

SPRINT 3

Date	11-10-2022
Team ID	PNT2022TMID35567
Project Name	Efficient water quality analysis and prediction using Machine Learning

CODE :

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
import pickle
data = pd.read_csv('./Data/water_dataX.csv', encoding='Latin-1', low_memory=False)
data.head()
data.describe()
data.info()
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')
print(data.dtypes)
data.isnull().sum()
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)
data.drop(['FECAL COLIFORM (MPN/100ml)'], axis=1, inplace=True)
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'})

```

```

data.drop(["station"], axis =1, inplace= True)
data.drop(["location"], axis =1, inplace= True)
data.drop(["state"], axis =1, inplace= True)
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>=6.4)
                                else 0))))))

```

```

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))))))

```

```

data['nco']= data.co.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))))))

```

```

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))))))

```

```

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))))))

```

```

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))))))

```

```

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
```

```
average= data.groupby('year')['wqi'].mean()
average.head()
```

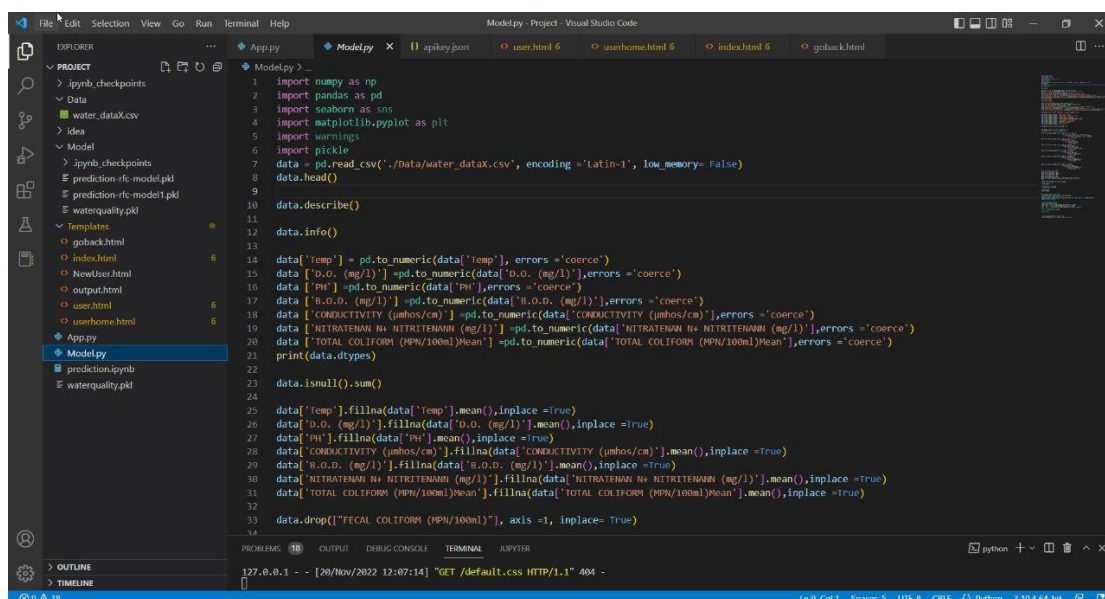
```
x= data.iloc[:,0:7].values
y = data.iloc[:,7:].values
```

```
print(x.shape)
print(y.shape)
```

```
from sklearn import linear_model
from sklearn.model_selection import train_test_split
reg= linear_model.LinearRegression()
X_train, X_test, y_train, y_test = train_test_split(x,y, test_size = 0.2, random_state
=10)
reg.fit(X_train, y_train)
```

```
from sklearn import metrics
y_pred = reg.predict(X_test)
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)))
metrics.r2_score(y_test, y_pred)
import pickle
pickle.dump(reg,open('reg_rf.pkl','wb'))
print("Training process is complete Model File Saved!")
```

SCREENSHOT :



```
1 import numpy as np
2 import pandas as pd
3 import seaborn as sns
4 import matplotlib.pyplot as plt
5 import warnings
6 import pickle
7 data = pd.read_csv('./data/water_dataX.csv', encoding = 'Latin-1', low_memory= False)
8 data.head()
9
10 data.describe()
11
12 data.info()
13
14 data['temp'] = pd.to_numeric(data['temp'], errors = 'coerce')
15 data['D.O. (mg/l)'] = pd.to_numeric(data['D.O. (mg/l)'], errors = 'coerce')
16 data['PH'] = pd.to_numeric(data['PH'], errors = 'coerce')
17 data['B.O.D. (mg/l)'] = pd.to_numeric(data['B.O.D. (mg/l)'], errors = 'coerce')
18 data['CONDUCTIVITY (umhos/cm)'] = pd.to_numeric(data['CONDUCTIVITY (umhos/cm)'], errors = 'coerce')
19 data['NITRATEMM N+ NITRITEMMN (mg/l)'] = pd.to_numeric(data['NITRATEMM N+ NITRITEMMN (mg/l)'], errors = 'coerce')
20 data['TOTAL COLIFORM (MPN/100ml)Mean'] = pd.to_numeric(data['TOTAL COLIFORM (MPN/100ml)Mean'], errors = 'coerce')
21 print(data.dtypes)
22
23 data.isnull().sum()
24
25 data['temp'].fillna(data['temp'].mean(),inplace =True)
26 data['D.O. (mg/l)'].fillna(data['D.O. (mg/l)'].mean(),inplace =True)
27 data['PH'].fillna(data['PH'].mean(),inplace =True)
28 data['CONDUCTIVITY (umhos/cm)'].fillna(data['CONDUCTIVITY (umhos/cm)'].mean(),inplace =True)
29 data['B.O.D. (mg/l)'].fillna(data['B.O.D. (mg/l)'].mean(),inplace =True)
30 data['NITRATEMM N+ NITRITEMMN (mg/l)'].fillna(data['NITRATEMM N+ NITRITEMMN (mg/l)'].mean(),inplace =True)
31 data['TOTAL COLIFORM (MPN/100ml)Mean'].fillna(data['TOTAL COLIFORM (MPN/100ml)Mean'].mean(),inplace =True)
32
33 data.drop(['FECAL COLIFORM (MPN/100ml)'], axis =1, inplace= True)
34
```

