In [5]:

```
import xlrd
import xlwt
import xlsxwriter
import numpy as np
import matplotlib.pyplot as plt
# %matplotlib inline
import neurolab as nl
import pandas as pd
import seaborn as sns
import os
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.model_selection import cross_val_score, cross_val_predict
from sklearn import metrics
import time
```

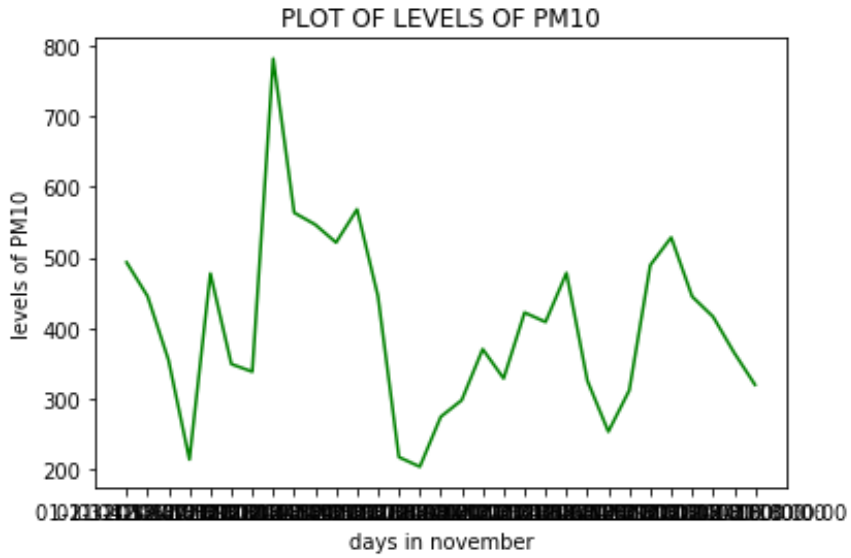
In [7]:

```
#data extraction

def extract(fname):
    wb = xlrd.open_workbook(fname)
    sheet = wb.sheet_by_index(0)

    ind_comp = []
    temp = []

    for i in range(sheet.nrows):
        for j in range(sheet.ncols):
            str1 = sheet.cell_value(i, j)
            str1 = str(str1).upper()
            if (str1.__contains__("FROM DATE")):
                ind_comp = [i, j]
                temp.append(ind_comp)

    count = 0
    start_row = temp[0][0] + 1
    head_col = []
    temp_list = []
    for i in temp:
        for j in range(sheet.ncols):
            count = i[0]
            str1 = sheet.cell_value(count, j)
            str1 = str(str1).upper()
            if (str1 != ''):
                head_col.append([j, str1])
    temp.append([sheet.nrows, 0])
    data_dict = {}
    for k in head_col:
        data_dict[k[1]] = []

    for i in range(len(temp) - 1):

        start = temp[i][0]
        end = temp[i + 1][0]
        for j in range(sheet.ncols):
            rl = []
            if (sheet.cell_value(start, j) != ""):
                name = str(sheet.cell_value(start, j)).upper()
                for k in range(start + 1, end):
                    str2 = str(sheet.cell_value(k, j)).upper()
                    if str2 != '':
                        rl.append(str(sheet.cell_value(k, j)).upper())
                    else:
                        break
                data_dict[name] = rl



    workbook = xlsxwriter.Workbook("data_new.xlsx", {'strings_to_numbers': Tr
ue})
    worksheet = workbook.add_worksheet()
    row = 0
    col = 0
    for i in data_dict.keys():
```

```
        row = 0
        worksheet.write(row, col, i)
        for j in data_dict[i]:
            row += 1
            worksheet.write(row, col, j)
        col += 1
    workbook.close()

file_names=["1mar_april","2april_may","3may_june","4june_july","5july_aug","6
aug_sept","7sept_oct","8oct_nov"]
for i in file_names:

    extract("data/"+i+'.xlsx')
```

