:(100 if(3>=x>=0)
                                  else(80 if(6>=x>=3)
                                    else (60 if(80>=x>=6)
                                        else(40 if(125>=x>=80)
                                            else 0)))))
```

In [20]:

```
#calculation of electric conductivity
data['nec']=data.co.apply(lambda x:(100 if(75>=x>=0)
                                    else(80 if(150>=x>=75)
                                      else (60 if(225>=x>=150)
                                          else(40 if(300>=x>=225)
                                              else 0)))))
```

In [21]:

```
#calculation of nitrate
data['nna']=data.na.apply(lambda x:(100 if(20>=x>=0)
                                    else(80 if(50>=x>=20)
                                      else (60 if(100>=x>=50)
                                          else(40 if(200>=x>=100)
                                              else 0)))))
```

In [22]:

```
#Calculation of Water Quality Index WQI
data['wph']=data.npH*0.165
data['wdo']=data.ndo*0.281
data['wbdo']=data.nbdo*0.234
data['wec']=data.nec*0.009
data['wna']=data.nna*0.028
data['wco']=data.nco*0.281
data['wqi']=data.wph+data.wdo+data.wbdo+data.wec+data.wna+data.wco
data
```

Out[22]:

	station	location	state	Temp	do	ph	co	bod	na	tc	...	nbdo	nec	nna	wph	wdo	wbdo	wec	wna	wco	wqi
0	1393	DAMANGANGA AT D/S OF MADHUBAN, DAMAN	DAMAN & DIU	30.600000	6.7	7.5	203.0	6.940049	0.100000	27.0	...	60	60	100	16.5	28.10	14.04	0.54	2.8	22.48	84.46
1	1399	ZUARI AT D/S OF PT. WHERE KUMBARJRIA CANAL JOI...	GOA	29.800000	5.7	7.2	189.0	2.000000	0.200000	8391.0	...	100	60	100	16.5	22.48	23.40	0.54	2.8	11.24	76.96
2	1475	ZUARI AT PANCHAWADI	GOA	29.500000	6.3	6.9	179.0	1.700000	0.100000	5330.0	...	100	60	100	13.2	28.10	23.40	0.54	2.8	11.24	79.28
3	3181	RIVER ZUARI AT BORIM BRIDGE	GOA	29.700000	5.8	6.9	64.0	3.800000	0.500000	8443.0	...	80	100	100	13.2	22.48	18.72	0.90	2.8	11.24	69.34
4	3182	RIVER ZUARI AT MARCAIM JETTY	GOA	29.500000	5.8	7.3	83.0	1.900000	0.400000	5500.0	...	100	80	100	16.5	22.48	23.40	0.72	2.8	11.24	77.14
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1986	1330	TAMBIRAPARANI AT ARUMUGANERI, TAMILNADU	NAN	26.209814	7.9	738.0	7.2	2.700000	0.518000	202.0	...	100	100	100	0.0	28.10	23.40	0.90	2.8	16.86	72.06
1987	1450	PALAR AT VANIYAMBADI WATER SUPPLY HEAD WORK, T...	NAN	29.000000	7.5	585.0	6.3	2.600000	0.155000	315.0	...	100	100	100	0.0	28.10	23.40	0.90	2.8	16.86	72.06
1988	1403	GUMTI AT U/S SOUTH TRIPURA,TRIPURA	NAN	28.000000	7.6	98.0	6.2	1.200000	1.623079	570.0	...	100	100	100	0.0	28.10	23.40	0.90	2.8	11.24	66.44
1989	1404	GUMTI AT D/S SOUTH TRIPURA, TRIPURA	NAN	28.000000	7.7	91.0	6.5	1.300000	1.623079	562.0	...	100	100	100	0.0	28.10	23.40	0.90	2.8	11.24	66.44
1990	1726	CHANDRAPUR, AGARTALA D/S OF HAORA RIVER, TRIPURA	NAN	29.000000	7.6	110.0	5.7	1.100000	1.623079	546.0	...	100	100	100	0.0	28.10	23.40	0.90	2.8	11.24	66.44

In [23]:
```
#Calculation of overall WQI for each year
average = data.groupby('year')['wqi'].mean()
average.head()
```

Out[23]:
```
year
2003    66.239545
2004    61.290000
2005    73.762689
2006    72.909714
2007    74.233000
Name: wqi, dtype: float64
```

Splitting Dependent and Independent Columns

In [24]:
```
data.head()
data.drop(['location','station','state'],axis =1,inplace=True)
```

In [25]:
```
data.head()
```

Out[25]:

	Temp	do	ph	co	bod	na	tc	year	npH	ndo	...	nbdo	nec	nna	wph	wdo	wbdo	wec	wna	wco	wqi
0	30.6	6.7	7.5	203.0	6.940049	0.1	27.0	2014	100	100	...	60	60	100	16.5	28.10	14.04	0.54	2.8	22.48	84.46
1	29.8	5.7	7.2	189.0	2.000000	0.2	8391.0	2014	100	80	...	100	60	100	16.5	22.48	23.40	0.54	2.8	11.24	76.96
2	29.5	6.3	6.9	179.0	1.700000	0.1	5330.0	2014	80	100	...	100	60	100	13.2	28.10	23.40	0.54	2.8	11.24	79.28
3	29.7	5.8	6.9	64.0	3.800000	0.5	8443.0	2014	80	80	...	80	100	100	13.2	22.48	18.72	0.90	2.8	11.24	69.34
4	29.5	5.8	7.3	83.0	1.900000	0.4	5500.0	2014	100	80	...	100	80	100	16.5	22.48	23.40	0.72	2.8	11.24	77.14

5 rows × 21 columns

In [26]:
```
x=data.iloc[:,1:7].values
```

In [27]:
```
x.shape
```

Out[27]: (1991, 6)

In [28]:
```
y=data.iloc[:,-1:].values
y.shape
```

Out[28]: (1991, 1)

In [29]:
```
print(x)
```

