

# **Configuration Manual**

Analysis and predictions of CO2 emissions using Neural Networks

Msc Data Analytics

Jeet Jaikishan Vyas

Student ID: x19197161

School of Computing

National College of Ireland

Supervisor: Prof. Jorge Basilio



### National College of Ireland

### **MSc Project Submission Sheet**

### School of Computing

Student Name:	Jeet Jaikishan Vyas				
Student ID:	X19197161				
Programme:	Msc Data Analytics Year: 2021				
Module:	Research Project				
Lecturer: Submission					
Due Date:	16/12/2021				
Project Title:	Analysis and prediction of CO2 emissions us	ng deep learning			

Word Count: 1736 Page Count: 11

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

<u>ALL</u> internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Jeet Jaikishan Vyas

**Date:** 16/12/2021

### PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

### **Office Use Only**

Office use offig	
Signature:	
Date:	
Penalty Applied (if applicable):	

# **Configuration Manual**

### Jeet Jaikishan Vyas Student ID: x19197161

# **1** Introduction

The configuration manual document will state the required software and hardware required for the research project "Analysis and predictions of CO2 emission using Neural Networks". The manual contains the code for the Data collection/cleaning, clustering and models.

# 2 System and Software Requirements

Below are the required system configurations required for the research to be conducted.

2.1 Hardware Requirements:

Processor: Intel Core i5 – 7300U CPU @ 2.30 GHz to 2.40 GHz Storage Capacity: 1 TB (Terabyte) HDD Hard Disk System Type: 64-bit processor, x64 GPU: AMD Radeon / NVIDIA GeForce Operating System: Windows 10 or 11 (64-bit operating system) Ram: 8 GB

2.2 Software Requirements:

Programming and Data loading / Processing / Cleaning / Modelling: Jupyter Notebook by Anaconda.

Clustering: RStudio

Visualizations: Python / PowerBI.

Other tools: Microsoft word, Snipping tool, Microsoft excel.

Microsoft word are used for creation of tables and showcasing the figures. Snipping tool is used for getting the screenshot. Microsoft excel contains the data used for the research.

# **3** Data collection and cleaning

The flow of the research project is showcased below:

3.1 Collection of data and cleaning

The data used for the research in the format of CSV file has been presented. This showcases the raw data which has been used for the research implementation.

	А	В	С	D	E	F
1	CountryNa	CountryCo	IndicatorN	IndicatorC	Year	Value
2	Arab Worl	ARB	Adolescen	SP.ADO.TF	1960	133.5609
3	Arab Worl	ARB	Age depen	SP.POP.DP	1960	87.7976
4	Arab Worl	ARB	Age depen	SP.POP.DP	1960	6.634579
5	Arab Worl	ARB	Age depen	SP.POP.DP	1960	81.02333
6	Arab Worl	ARB	Arms expo	MS.MIL.XF	1960	3000000
7	Arab Worl	ARB	Arms impo	MS.MIL.M	1960	5.38E+08
8	Arab Worl	ARB	Birth rate,	SP.DYN.CE	1960	47.69789
9	Arab Worl	ARB	CO2 emiss	EN.ATM.C	1960	59563.99
10	Arab Worl	ARB	CO2 emiss	EN.ATM.C	1960	0.643964
11	Arab Worl	ARB	CO2 emiss	EN.ATM.C	1960	5.041292
12	Arab Worl	ARB	CO2 emiss	EN.ATM.C	1960	84.85147
13	Arab Worl	ARB	CO2 emiss	EN.ATM.C	1960	49541.71
14	Arab Worl	ARB	CO2 emiss	EN.ATM.C	1960	4.726981
15	Arab Worl	ARB	Death rate	SP.DYN.CE	1960	19.75445
16	Arab Worl	ARB	Fertility rat	SP.DYN.TF	1960	6.924027
17	Arab Worl	ARB	Fixed telep	IT.MLT.MA	1960	406833
18	Arab Worl	ARB	Fixed telep	IT.MLT.MA	1960	0.616701
19	Arab Worl	ARB	Hospital be	SH.MED.B	1960	1.929622
20	Arab Worl	ARB	Internation	SM.POP.TO	1960	2.990637
21	Arab Worl	ARB	Internation	SM.POP.TC	1960	3324685
22	Arab Worl	ARB	Life expect	SP.DYN.LE	1960	47.88325
23	Arab Worl	ARB	Life expect	SP.DYN.LE	1960	45.86295
24	Arab Worl	ARB	Life expect	SP.DYN.LE	1960	46.84706
25	Arab Worl	ARB	Merchandi	TX.VAL.MF	1960	4.65E+09