In [8]:

```
#PM10 level hike in november

data = pd.read_excel('data1.xlsx')
fd=list(data['FROM DATE'])
pm=list(data['PM10'])
plt.plot(fd, pm, color='g')
plt.xlabel('days in november')
plt.ylabel('levels of PM10')
plt.title('PLOT OF LEVELS OF PM10')
plt.show()
```

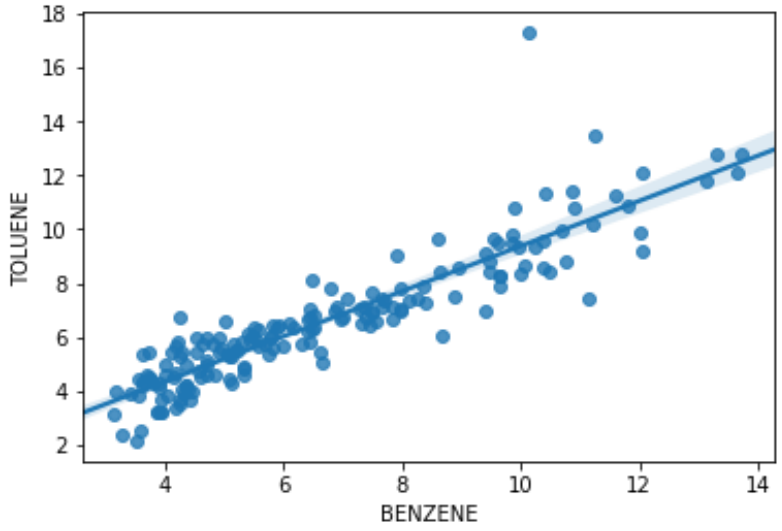


In [12]:

```
#

ls = ['BENZENE','TOLUENE','NOX','CO']
grant =[]


train1 = pd.read_excel('dt2_train.xlsx')
test1 = pd.read_excel('dt2_test.xlsx')
# print(train)

train = train1[['BENZENE','TOLUENE','NOX','CO']].copy()
test = test1[['BENZENE','TOLUENE','NOX','CO']].copy()



for i in ls:
    for j in ls:


        if not j in grant:
            if (i!=j):
                grant.append(i)
                d1 = i
                d2 = j

                train[d1] = train[d1].apply(np.sqrt)
                train[d2] = train[d2].apply(np.sqrt)
                test[d1] = test[d1].apply(np.sqrt)
                test[d2] = test[d2].apply(np.sqrt)

                x_train = train[d1]
                y_train = train[d2]
                x_test = test[d1]
                y_test = test[d2]

                x_train = np.array(x_train)
                y_train = np.array(y_train)
                x_test = np.array(x_test)
                y_test = np.array(y_test)

                x_train = x_train.reshape(-1, 1)
                x_test = x_test.reshape(-1, 1)


                clf = LinearRegression(normalize=True)
                clf.fit(x_train, y_train)
                y_pred = clf.predict(x_test)
                r2 = r2_score(y_test, y_pred)


                print('\nLinear model R2 value for ',i,' and ',j,' : ', r2,'
\n')

                plt.clf()
                plt.cla()
                sns.regplot(y_test, y_pred)
                plt.xlabel(i)
```

```
                plt.ylabel(j)
                plt.show()



                scores = cross_val_score(clf, x_train, y_train, cv=10)


                print("Scores from cross validation:", scores)
                print ('\n')
                print("Mean:", scores.mean(),'\n')
                print("Standard deviation:", scores.std())
                print('##########################################################
####################################\n\n')
```