```
[[6.70000000e+00 7.50000000e+00 2.03000000e+02 6.94004877e+00
  1.00000000e-01 2.70000000e+01]
 [5.70000000e+00 7.20000000e+00 1.89000000e+02 2.00000000e+00
  2.00000000e-01 8.39100000e+03]
 [6.30000000e+00 6.90000000e+00 1.79000000e+02 1.70000000e+00
  1.00000000e-01 5.33000000e+03]
 ...
 [7.60000000e+00 9.80000000e+01 6.20000000e+00 1.20000000e+00
  1.62307871e+00 5.70000000e+02]
 [7.70000000e+00 9.10000000e+01 6.50000000e+00 1.30000000e+00
  1.62307871e+00 5.62000000e+02]
 [7.60000000e+00 1.10000000e+02 5.70000000e+00 1.10000000e+00
  1.62307871e+00 5.46000000e+02]]
```

In [30]:
```
print(y)
```

```
[[84.46]
 [76.96]
 [79.28]
 ...
 [66.44]
 [66.44]
 [66.44]]
```

Splitting the Data Into Train and Test

In [31]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.2,random_state=10)
```

Random_Forest_Regression

In [32]:
```
#Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [33]:
```
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 10, random_state = 0)
regressor.fit(x_train, y_train)
y_pred = regressor.predict(x_test)
```

```
/tmp/wsuser/ipykernel_224/1531495111.py:3: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of
y to (n_samples,), for example using ravel().
  regressor.fit(x_train, y_train)
```

Model Evaluation

In [34]:
```
from sklearn import metrics
print('MAE:',metrics.mean_absolute_error(y_test,y_pred))
print('MSE:',metrics.mean_squared_error(y_test,y_pred))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
```

