SPRINT 3

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

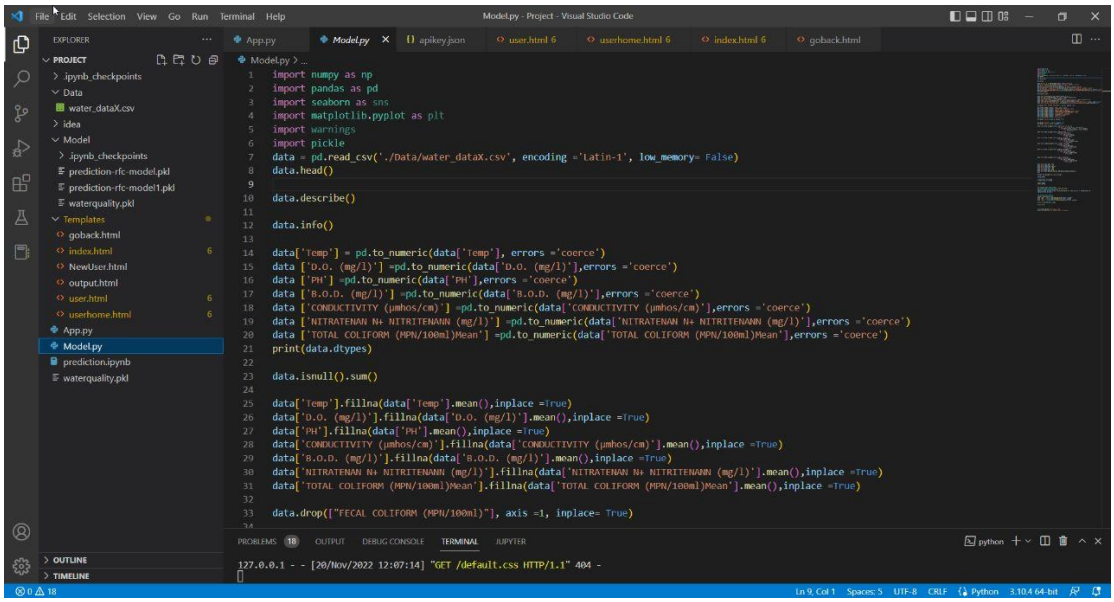
CODE :

