Project Development Phase
Model Performance Test

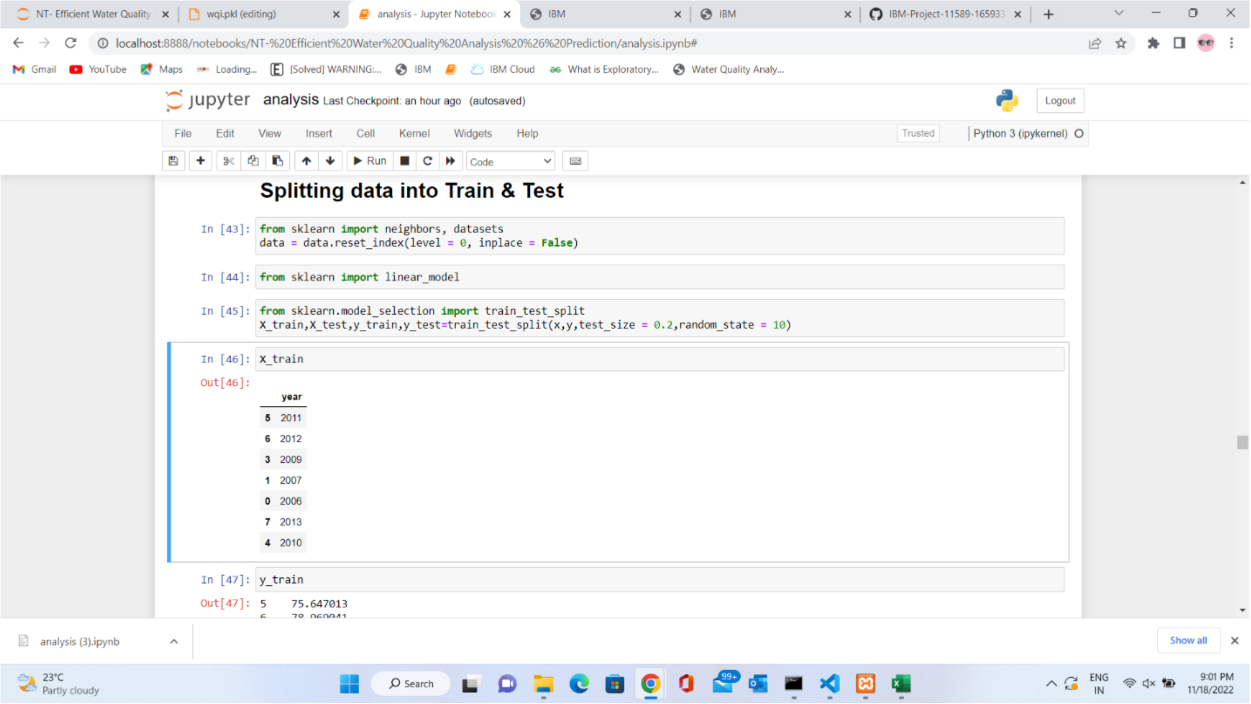
Date	10 November 2022
Team ID	PNT2022TMID08566
Project Name	Project – Efficient Water Quality analysis & prediction
Maximum Marks	10 Marks

Model Performance Testing:

Project team shall fill the following information in model performance testing template.