```
File Edit Selection View Go Run Terminal Help
Modelpy - Project - Visual Studio Code

EXPLORER
PROJECT
  > jpyrb_checkpoints
  > Data
    water_data.csv
  > idea
  > Model
    > jpyrb_checkpoints
    > prediction-rcf-model.pkl
    > prediction-rcf-model1.pkl
    > waterquality.pkl
  > Templates
    > goback.html
    > index.html
    > NewUser.html
    > output.html
    > user.html
    > userhome.html
  > Apppy
  > Modelpy
  > prediction.jpyrb
  > waterquality.pkl

Modelpy > ...
33 data.drop(["FECAL COLIFORM (MPN/100ml)"], axis =1, inplace= True)
34
35 data = data.rename (columns = {'D.O. (mg/l)':'do'})
36 data = data.rename (columns = {'CONDUCTIVITY (umhos/cm)':'co'})
37 data = data.rename (columns = {'B.O.D. (mg/l)':'bod'})
38 data = data.rename (columns = {'NITRATEAN N+ NITREITENANN (mg/l)':'na'})
39 data = data.rename (columns = {'TOTAL COLIFORM (MPN/100ml)Mean':'tc'})
40 data = data.rename (columns = {'STATION CODE':'station'})
41 data = data.rename (columns = {'LOCATIONS':'location'})
42 data = data.rename (columns = {'STATE':'state'})
43 data = data.rename (columns = {'PH':'ph'})
44
45 data.drop(["station"], axis =1, inplace= True)
46
47
48 data.drop(["location"], axis =1, inplace= True)
49 data.drop(["state"], axis =1, inplace= True)
50
51 data["npl"] = data.ph.apply(lambda x:(100 if ( 8.5 >= x >= 7)
52                               else (80 if (8.6 >=x= 8.5) or (6.9>=x>6.8)
53                               else (60 if (8.8>= x >=8.6) or (6.8 >= x=6.7)
54                               else (40 if (9>=x>8.8) or (6.7 >=x>6.6)
55                               else 0))))
56 data["ndo"] = data.do.apply(lambda x:(100 if (x>=6)
57                               else (80 if (6>=x>5.1)
58                               else (60 if (5>=x>4.1)
59                               else (40 if (4>=x>3)
60                               else 0))))
61 data["nco"] = data.co.apply(lambda x:(100 if (5>=x>=0)
62                               else (80 if (500>=x>=50)
63                               else (60 if (500>=x>=50)
64                               else (40 if (10000>=x>=500)
65                               else 0))))
66 data["nbdo"] = data.bod.apply(lambda x:(100 if (3>=x>=0)
67                               else (80 if (6>=x>=1)
68                               else (60 if (80>=x>=6)
69                               else (40 if (125>=x>=80)
70                               else 0))))
71 data["nec"] = data.co.apply(lambda x:(100 if (75>=x>=0)
72                               else (80 if (150>=x>=75)
73                               else (60 if (225>=x>=150)
74                               else (40 if (300>=x>=225)
75                               else 0))))
76
77
78 data["nna"] = data.na.apply(lambda x:(100 if (20>=x>=0)
79                               else (80 if (50>=x>=20)
80                               else (60 if (100>=x>=50)
81                               else (40 if (200>=x>=100)
82                               else 0))))
83
84 data["wph"] = data.nph * 0.165
85 data["wdo"] = data.ndo * 0.281
86 data["wbdo"] = data.nbdo * 0.234
87 data["wec"] = data.nec * 0.009
88 data["wna"] = data.nna * 0.028
89 data["wco"] = data.nco * 0.281
90 data["wqi"] = data.wph+data.wdo+data.wbdo+data.wec+data.wna+data.wco
91 data
92
93 average= data.groupby('year')['wqi'].mean()
94
95 average.head()
96
97 x= data.iloc[:,0:7].values
98 y = data.iloc[:,7:].values
99
100 print(x.shape)
101 print(y.shape)
102
103 from sklearn import linear_model
104 from sklearn.model_selection import train_test_split
105 reg= linear_model.LinearRegression()
106 X_train, X_test, y_train, y_test = train_test_split (x,y, test_size = 0.2, random_state =10)
107 reg.fit(X_train, y_train)
108
109 from sklearn import metrics
110 y_pred = reg.predict(X_test)
111 print ('MAE :', metrics.mean_absolute_error(y_test, y_pred))
112 print ('MSE :', metrics.mean_squared_error(y_test, y_pred))
113 print ('RMSE :', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
114
115 metrics.r2_score(y_test, y_pred)
116
117 import pickle
118
119
120
121
122 pickle.dump(reg,open('reg_rf.pkl','wb'))
123 print("Training process is complete Model File Saved")
124
125
```

