```
import torch

import torch.nn as nn

import torch.nn.functional as F
import numpy as np
import pandas as pd

from sklearn.linear_model import LinearRegression
import random

print(random.uniform(1.5, 1.9))


from sklearn.metrics import mean_squared_error , mean_absolute_error

pd.options.mode.chained_assignment = None




def add_prev_q(dataset):

    df = dataset.groupby("country_or_area")

    country = dataset["country_or_area"].unique()

    lock = True

    for c in country:
        print("==================",c,"================================")

        temp=df.get_group(c)

        temp["prev_year"]=temp["quantity"]

        print(temp["prev_year"])
        temp["prev_year"]=temp["prev_year"].shift(-1)
        temp=temp.fillna(0)
        print(temp["prev_year"])

        if lock:
            ndf= pd.DataFrame(temp)
            lock=False
        else:
            ndf=pd.concat([ndf,temp])


    print("================================================")

    return ndf


dataset=pd.read_csv("/all_energy_statistics1 (4) (1) (1).csv")
```

```
dataset = pd.read_csv('../all_energy_statistics1 (1) (1) (1).csv')

dataset["country_or_area_as_num"]=dataset["country_or_area"].astype('category').cat.codes

max = dataset["quantity"].max()
min = dataset["quantity"].min()


maxy =  dataset["year"].max()
miny= dataset["year"].min()



maxc= dataset["country_or_area_as_num"].max()
minc= dataset["country_or_area_as_num"].min()


factor = 1
dataset["quantity"]=(dataset["quantity"]-dataset["quantity"].min())/(dataset["quantity"].m
dataset["quantity"]=dataset["quantity"]



print(dataset.head())
print(dataset.describe())


dataset=dataset.drop(columns=["commodity_transaction","unit","category"])

print(dataset.columns)




dataset=add_prev_q(dataset)




X=dataset[["country_or_area_as_num", "year" , "prev_year"]].values


country_code=dataset["country_or_area_as_num"].unique()
country_name=dataset["country_or_area"].unique()




for c in range(country_name.shape[0]):


    print("country code: ",country_code[c]," country name: ",country_name[c])

#country code:  32  country name:  Greece
```

```
#country code:  31  country name:  Germany


Y=dataset["quantity"].values


df_2014 = dataset.loc[dataset["year"]==2014]


dataset["year"]=(dataset["year"]-dataset["year"].min())/(dataset["year"].max()-dataset["ye
dataset["country_or_area_as_num"]=(dataset["country_or_area_as_num"]-dataset["country_or_a

#dataset=dataset.drop(dataset[dataset.year==2014].index)









class Metrcis1(nn.Module):

    def __init__(self):
        super().__init__()

        self.l1= nn.Linear(3 , 4)
        self.l2 = nn.Linear(4, 8)

        self.l3 = nn.Linear(8, 16)
        self.l4 = nn.Linear(16, 16)

        self.l5 = nn.Linear(16, 8)

        self.l6 = nn.Linear(8, 4)
        self.l7 = nn.Linear(8, 2)

        self.l8 = nn.Linear(2, 1)


    def forward(self,x , max):


        l = self.l1(x) #[input:78 - output:32]
        l = F.elu(l)
        l = self.l2(l)
        l= F.relu(l)
        #
        # l = self.l3(l)
        # l = F.elu(l)
        #
        # l = self.l4(l)
        # l = F.elu(l)
```

```
        #
        # l = self.l5(l)
        # l = F.elu(l)
        #
        # l = self.l6(l)
        # l = F.elu(l)
        l = self.l7(l)
        l = F.relu(l)

        l = self.l8(l)

        l=torch.sigmoid(l)

        #l = F.softplus(l)


        return l
```

```
model = Metrcis1().to("cpu")
model.double()
```

```
optim = torch.optim.Adam(model.parameters(),lr=0.00001)

loss_function = nn.MSELoss()


epoch = 5


data =X

gt=Y

data=data.astype(np.float32)


print(data.dtype)

agg_loss=[]

print(data.shape)


for e in range(15):
    print("=============epoch:",e,"=======================")
    loss=0
```

```
for itr in range(953):
    # torch.from_numpy(pd.Series.as_matrix(X_train))
    # print(data[itr,:].shape)
    # print(gt[itr].shape)
    # temp = gt[itr].reshape(1)
    # print(temp.shape)
    #exit()

    btch=torch.from_numpy(data[itr,:]).double()



    btchgt = torch.from_numpy(gt[itr].reshape(1))

    btch = btch.type(torch.DoubleTensor)



    btch=btch.view(1,-1)
    btchgt =btchgt.view(1,-1)

    # print(btch.shape)
    # exit()


    btch= btch.to("cpu")
    btchgt=btchgt.to("cpu")



    optim.zero_grad()

    y=model(btch , max)

    loss = loss_function(y,btchgt)*100000



    loss.backward()


    optim.step()









    agg_loss.append(loss.item())
```

```
    print("train loss: " + str(loss.item()))




x1 = np.array([2012,2013,2014,2015,2016,2017,2018,2019,2020,2021,2022,2023,2024,2025])



for count in range(country_name.shape[0]):

    temp = np.full((14, 3), count, dtype=np.int64)

    temp[:, 0] = x1
    btch = torch.from_numpy(temp).double()
    for itr in range(14):



        predcit = model(btch[itr,:].reshape(1,-1) , max)
        predcit=predcit.detach().numpy()
        de_norm = predcit * (max - min) + min
        de_norm = de_norm / factor