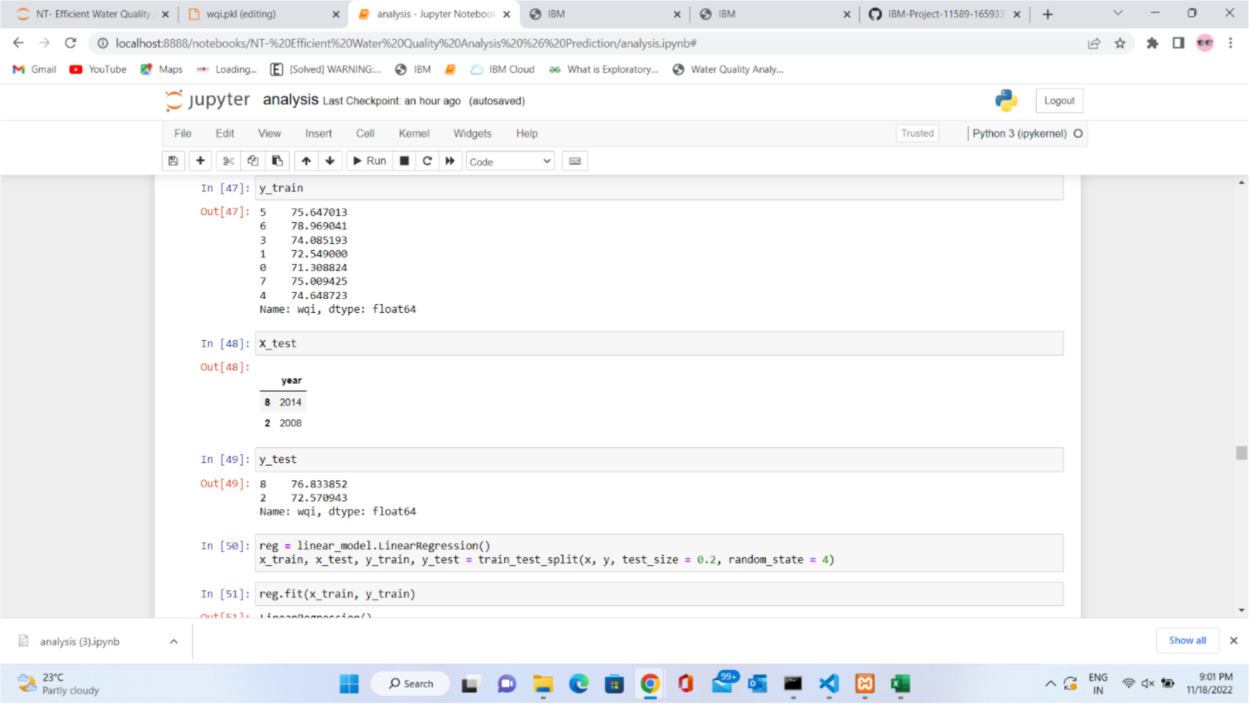
S.No.	Parameter	Values	Screenshot
1.	Metrics	**Regression Model:** MAE - , MSE - , RMSE - , R2 score - **Classification Model:** Confusion Matrix - , Accuray Score- & Classification Report -	Below
2.	Tune the Model	Hyperparameter Tuning - Validation Method -	Below

1.Metrics- Linear Regression



Splitting data into Train & Test

```
In [43]: from sklearn import neighbors, datasets
         data = data.reset_index(level = 0, inplace = False)

In [44]: from sklearn import linear_model

In [45]: 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)

In [46]: X_train

Out[46]:
            year
         5  2011
         6  2012
         3  2009
         1  2007
         0  2006
         7  2013
         4  2010

In [47]: y_train

Out[47]: 5      75.647013
```



In [47]:
```
y_train
```

Out[47]:
```
5    75.647013
6    78.969041
3    74.085193
1    72.549000
0    71.308824
7    75.009425
4    74.648723
Name: wqi, dtype: float64
```

In [48]:
```
x_test
```

Out[48]:

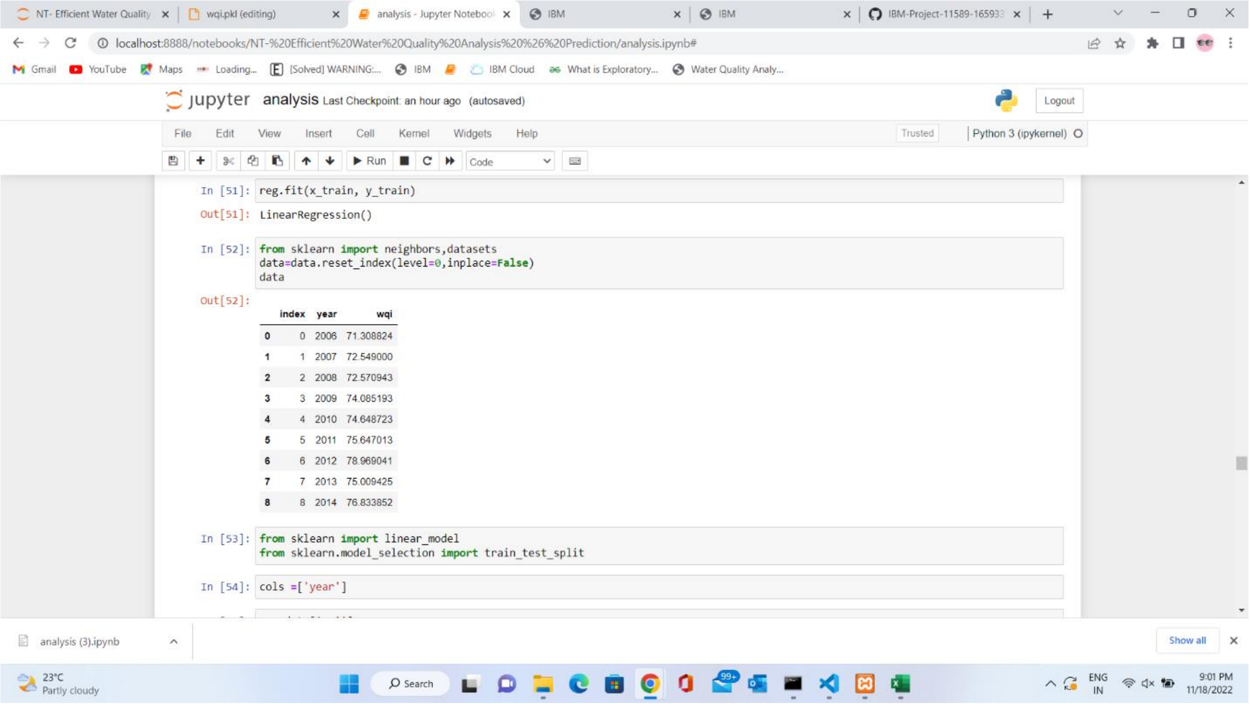
	year
8	2014
2	2008

In [49]:
```
y_test
```

Out[49]:
```
8    76.833852
2    72.570943
Name: wqi, dtype: float64
```

In [50]:
```
reg = linear_model.LinearRegression()
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 4)
```

In [51]:
```
reg.fit(x_train, y_train)
```



In [51]:
```
reg.fit(x_train, y_train)
```

Out[51]:
```
LinearRegression()
```

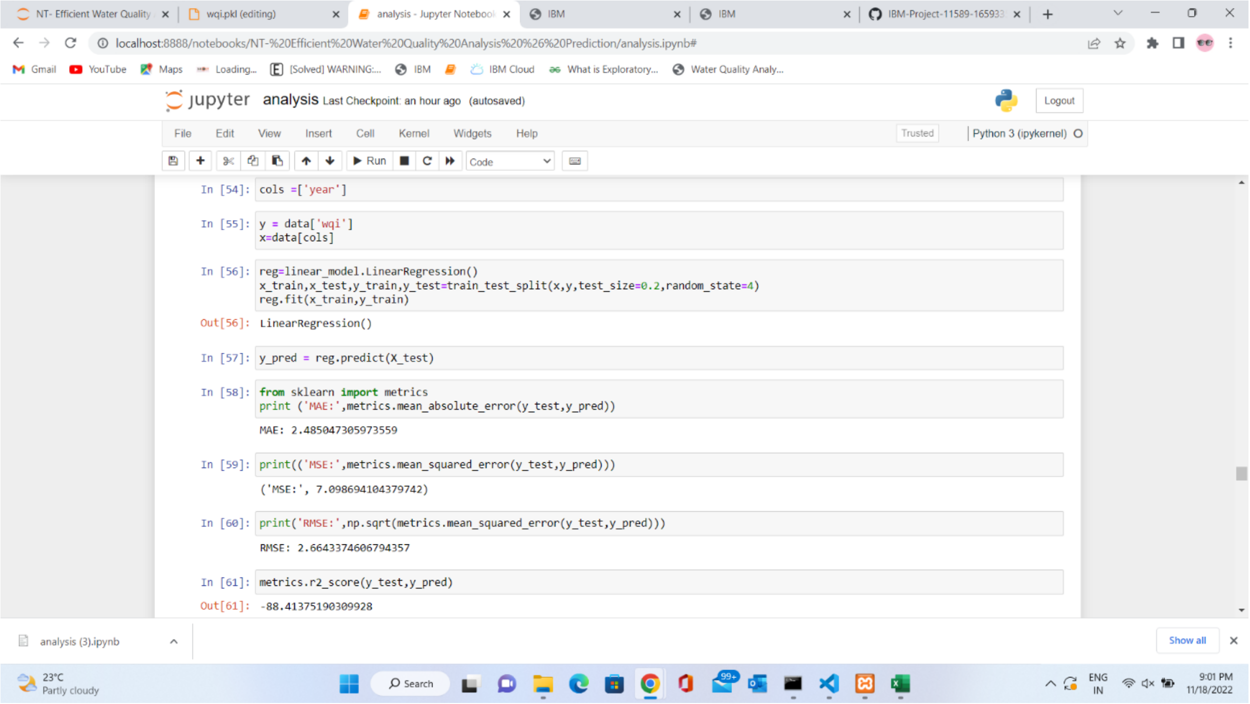
In [52]:
```
from sklearn import neighbors,datasets
data=data.reset_index(level=0,inplace=False)
data
```

Out[52]:

	index	year	wqi
0	0	2006	71.308824
1	1	2007	72.549000
2	2	2008	72.570943
3	3	2009	74.085193
4	4	2010	74.648723
5	5	2011	75.647013
6	6	2012	78.969041
7	7	2013	75.009425
8	8	2014	76.833852

In [53]:
```
from sklearn import linear_model
from sklearn.model_selection import train_test_split
```