Figure 3.1: Raw data Indicators

To know about the data being used for the research, Figure 1 shows the comma separated (CSV) file for the indicators data. There are a total of 6 columns and 5656458 rows in the data. The data includes the countries name and code, indicators name and code, years and the value for the indicators.

n. Countr	B C D E vClShortNam TableNam LongNam Alpha2	F G H I J		L M N		P Q roi SystemOfi Alternativ Po		S T	U V	W	X	l IntertHou Ca	Z AA	AB	AC	AD	AE	AF
Lountr AFG	Afghanist Afghanist Islamic St AF	Afghan af Fiscal yea South Asi; Low incomeGr W02C	2002/03	Value adc IDA	HIPC	Country uses the 1993			General tr Consolid				tegrated househo		atestindi L	2013	2000	rw ithora
ALB	Albania Albania Republic (AL	Albanian lek Europe & Upper mic AL	-	1996 Value adc IBRD	niec										2011	2013	2000	
DZA	Algeria Algeria People's [DZ	Algerian dinar Middle Ea Upper mic DZ	Original c 1980	Value adc IBRD		Country uses the 19 Ro Country uses the 19			General ti Budgetan Special tr Budgetan			Demograp Liv	tegrated househo	2012	2011 2010	2015	2006	
ASM		U.S. dollar East Asia Upper mic AS	1980	Value auc IDRU		Country uses the 19 Country uses the 19 20					2008	Multiple ( In	Yes	2007	2010	2013	2001	
ADO	American American American AS Andorra Andorra Principali AD	Euro Europe & High inco AD				V Country uses the 19 20						ulation data		2007		2006		
GO			2000		asic prices	Country uses the 1968 Country u 1991Â-96										2006	2007	
	Angola Angola People's FAO	Angolan k April 201: Sub-Sahai Upper mic AO	2002	Value adc IBRD					Special tr Budgetar			Malaria i in	tegrated househo				2005	
TG	Antigua a Antigua a Antigua a AG	East Caril April 201: Latin Ame High inco AG	2006	Value adc IBRD		Country uses the 19	2011	IMF Balance of Pay	General ti Budgetar	General D	2011		Yes	2007		2013	2005	
RB	Arab Wor Arab Wor Arab Wor 1A	Arab World aggregate. Arab W 1A																
RG	Argentina Argentina Argentine AR	Argentine The base Latin Ame High inco AR	2004	Value adc IBRD		Country u 1971Â-84			Special tr Consolid			Multiple I Inf		2013	2002	2013	2011	
RM	Armenia Armenia Republic (AM	Armenian dram Europe & Lower mic AM	Original c	1996 Value adc IBRD		Country u 1990Â-95	-		Special tr Consolid			Demograt Inf		2013/14	2008	2013	2012	
BW	Aruba Aruba AW	Aruban fl SNA data Latin Ame High inco AW	2000			(V Country uses the 19		IMF Balance of Pay			2010		Yes			2012		
IS	Australia Australia Commony AU	Australia: Fiscal yea East Asia High inco AU		013/14 Value added at b				IMF Balance of Pay			2011		penditu Yes	2011	2011	2013	2000	