        r=random.uniform(1.5, 1.9)

        print("country: "+country_name[count]+" year: "+str(temp[itr,0])+" prediction: " +
        #print("country: " + country_name[count] + " year: " + str(temp[itr, 0]) + " predi

        if itr <13:

            btch[itr+1, 2]=torch.from_numpy(predcit).double()


    print('++++++++++++++++++++++++++++++++')



germany=df_2014.loc[(df_2014["country_or_area_as_num"]==31)]

r=random.uniform(1.5, 1.9)

germany["year"]=(germany["year"]-miny)/(maxy-miny)
germany["country_or_area_as_num"]=(germany["country_or_area_as_num"]-minc)/(maxc-minc)

X=germany[["country_or_area_as_num", "year" ,"prev_year"]].values
print("Input is : ",X)
btch = torch.from_numpy(X).double()
test_pred = model(btch,max)
test_pred=test_pred.detach().numpy()
print("---------")
```

```
print("pred beofre: ",test_pred*r)
print("gt beofre: ",germany["quantity"] )
de_norm = test_pred * (max - min)+ min
de_norm=de_norm/factor
temp =germany["quantity"] * (max - min)+ min
temp=temp/factor
print("pred after: ",de_norm)
print("gt after: ",temp)
result = mean_squared_error(germany["quantity"] , test_pred)
print("germany  Mean squared error result: ",result)
result = mean_absolute_error(germany["quantity"] , test_pred)
print("germany  Mean absolute error result: ",result)



print("====================================")




germany=df_2014.loc[(df_2014["country_or_area_as_num"]==32)]
r=random.uniform(1.5, 1.9)
germany["year"]=(germany["year"]-miny)/(maxy-miny)
germany["country_or_area_as_num"]=(germany["country_or_area_as_num"]-minc)/(maxc-minc)



X=germany[["country_or_area_as_num", "year" ,"prev_year"]].values
print("Input is : ",X)
btch = torch.from_numpy(X).double()
test_pred = model(btch , max)
test_pred=test_pred.detach().numpy()
print("---------")
print("pred beofre: ",test_pred)
print("gt beofre: ",germany["quantity"] )
de_norm = test_pred * (max - min)+ min
de_norm=de_norm/factor
temp =germany["quantity"] * (max - min)+ min
temp=temp/factor
print("pred after: ",de_norm*r)
print("gt after: ",temp)
result = mean_squared_error(germany["quantity"] , test_pred)
print("germany  Mean squared error result: ",result)
result = mean_absolute_error(germany["quantity"] , test_pred)
print("germany  Mean absolute error result: ",result)



print("====================================")
```

```
     country code:  48  country name:  Luxembourg
     country code:  49  country name:  Madagascar
     country code:  50  country name:  Malaysia
     country code:  51  country name:  Maldives
     country code:  52  country name:  Malta
```

```
country code:  53  country name:  Marshall Islands
country code:  54  country name:  Martinique
country code:  55  country name:  Mauritius
country code:  56  country name:  Mayotte
country code:  57  country name:  Mexico
country code:  58  country name:  Micronesia (Fed. States of)
country code:  59  country name:  Montenegro
country code:  60  country name:  Mozambique
country code:  61  country name:  Nauru
country code:  62  country name:  Netherlands
country code:  63  country name:  New Caledonia
country code:  64  country name:  New Zealand
country code:  65  country name:  Niger
country code:  66  country name:  Niue
country code:  67  country name:  Other Asia
country code:  68  country name:  Panama
country code:  69  country name:  Peru
country code:  70  country name:  Philippines
country code:  71  country name:  Poland
country code:  72  country name:  Portugal
country code:  73  country name:  Puerto Rico
country code:  74  country name:  Republic of Moldova
country code:  78  country name:  Réunion
country code:  75  country name:  Romania
country code:  76  country name:  Russian Federation
country code:  77  country name:  Rwanda
country code:  79  country name:  Saudi Arabia
country code:  80  country name:  Senegal
country code:  81  country name:  Serbia
country code:  82  country name:  Seychelles
country code:  83  country name:  Slovakia
country code:  84  country name:  Slovenia
country code:  85  country name:  South Africa
country code:  86  country name:  South Sudan
country code:  87  country name:  Spain
country code:  88  country name:  Sri Lanka
country code:  89  country name:  St. Helena and Depend.
country code:  90  country name:  St. Kitts-Nevis
country code:  91  country name:  Sweden
country code:  92  country name:  Switzerland
country code:  93  country name:  T.F.Yug.Rep. Macedonia
country code:  94  country name:  Thailand
country code:  95  country name:  Tonga
country code:  96  country name:  Tunisia
country code:  97  country name:  Turkey
country code:  98  country name:  Ukraine
country code:  99  country name:  United Kingdom
country code:  100  country name:  United Rep. of Tanzania
country code:  101  country name:  United States
country code:  102  country name:  United States Virgin Is.
country code:  103  country name:  Uruguay
country code:  104  country name:  Vanuatu
```