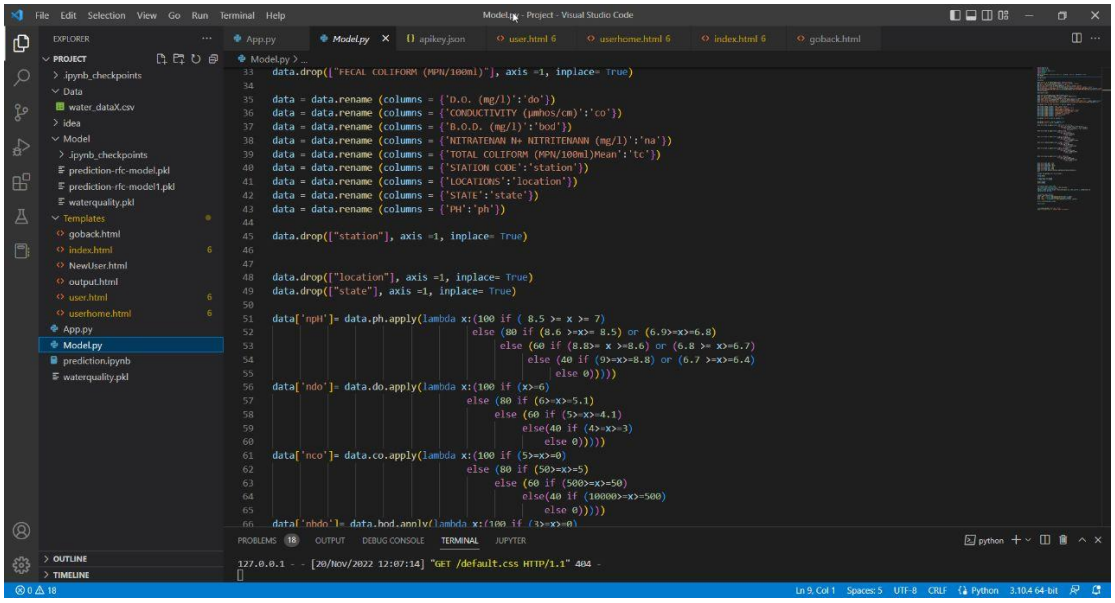
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
import pickle
data = pd.read_csv('./Data/water_dataX.csv', encoding ='Latin-1', low_memory=
False)
data.head()
data.describe()
data.info()
data['Temp'] = pd.to_numeric(data['Temp'], errors ='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'})
```

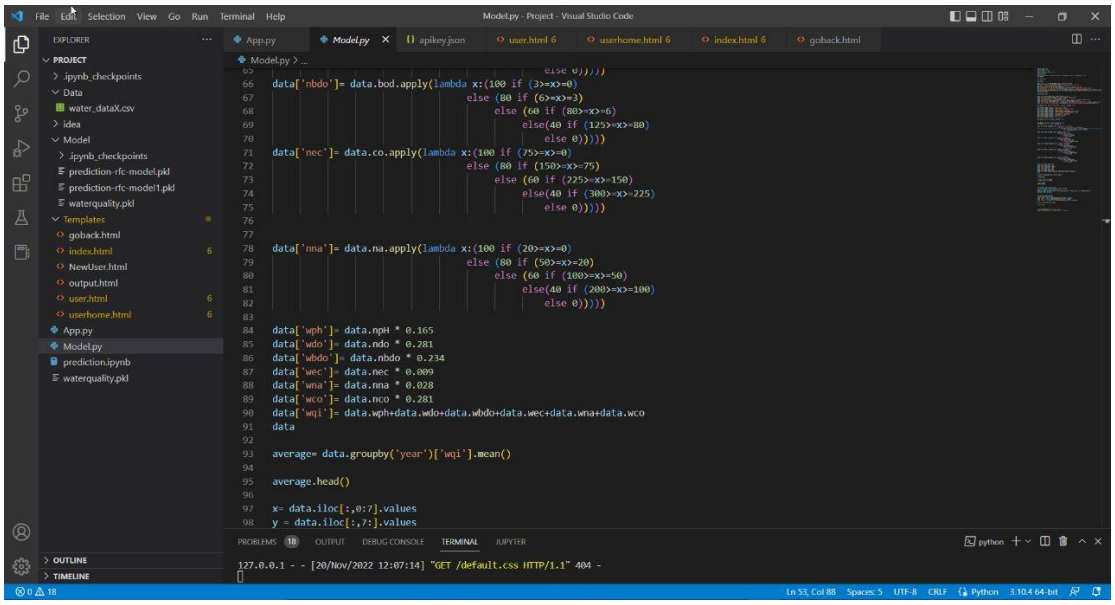
```
data = data.rename (columns = {'CONDUCTIVITY (µmhos/cm)':'co'})
data = data.rename (columns = {'B.O.D. (mg/l)':'bod'})
data = data.rename (columns = {'NITRATENAN N+ NITRITENANN (mg/l)':'na'})
data = data.rename (columns = {'TOTAL COLIFORM (MPN/100ml)Mean':'tc'})
data = data.rename (columns = {'STATION 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
```

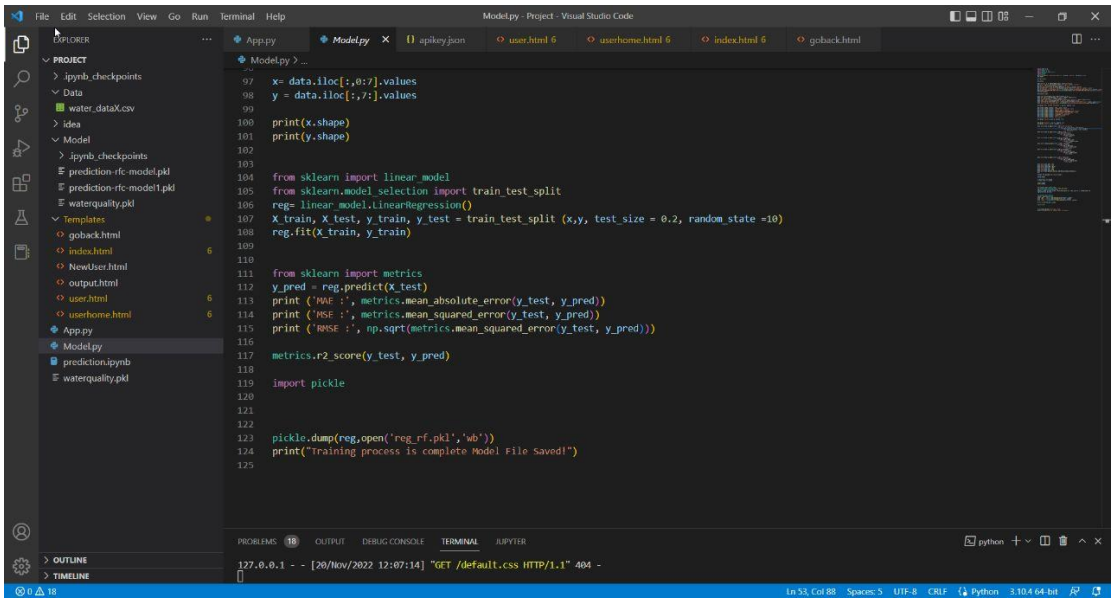
```
data['wco']= data.nco * 0.281
data['wqi']= data.wph+data.wdo+data.wbdo+data.wec+data.wna+data.wco
data

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 :




File Edit Selection View Go Run Terminal Help
Model.py - Project - Visual Studio Code
EXPLORER
PROJECT
App.py
Model.py
apikey.json
user.html 6
userhome.html 6
index.html 6
goback.html
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)
PROBLEMS 18 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
python
127.0.0.1 - - [20/Nov/2022 12:07:14] "GET /default.css HTTP/1.1" 404 -
OUTLINE
TIMELINE
Ln 9, Col 1 Spaces: 5 UTF-8 CRLF Python 3.10.4 64-bit


File Edit Selection View Go Run Terminal Help
Model.py - Project - Visual Studio Code
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
PROBLEMS 18 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
python
127.0.0.1 - - [20/Nov/2022 12:07:14] "GET /default.css HTTP/1.1" 404 -
OUTLINE
TIMELINE
Ln 53, Col 88 Spaces: 5 UTF-8 CRLF Python 3.10.4 64-bit


File Edit Selection View Go Run Terminal Help
Model.py - Project - Visual Studio Code
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!")
PROBLEMS 18 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
python
127.0.0.1 - - [20/Nov/2022 12:07:14] "GET /default.css HTTP/1.1" 404 -
OUTLINE
TIMELINE
Ln 53, Col 88 Spaces: 5 UTF-8 CRLF Python 3.10.4 64-bit