```
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
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import OrdinalEncoder
from sklearn.metrics import mean_squared_error , mean_absolute_error

pd.options.mode.chained_assignment = None


# prepare input data
def prepare_inputs(X_train, X_test):
    oe = OrdinalEncoder()
    oe.fit(X_train)
    X_train_enc = oe.transform(X_train)
    X_test_enc = oe.transform(X_test)
    return X_train_enc, X_test_enc


# prepare target
def prepare_targets(y_train, y_test):
    le = LabelEncoder()
    le.fit(y_train)
    y_train_enc = le.transform(y_train)
    y_test_enc = le.transform(y_test)
    return y_train_enc, y_test_enc



dataset=pd.read_csv("/content/all_energy_statistics1 (4) (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"].round(3)


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



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

print(X)
x1 ,x2=prepare_inputs(X,X)

print(x1)



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)




print(X.shape)
print(Y.shape)

print(type(Y))
print(Y.dtype)

print(Y[0:4])
```

```
class Metrcis1(nn.Module):

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

        self.l1= nn.Linear(2 , 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(4, 2)

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


    def forward(self,x):


        l = self.l1(x) #[input:78 - output:32]
        l = F.elu(l)
        l = self.l2(l)
        l= F.elu(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.elu(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=0.0001)

loss_function = nn.L1Loss()


epoch = 5


data =X

gt=Y


data=data.astype(np.float32)


print(data.dtype)

agg_loss=[]

print(data.shape)


for e in range(25):
    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)

        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, 2), 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))
        predcit=predcit.detach().numpy()
        de_norm = predcit * (max - min) + min
        de_norm = de_norm / factor

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

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




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


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"]].values
print("Input is : ",X)
btch = torch.from_numpy(X).double()
test_pred = model(btch)
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)
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)]
```

```
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"]].values
print("Input is : ",X)
btch = torch.from_numpy(X).double()
test_pred = model(btch)
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)
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: United States Virgin Is. year: 2021 prediction: [[45553.2676892]]
country: United States Virgin Is. year: 2022 prediction: [[45571.14200894]]
country: United States Virgin Is. year: 2023 prediction: [[45589.00730139]]
country: United States Virgin Is. year: 2024 prediction: [[45606.86355842]]
country: United States Virgin Is. year: 2025 prediction: [[45624.71077186]]
++++++++++++++++++++++++++++++++
country: Uruguay year: 2012 prediction: [[45260.53465748]]
country: Uruguay year: 2013 prediction: [[45278.53589867]]
country: Uruguay year: 2014 prediction: [[45296.52834274]]
country: Uruguay year: 2015 prediction: [[45314.51198243]]
country: Uruguay year: 2016 prediction: [[45332.4868104]]
country: Uruguay year: 2017 prediction: [[45350.4528193]]
country: Uruguay year: 2018 prediction: [[45368.41000172]]
country: Uruguay year: 2019 prediction: [[45386.35835019]]
country: Uruguay year: 2020 prediction: [[45404.29785723]]
country: Uruguay year: 2021 prediction: [[45422.22851529]]
country: Uruguay year: 2022 prediction: [[45440.1503168]]
country: Uruguay year: 2023 prediction: [[45458.06325414]]
country: Uruguay year: 2024 prediction: [[45475.96731966]]
country: Uruguay year: 2025 prediction: [[45493.86250567]]
++++++++++++++++++++++++++++++++
country: Vanuatu year: 2012 prediction: [[45129.23445779]]
country: Vanuatu year: 2013 prediction: [[45147.27825575]]
country: Vanuatu year: 2014 prediction: [[45165.31340457]]
country: Vanuatu year: 2015 prediction: [[45183.33989781]]
country: Vanuatu year: 2016 prediction: [[45201.357729]]
country: Vanuatu year: 2017 prediction: [[45219.36689159]]
country: Vanuatu year: 2018 prediction: [[45237.36737896]]
country: Vanuatu year: 2019 prediction: [[45255.35918443]]
country: Vanuatu year: 2020 prediction: [[45273.34230128]]
```

```
country: Vanuatu year: 2021 prediction: [[45291.3167227]]
country: Vanuatu year: 2022 prediction: [[45309.28244185]]
country: Vanuatu year: 2023 prediction: [[45327.23945181]]
country: Vanuatu year: 2024 prediction: [[45345.18774563]]
country: Vanuatu year: 2025 prediction: [[45363.12731629]]
++++++++++++++++++++++++++++++++
Input is :  [[0.29807692 1.        ]]
---------
pred beofre:  [[0.55640693]]
gt beofre:  281    1.0
Name: quantity, dtype: float64
pred after:  [[20061.80817806]]
gt after:  281    36056.0
Name: quantity, dtype: float64
germany  Mean squared error result:  0.19677481396714538
germany  Mean absolute error result:  0.44359307249679336
==================================
Input is :  [[0.30769231 1.        ]]
---------
pred beofre:  [[0.55639643]]
gt beofre:  305    0.105
Name: quantity, dtype: float64
pred after:  [[20061.42951363]]
gt after:  305    3785.88
Name: quantity, dtype: float64
germany  Mean squared error result:  0.2037587328489651
germany  Mean absolute error result:  0.4513964253834595
==================================
```

## New Section

```
from google.colab import drive
drive.mount('/content/drive')
```

✓ 27s completed at 4:17 AM