```
MAE: 0.9872080200501312
MSE: 5.555095879699248
RMSE: 2.3569250899634566
```

In [35]:
```
#accuracy of the model
metrics.r2_score(y_test, y_pred)
```

Out[35]:
```
0.96971918125809
```

Save The Model

In [36]:
```
import pickle
pickle.dump(regressor,open('wqi.pkl', 'wb'))
model = pickle.load(open('wqi.pkl','rb'))
```

In [37]:
```
from ibm_watson_machine_learning import APIClient

wml_credentials={
    "url":"https://us-south.ml.cloud.ibm.com",
    "apikey":"yXIqwSJMW8_msu96HBxRumGLj14Q7YF7HqpUsusCqrCI"

}
client = APIClient(wml_credentials)
```

In [38]:
```
def guid_from_space_name(client,space_name):
    space = client.spaces.get_details()
#         print(space)
    return(next(item for item in space['resources'] if item['entity']['name']==space_name)['metadata']['id'])
```

In [39]:
```
space_uid = guid_from_space_name(client,'Models')
print("Spcae UID = "+ space_uid)
```

```
Spcae UID = dc1f4882-c69d-4ed0-893b-d28c4033e082
```

In [40]:
```
client.set.default_space(space_uid)
```

Out[40]:
```
'SUCCESS'
```

In [41]:

```
client.software_specifications.list()
```

NAME	ASSET_ID	TYPE
default_py3.6	0062b8c9-8b7d-44a0-a9b9-46c416adcbd9	base
kernel-spark3.2-scala2.12	020d69ce-7ac1-5e68-ac1a-31189867356a	base
pytorch-onnx_1.3-py3.7-edt	069ea134-3346-5748-b513-49120e15d288	base
scikit-learn_0.20-py3.6	09c5a1d0-9c1e-4473-a344-eb7b665ff687	base
spark-mllib_3.0-scala_2.12	09f4cff0-90a7-5899-b9ed-1ef348aebdee	base
pytorch-onnx_rt22.1-py3.9	0b848dd4-e681-5599-be41-b5f6fccc6471	base
ai-function_0.1-py3.6	0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda	base
shiny-r3.6	0e6e79df-875e-4f24-8ae9-62dcc2148306	base
tensorflow_2.4-py3.7-horovod	1092590a-307d-563d-9b62-4eb7d64b3f22	base
pytorch_1.1-py3.6	10ac12d6-6b30-4ccd-8392-3e922c096a92	base
tensorflow_1.15-py3.6-ddl	111e41b3-de2d-5422-a4d6-bf776828c4b7	base
autoai-kb_rt22.2-py3.10	125b6d9a-5b1f-5e8d-972a-b251688ccf40	base
runtime-22.1-py3.9	12b83a17-24d8-5082-900f-0ab31fbfd3cb	base
scikit-learn_0.22-py3.6	154010fa-5b3b-4ac1-82af-4d5ee5abbc85	base
default_r3.6	1b70aec3-ab34-4b87-8aa0-a4a3c8296a36	base
pytorch-onnx_1.3-py3.6	1bc6029a-cc97-56da-b8e0-39c3880dbbe7	base
kernel-spark3.3-r3.6	1c9e5454-f216-59dd-a20e-474a5cdf5988	base
pytorch-onnx_rt22.1-py3.9-edt	1d362186-7ad5-5b59-8b6c-9d0880bde37f	base
tensorflow_2.1-py3.6	1eb25b84-d6ed-5dde-b6a5-3fbdf1665666	base
spark-mllib_3.2	20047f72-0a98-58c7-9ff5-a77b012eb8f5	base
tensorflow_2.4-py3.8-horovod	217c16f6-178f-56bf-824a-b19f20564c49	base
runtime-22.1-py3.9-cuda	26215f05-08c3-5a41-a1b0-da66306ce658	base
do_py3.8	295addb5-9ef9-547e-9bf4-92ae3563e720	base
autoai-ts_3.8-py3.8	2aa0c932-798f-5ae9-abd6-15e0c2402fb5	base
tensorflow_1.15-py3.6	2b73a275-7cbf-420b-a912-eae7f436e0bc	base
kernel-spark3.3-py3.9	2b7961e2-e3b1-5a8c-a491-482c8368839a	base
pytorch_1.2-py3.6	2c8ef57d-2687-4b7d-acce-01f94976dac1	base
spark-mllib_2.3	2e51f700-bca0-4b0d-88dc-5c6791338875	base
pytorch-onnx_1.1-py3.6-edt	32983cea-3f32-4400-8965-dde874a8d67e	base
spark-mllib_3.0-py37	36507ebe-8770-55ba-ab2a-eafe787600e9	base
spark-mllib_2.4	390d21f8-e58b-4fac-9c55-d7ceda621326	base
autoai-ts_rt22.2-py3.10	396b2e83-0953-5b86-9a55-7ce1628a406f	base
xgboost_0.82-py3.6	39e31acd-5f30-41dc-ae44-60233c80306e	base