Linear model R2 value for BENZENE and TOLUENE : 0.8176179835
677296

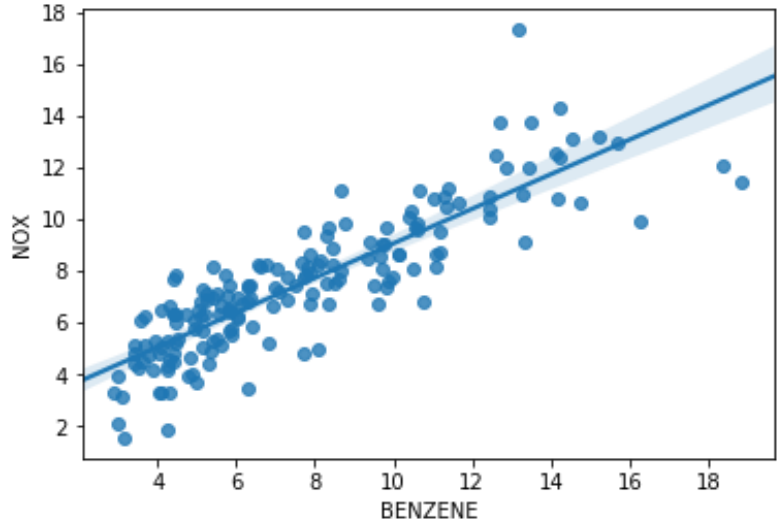


Scores from cross validation: [0.73300085 0.82281981 0.84113478
0.89587442 0.86982661 0.84288755
 0.87361323 0.83562003 0.89168104 0.88614707]

Mean: 0.8492605390850343

Standard deviation: 0.04566959683792611
##
########################

Linear model R2 value for BENZENE and NOX : 0.73483358586869
58

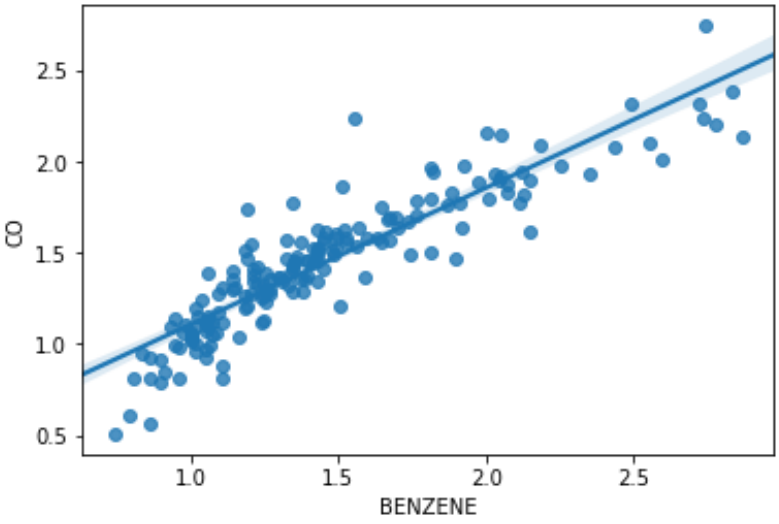


Scores from cross validation: [0.62248177 0.67006378 0.69618388
0.78836117 0.78198392 0.72072513
 0.79546237 0.7139583 0.7788766 0.82954362]

Mean: 0.7397640533160024

Standard deviation: 0.06194963422000329
##
########################

Linear model R2 value for BENZENE and CO : 0.814307734698559
5

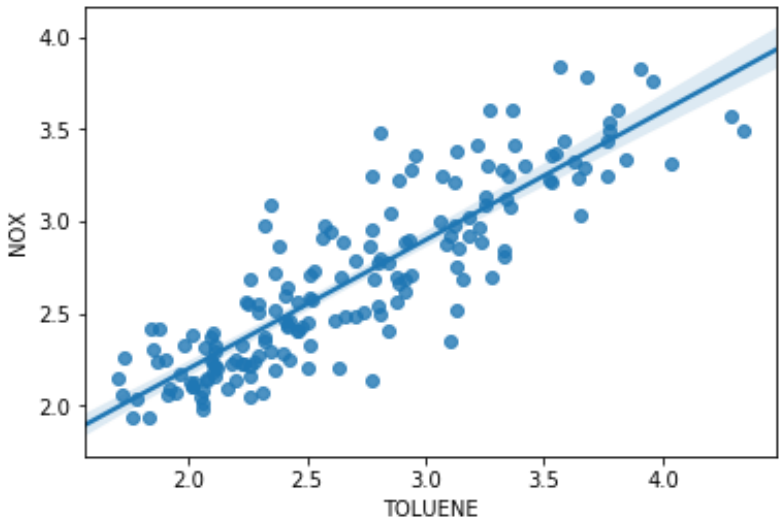


Scores from cross validation: [0.5521904 0.71549029 0.86028693
0.86945061 0.81538275 0.42589965
 0.45604983 0.46651725 0.84423513 0.83320185]