```
File Edit Selection View Go Run Terminal Help
Modelpy - Project - Visual Studio Code

EXPLORER
PROJECT
  > jpyrb_checkpoints
  > Data
    water_data.csv
  > idea
  > Model
    > jpyrb_checkpoints
    > prediction-rcf-model.pkl
    > prediction-rcf-model1.pkl
    > waterquality.pkl
  > Templates
    > goback.html
    > index.html
    > NewUser.html
    > output.html
    > user.html
    > userhome.html
  > Apppy
  > Modelpy
  > prediction.jpyrb
  > waterquality.pkl

Modelpy > ...
66 data["nbdo"] = data.bod.apply(lambda x:(100 if (3>=x>=0)
67                               else (80 if (6>=x>=1)
68                               else (60 if (80>=x>=6)
69                               else (40 if (125>=x>=80)
70                               else 0))))
71 data["nec"] = data.co.apply(lambda x:(100 if (75>=x>=0)
72                               else (80 if (150>=x>=75)
73                               else (60 if (225>=x>=150)
74                               else (40 if (300>=x>=225)
75                               else 0))))
76
77
78 data["nna"] = data.na.apply(lambda x:(100 if (20>=x>=0)
79                               else (80 if (50>=x>=20)
80                               else (60 if (100>=x>=50)
81                               else (40 if (200>=x>=100)
82                               else 0))))
83
84 data["wph"] = data.nph * 0.165
85 data["wdo"] = data.ndo * 0.281
86 data["wbdo"] = data.nbdo * 0.234
87 data["wec"] = data.nec * 0.009
88 data["wna"] = data.nna * 0.028
89 data["wco"] = data.nco * 0.281
90 data["wqi"] = data.wph+data.wdo+data.wbdo+data.wec+data.wna+data.wco
91 data
92
93 average= data.groupby('year')['wqi'].mean()
94
95 average.head()
96
97 x= data.iloc[:,0:7].values
98 y = data.iloc[:,7:].values
99
100 print(x.shape)
101 print(y.shape)
102
103 from sklearn import linear_model
104 from sklearn.model_selection import train_test_split
105 reg= linear_model.LinearRegression()
106 X_train, X_test, y_train, y_test = train_test_split (x,y, test_size = 0.2, random_state =10)
107 reg.fit(X_train, y_train)
108
109 from sklearn import metrics
110 y_pred = reg.predict(X_test)
111 print ('MAE :', metrics.mean_absolute_error(y_test, y_pred))
112 print ('MSE :', metrics.mean_squared_error(y_test, y_pred))
113 print ('RMSE :', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
114
115 metrics.r2_score(y_test, y_pred)
116
117 import pickle
118
119
120
121
122 pickle.dump(reg,open('reg_rf.pkl','wb'))
123 print("Training process is complete Model File Saved")
124
125
```

```
File Edit Selection View Go Run Terminal Help
Modelpy - Project - Visual Studio Code

EXPLORER
PROJECT
  > jpyrb_checkpoints
  > Data
    water_data.csv
  > idea
  > Model
    > jpyrb_checkpoints
    > prediction-rcf-model.pkl
    > prediction-rcf-model1.pkl
    > waterquality.pkl
  > Templates
    > goback.html
    > index.html
    > NewUser.html
    > output.html
    > user.html
    > userhome.html
  > Apppy
  > Modelpy
  > prediction.jpyrb
  > waterquality.pkl

Modelpy > ...
97 x= data.iloc[:,0:7].values
98 y = data.iloc[:,7:].values
99
100 print(x.shape)
101 print(y.shape)
102
103 from sklearn import linear_model
104 from sklearn.model_selection import train_test_split
105 reg= linear_model.LinearRegression()
106 X_train, X_test, y_train, y_test = train_test_split (x,y, test_size = 0.2, random_state =10)
107 reg.fit(X_train, y_train)
108
109 from sklearn import metrics
110 y_pred = reg.predict(X_test)
111 print ('MAE :', metrics.mean_absolute_error(y_test, y_pred))
112 print ('MSE :', metrics.mean_squared_error(y_test, y_pred))
113 print ('RMSE :', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
114
115 metrics.r2_score(y_test, y_pred)
116
117 import pickle
118
119
120
121
122 pickle.dump(reg,open('reg_rf.pkl','wb'))
123 print("Training process is complete Model File Saved")
124
125
```