```
pytorch-onnx_1.2-py3.6-edt     40589d0e-7019-4e28-8daa-fb03b6f4fe12  base
pytorch-onnx_rt22.2-py3.10     40e73f55-783a-5535-b3fa-0c8b94291431  base
default_r36py38                41c247d3-45f8-5a71-b065-8580229facf0  base
autoai-ts_rt22.1-py3.9         4269d26e-07ba-5d40-8f66-2d495b0c71f7  base
autoai-obm_3.0                 42b92e18-d9ab-567f-988a-4240ba1ed5f7  base
pmml-3.0_4.3                   493bcb95-16f1-5bc5-bee8-81b8af80e9c7  base
spark-mllib_2.4-r_3.6          49403dff-92e9-4c87-a3d7-a42d0021c095  base
xgboost_0.90-py3.6             4ff8d6c2-1343-4c18-85e1-689c965304d3  base
pytorch-onnx_1.1-py3.6         50f95b2a-bc16-43bb-bc94-b0bed208c60b  base
autoai-ts_3.9-py3.8            52c57136-80fa-572e-8728-a5e7cbb42cde  base
spark-mllib_2.4-scala_2.11     55a70f99-7320-4be5-9fb9-9edb5a443af5  base
spark-mllib_3.0                5c1b0ca2-4977-5c2e-9439-ffd44ea8ffe9  base
autoai-obm_2.0                 5c2e37fa-80b8-5e77-840f-d912469614ee  base
spss-modeler_18.1              5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b  base
cuda-py3.8                     5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e  base
autoai-kb_3.1-py3.7            632d4b22-10aa-5180-88f0-f52dfb6444d7  base
pytorch-onnx_1.7-py3.8         634d3cdc-b562-5bf9-a2d4-ea90a478456b  base
----------------------------  ------------------------------------  ----
Note: Only first 50 records were displayed. To display more use 'limit' parameter.
```

In [58]:

```
software_spec_uid = client.software_specifications.get_uid_by_name("runtime-22.1-py3.9")
#software_spec_uid = client.software_specifications.get_uid_by_name("default_py3.6")
software_spec_uid
```

Out[58]:

```
'12b83a17-24d8-5082-900f-0ab31fbfd3cb'
```

In [60]:

```
model_details = client.repository.store_model(model=regressor,meta_props={
    client.repository.ModelMetaNames.NAME:"water_dataX",
    client.repository.ModelMetaNames.TYPE:"scikit-learn_1.0     ",
    client.repository.ModelMetaNames.SOFTWARE_SPEC_UID:software_spec_uid}
                                             )
model_id=client.repository.get_model_id(model_details)
```

In [61]:

```
model_id
```

Out[61]:

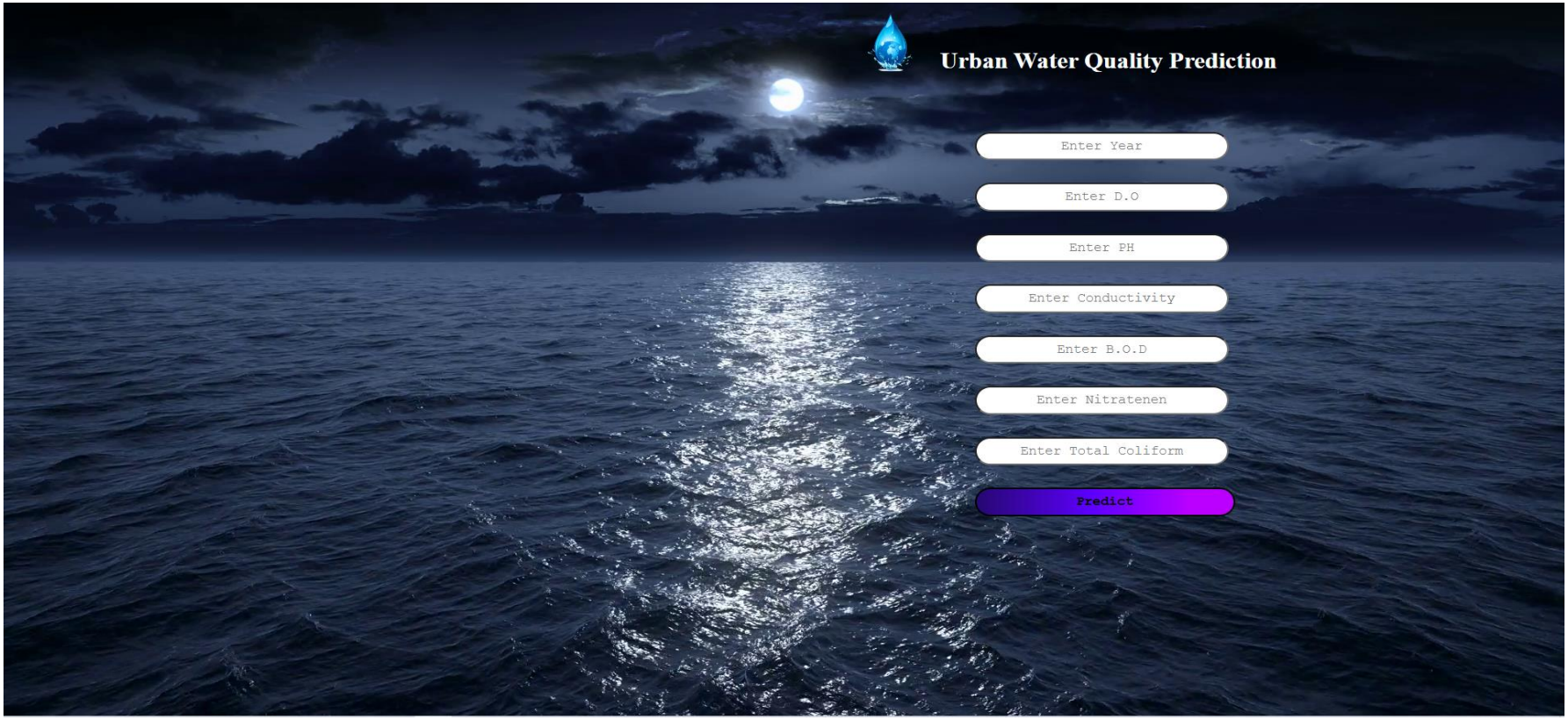
```
'7dd68f49-673c-498a-b799-68088f6c52a3'
```

In [62]:
```
x_train[0]
```

Out[62]:
```
array([-0.684425  , -0.06140958, -0.20384397,  0.06998719,  0.86213052,
       -0.042079  ])
```

In [63]:
```
regressor.predict([[-0.684425  , -0.06140958, -0.20384397,  0.06998719,  0.86213052,
       -0.042079]])
```

Out[63]:
```
array([67.06])
```


Urban Water Quality Prediction
Enter Year
Enter D.O
Enter PH
Enter Conductivity
Enter B.O.D
Enter Nitratenen
Enter Total Coliform
Predict