In [54]:
```
cols =['year']
```



```
In [54]: cols =['year']

In [55]: y = data['wqi']
         x=data[cols]

In [56]: reg=linear_model.LinearRegression()
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=4)
         reg.fit(x_train,y_train)

Out[56]: LinearRegression()

In [57]: y_pred = reg.predict(X_test)

In [58]: from sklearn import metrics
         print ('MAE:',metrics.mean_absolute_error(y_test,y_pred))

         MAE: 2.485047305973559

In [59]: print(('MSE:',metrics.mean_squared_error(y_test,y_pred)))

         ('MSE:', 7.098694104379742)

In [60]: print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))

         RMSE: 2.6643374606794357

In [61]: metrics.r2_score(y_test,y_pred)

Out[61]: -88.41375190309928
```

jupyter analysis Last Checkpoint: an hour ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
analysis (3).ipynb
Show all

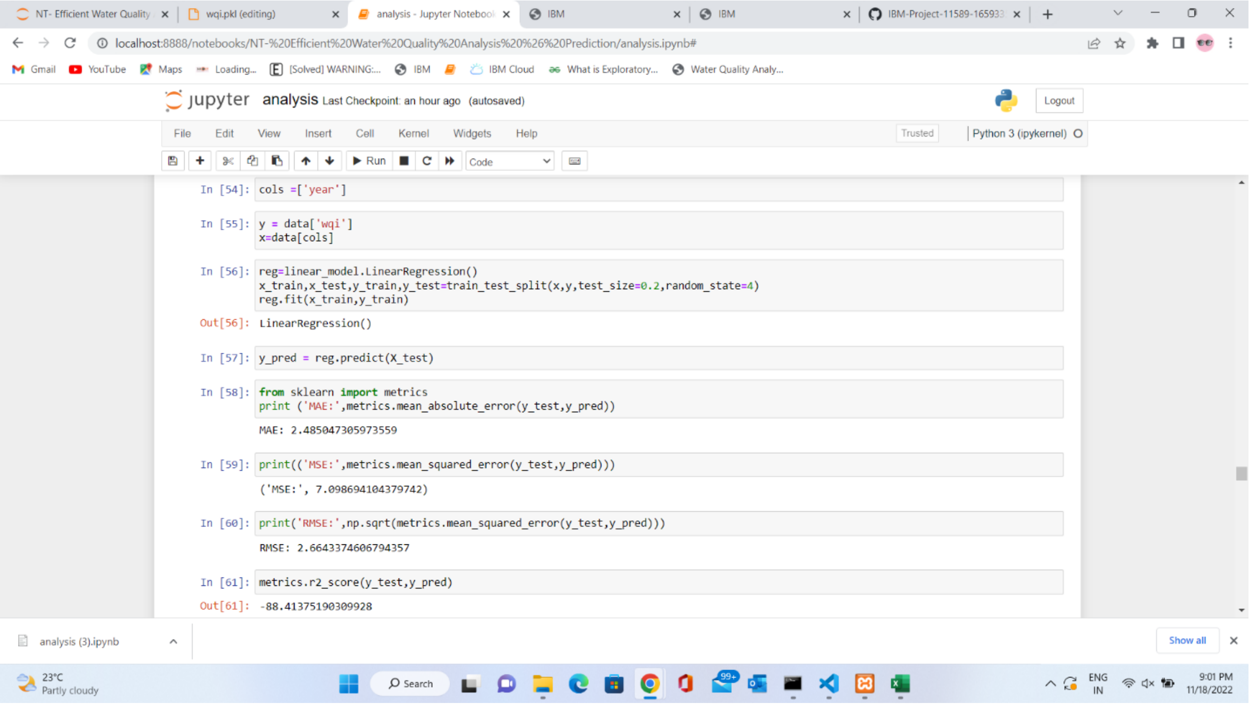
2. Tune the Model

Hyperparameter Tuning:

- The number of features is important and should be tuned in linear regression
- Initially all parameters in the dataset are taken as independent values to arrive at the dependent decision of Exploratory Analysis of Water Quality Prediction
- But the result was not accurate so used only 8 more correlated values as independent values to arrive at the dependent decision of Exploratory Analysis of Water Quality Prediction.

Validation Method:

- It involves partitioning the training data set into subsets, where one subset is held out to test the performance of the model.
- This data set is called the validation data set. Cross validation is to use different models and identify the best: Linear Regression Model performance values:



```
In [54]: cols =['year']

In [55]: y = data['wqi']
         x=data[cols]

In [56]: reg=linear_model.LinearRegression()
         x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=4)
         reg.fit(x_train,y_train)

Out[56]: LinearRegression()

In [57]: y_pred = reg.predict(X_test)

In [58]: from sklearn import metrics
         print ('MAE:',metrics.mean_absolute_error(y_test,y_pred))

         MAE: 2.485047305973559

In [59]: print(('MSE:',metrics.mean_squared_error(y_test,y_pred)))

         ('MSE:', 7.098694104379742)

In [60]: print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,y_pred)))

         RMSE: 2.6643374606794357

In [61]: metrics.r2_score(y_test,y_pred)

Out[61]: -88.41375190309928
```