Π	Austria Austria Republic (AT	Euro A simple (Europe & High inco AT	Original c		asi Euro ari	ea Country uses the 20 Ro								2010	2010	2013	2002	
E	Azerbaija Azerbaija Republic (AZ	New Azeri April 201, Europe & Upper micAZ	2000	Value adc IBRD		Country u 1992Â-95		IMF Balar Actual				Demograt Liv		2015	2011	2013	2012	
IR.	Bahrain Bahrain Kingdom BH	Bahraini Based on Middle Ea High inco BH	2010		roducer pri	ce Country uses the 19		IMF Balance of Pay			2010		Yes		2010	2011	2003	
D	Banglade: Banglade: People's FBD	Banglade: Fiscal yea South Asi: Lower mic BD	2005/06	Value adc IDA		Country uses the 19		IMF Balar Prelimina					tegrated househo	2008		2007	2008	
В	Barbados Barbados Barbados BB	Barbados dollar Latin Ame High inco BB	1974		roducer pri	ice Country uses the 19		IMF Balance of Pay				Multiple Ind		2010. Popu	lation ar	2013	2005	
1	Belarus Belarus Republic BY	Belarusian rubel Europe & Upper mic BY	Original c	2000 Value adc IBRD		Country u 1990Â-95			General tr Consolid			Multiple I Inf	tegratec Yes		2011	2013	2000	
	Belgium Belgium Kingdom BE	Euro A simple r Europe & High inco BE	Original c	2010 Value added at b	asi Euro ar	ea Country uses the 20 Ro	olling	IMF Balance of Pay	Special tr Consolid	Special Di	2011	Int	tegratec Yes	2010	2010	2013	2007	
2	Belize Belize BZ	Belize dollar Latin Ame Upper mic BZ	2000	Value adc IBRD		Country uses the 19	2011	IMF Balar Actual	General ti Budgetan	General D			bor force survey (			2013	2000	
N	Benin Benin Republic (BJ	West African CFA fra Sub-Sahai Low incor BJ	2007	Value adc IDA	HIPC	Country u 1992	2011	IMF Balar Actual	Special tr Budgetar	General D	2013	Multiple I Co	re Welfare Indica	2011/12		2013	2001	
U	Bermuda Bermuda The Bermi BM	Bermuda dollar North Ami High inco BM	2006	Value added at b	asic prices	(V Country uses the 19	2011	IMF Balance of Pay	General trade syste	n	2010		Yes			2013		
1	Bhutan Bhutan Kingdom BT	Bhutanes: April 201: South Asi: Lower mic BT	2000	Value adc IDA		Country uses the 19	2011	IMF Balar Actual	General tr Consolid	General D	2005	Multiple I Inf	tegrated househo	2009		2011	2008	
L	Bolivia Bolivia Plurinatic BO	Bolivian Boliviano Latin Ame Lower mic BO	1990	Value adc Blend	HIPC	Country u 1960Â-85	2011	IMF Balar Actual	General tr Consolid	General D	2012	Demograt Inf	tegrated househo	2013		2013	2000	