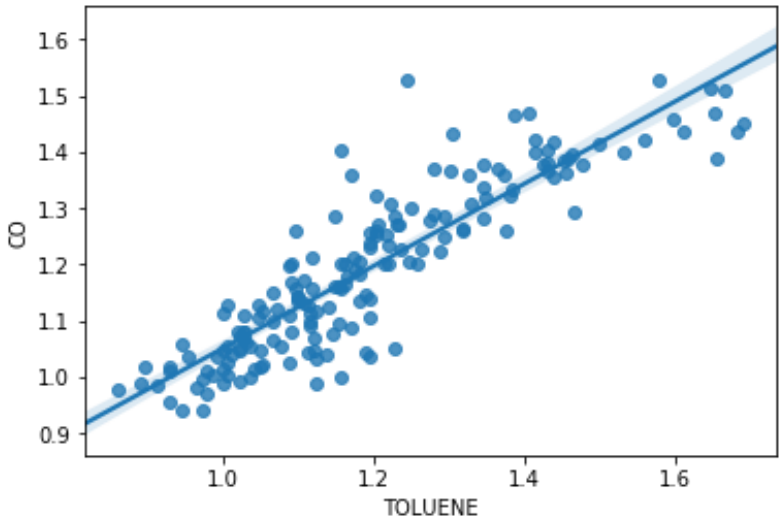
Mean: 0.6838704687085431

Standard deviation: 0.17745646982861152
##
########################

Linear model R2 value for TOLUENE and NOX : 0.75711860989320
79




```
Scores from cross validation: [0.65094959 0.6619149  0.64839459
0.70775771 0.80629543 0.81001456
 0.79731649 0.67950749 0.77170168 0.75589908]


Mean: 0.7289751531680901

Standard deviation: 0.06304151571464923
#########################################################################
#########################


Linear model R2 value for  TOLUENE  and  CO  :  0.788324539948127
2
```

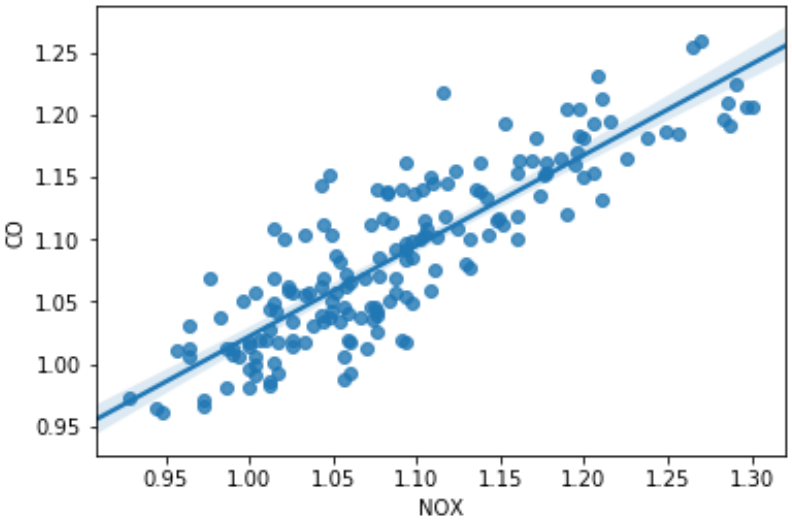


Scores from cross validation: [0.79090165 0.81851605 0.84005439
0.82953031 0.79969067 0.31551629
 0.3573407 0.15237641 0.83824551 0.84510607]

Mean: 0.6587278055171779

Standard deviation: 0.25630237884331014
##
########################

Linear model R2 value for NOX and CO : 0.7465323346390835



Scores from cross validation: [0.62966926 0.67099565 0.77510037
0.72210085 0.76735423 0.31614615
 0.22709787 0.15646837 0.76778404 0.76409527]

Mean: 0.5796812066678888

Standard deviation: 0.23383677379542211
##
########################