ł	Bosnia ar Bosnia ar Bosnia ar BA	Bosnia ar Based on Europe & Upper mic BA	Original c	2010 Value adc IBRD		Country uses the 19 Ro	olling	IMF Balar Actual	Special tr Consolid	General D	2013	Multiple I Liv	ving Sta Yes			2013	2012	
A	Botswana Botswana Republic (BW	Botswana Fiscal yea Sub-Sahai Upper mic BW	2006	Value adc IBRD		Country uses the 19	2011	IMF Balar Estimate	General ti Budgetar	General D	2011	Multiple   Ex	penditure survey/	2011. Pop	2011	2013	2000	
A	Brazil Brazil Federativ BR	Brazilian real Latin Ame Upper mic BR	1995	2000 Value adc IBRD		Country uses the 19	2011	IMF Balar Actual	General tr Consolid	Special Di	2010	World He Int	tegrated househo	2006	2011	2013	2010	
N	Brunei Brunei Da Brunei Da BN	Brunei dollar East Asia High inco BN	2000	Value added at p	roducer pri	ce Country uses the 19	2011		Special trade system	General D	2011		Yes			2013	1994	
R	Bulgaria Bulgaria Republic (BG	Bulgarian The new r Europe & Upper micBG	Original c	2010 Value adc IBRD		Country u 1978Â-85 Ro	olling	IMF Balar Actual	Special tr Consolid	Special Di	2011	Living Sta Ex	penditu Yes	2010	2011	2013	2009	
A	Burkina F Burkina F Burkina F BF	West African CFA fri Sub-Sahai Low incon BF	1999	Value adc IDA	HIPC	Country u 1992Â-93	2011	IMF Balar Actual	General tr Budgetan	General D		and the second second	vre Welfare Indica	2010		2013	2005	
1	Burundi Burundi Republic BI	Burundi franc Sub-Sahai Low incor Bl	2005	Value adc IDA	HIPC	Country uses the 19		IMF Balar Actual	Special tr Consolid				re Welfare Indica		2010	2012	2000	
v	Cabo Verc Cabo Verc Republic (CV	Cabo Verc Cabo Verc Sub-Sahai Lower mic CV	2007	Value adc Blend		Country uses the 19	2011	IMF Balar Actual	General tr Budgetan	General D		Demograt Co		2014		2013	2001	
M	Cambodia Cambodia Kingdom (KH	Cambodian riel East Asia Low incor KH	2000	Value adc IDA		Country uses the 19		IMF Balar Actual	Special tr Budgetan				tegrated househo			2013	2006	
(R	Cameroor Cameroor Republic (CM	Central African CFA Sub-Sahai Lower mic CM	2000	Value adc Blend	HIPC	Country uses the 19		IMF Balar Actual	Special tr Budgetan				iority survey (PS).			2012	2000	
N	Canada Canada Canada CA	Canadian Fiscal yea North Am High inco CA	Original c			IV Country uses the 20	-	IMF Balance of Pay			2011		bor for Yes	2011	2011	2013	1986	
S	Caribbear Caribbear Caribbear S3	Caribbean small states aggree \$3	ongine (		and build			service of they	and a consolid	apresion DI							2000	
(M	Cavman I: Cavman I: Cavman I: KY	Cavman Islands dol Latin Ame High inco KY	2007			Country uses the 19	2011		General trade syste	m	2010		Yes					

Figure 3.2: Raw Data Countries

The data about the countries has been shown in Fig 2. The data contains 247 rows and 31 columns. The information about all the countries is distinguished in the data. The country names in three different columns, currency units for the countries, encoded country names, the population census and the industrial, water withdrawal, agricultural data. The total data shows the information that are relevant for a country.

```
1 import pandas as pd
2 import numpy as np
3 import seaborn as sns
4 import matplotlib.pyplot as plt
```

```
1 df_itr = pd.read_csv('E:\\NCI\\Semester 3\\Thesis\\Dataset\\Indicators.csv')
2 df_ctrs = pd.read_csv("E:\\NCI\\Semester 3\\Thesis\\Dataset\\Country.csv")
3 df_series = pd.read_csv("E:\\NCI\\Semester 3\\Thesis\\Dataset\\Series.csv")
```

Figure 3.3: Loading Libraries and Data Frame Creation

The raw data was loaded on the python environment by anaconda. The tables indicators, country and series were selected for the research. For the research, indicators and country data is being considered. The series data has been just used for information purposes. The pandas library in python used for data manipulation and analysis. The data should be loaded into data frames.

<pre>1 df_ctrs.isna().sum()</pre>		
CountryCode	Ø	
ShortName	0	
TableName	0	
LongName	0	
Alpha2Code	3	
CurrencyUnit	33	
SpecialNotes	83	
Region	33	
IncomeGroup	33	
Wb2Code	1	
NationalAccountsBaseYear	42	
NationalAccountsReferenceYear	193	
SnaPriceValuation	49	
LendingCategory	103	
OtherGroups	188	
SystemOfNationalAccounts	33	
AlternativeConversionFactor	200	
PppSurveyYear	56	
BalanceOfPaymentsManualInUse•	66	
ExternalDebtReportingStatus	123	
SystemOfTrade	47	
GovernmentAccountingConcept	86	
ImfDataDisseminationStandard	64	
LatestPopulationCensus	34	
LatestHouseholdSurvey	100	
SourceOfMostRecentIncomeAndExpenditureData	89	
VitalRegistrationComplete	135	
LatestAgriculturalCensus	105	
LatestIndustrialData	134	
LatestTradeData	61	
LatestWaterWithdrawalData	67	
dtype: int64		

Figure 3.4: Data cleaning for country data

1	df_itr.isn	a().sum()			
Cour	ntryName	0			
Cour	tryCode	0			
Indi	catorName	0			
Indi	catorCode	0			
Year		0			
Value Ø					
 - C		a in indiantana			

Figure 3.5: Sum of null values in indicators data.

### 3.1.1 Data Cleaning

To ensure the there are no null values in the data data cleaning techniques were implemented. The pandas library allows calculating the null values and filling the null values. Figure 3.3 shows the representation. The data frame was stored in a new data frame named countries which will be used for further research steps. On the other hand, after implementation there were no null values found in the indicators data.

1	df_itr.duplicated()
0	False
1	False
2	False
з	False
4	False
5656	5453 False
5656	5454 False
5656	5455 False
5656	5456 False
5656	5457 False
Leng	th: 5656458, dtype: bool
1	df_countries.duplicated()
0	False
1	False
2	False
з	False
4	False
242	
243	
244	
245	
246	
Leng	gth: 247, dtype: bool
r	
1	df_series.duplicated()
ø	False
1	False
2	False
з	False
4	Falco

Figure 3.5: Duplicated values evaluation on the data

In order to find any duplicates in the data, the duplicated function shows if there are any duplicates in the data.

# **4 Data Visualisations**

For the research, there were bar graphs visualized using the matplotlib library in python. Three indicators were chosen for this step. The indicators were CO2 emissions per metric tons per capita, Age dependency ratio, CO2 emissions from liquid fuel consumptions. Visualisations for 6 countries were implemented for the research. The countries for this are United states of America, Australia, Ireland, India, China and Ireland. The graphs showed the

trend of the indicators from the year 1960 - 2010. The behaviour of the indicators according to the countries shows the flow. The indicators have been named and the plot function for bar graphs should be implemented. The world data shows the overall values of the specified indicator.

The following images show the graphical representations of overall world for the three indicators, the representations for the 6 countries including the world are visualized in python.

```
indicator1 = 'CO2 emissions \(metric'
world0 = 'World'

    x = df_process['IndicatorName'].str.contains(indicator1)
    y = df_process['CountryName'].str.contains(world0)

    total = df_process[x & y]

    x = total['Year'].values
    y = total['Value'].values
    plt.title("CO2 Emissions metric ton per capita - World")
    plt.xlabel("Years")
    plt.ylabel("Emission range")
    plt.bar(x,y)
```

Figure 4.1: CO2 emissions metric tons per capita – World

The trends of CO2 emissions metric tons per capita had seen a fluctuating one. From 1960 there has been a rise in the emissions up to 1980. There was slight fall after which the trend was quite stable and a rise in the emissions were seen after 2000 which kept rising until 2010.

```
indicator2 = 'Age dependency ratio'
 1
      world0 =
                       'World
  2
  3
      x1 = df_process['IndicatorName'].str.contains(indicator2)
y1 = df_process['CountryName'].str.contains(world0)
  4
  5
  6
      total = df_process[x1 & y1]
 8
     x1 = total['Year'].values
y1 = total['Value'].values
plt.title("Age dependency ratio - World")
plt.xlabel("Years")
plt.ylabel("Total %")
 9
10
11
12
13
14 plt.bar(x1,y1)
15 plt.show()
```

Figure 4.2: Age dependency Ratio – World

The age dependency ratio follows the formula for the number of people engaged in the working class compared to the people not engaged in the working class. The people distinguished in the working population are between the age group of 18 - 59 comparing the people in age group of 0 - 17 and 59 and above. The classes are defined as minors and senior citizens or retired individuals. The indicator should be considered as an effect on the economy of an individual country.

```
indicator3 = 'CO2 emissions from liquid fuel consumption'
 1
     world0 = 'World'
 2
 3
    x2 = df_process['IndicatorName'].str.contains(indicator3)
y2 = df_process['CountryName'].str.contains(world0)
 4
 5
 6
 7
    total = df process[x2 & y2]
 8
    x2 = total['Year'].values
y2 = total['Value'].values
plt.title("CO2 Emissions liquid fuel consumption - World")
 9
10
11
    plt.xlabel("Years")
12
    plt.ylabel("Emission range")
13
14
    plt.bar(x2,y2)
15 plt.show()
```

The code showcases the indicator liquid fuel consumptions for the world. The overall trends represent the indicator name, country name includes the world and the value for the same.

### **5** Clustering

```
library(readr)
library(reshape)
library(cluster)
 3
 45
      Indicators <- read.csv("E:\\NCI\\Semester 3\\Thesis\\Dataset\\Indicators.csv")</pre>
     6789
10
11
12
13
14
15
      indic <- cast(Indicators, CountryName ~ IndicatorCode, mean)
      indic <- cast(Indicators, countryName ~ IndicatorCode,
indic <- indic[,c('CountryName',code)]
row.names(indic) <- indic[,1]
indic <- indic[,c(-1,-2,-4,-6,-9,-14,-17,-19,-25,-29)]
indic <- na.omit(indic)</pre>
16
18
19
20
     Cnames <- rownames(indic)
21
22
23
24
25
      indic.norm <-</pre>
                             sapply(indic, scale)
      row.names(indic.norm) <- Cnames
      set.seed(42)
26
27
28
29
      indic_cluster <- kmeans(indic.norm, 3)
indic_cluster$size</pre>
      par(mfrow=c(1 \ 1))
      clusplot(indic.norm,indic_cluster$cluster,color = TRUE, shade = TRUE, labels = 2, lines = 0)
30
31
      indic.norm <- data.frame(cbind(indic_cluster$cluster, rownames(indic), indic.norm))
colnames(indic.norm) <- c("Cluster_No", "CountryName", colnames(indic.norm)[c(-1,-2)])
rownames(indic.norm) <- NULL</pre>
32
     indic.norm -
34
35
36
37
     developed_countries <- subset(indic.norm, Cluster_No == 1, select = CountryName)
developing_countries <- subset(indic.norm, Cluster_No == 2, select = CountryName)
underdeveloped_countries <- subset(indic.norm, Cluster_No == 3, select = CountryName)</pre>
38
39
     print(developed_countries)
print(developing_countries)
print(underdeveloped_countr
40
41
```

#### Figure 5: Clustering of Countries

The aim for clustering was to develop clusters of the countries in the indicators data. For this necessary step were carried in RStudio. The libraries for conducting are readr, reshape and cluster. The readr library helps in reading the CSV format files, reshape library helps in reshapes a data frame between wide and long formats in order to achieve repeated measurements in separate columns. The cluster library helps in getting the clusters using the algorithms on a specific data. The indicator data loaded in the environment as the first initial step. The indicator codes should be taken into consideration for creation of clusters. 31 indicators were chosen for the same. The transformation of data carried out by changing the formation of rows to columns. The cast function used to get the aggregate data with the aggregate function and formula. In this case, the relationship between the columns using the tilde function. The transformation makes the data ready to standardise and apply the algorithm for clustering. The sapply function takes the data frame as an input and the potential outcome will be a vector or matrix, therefore the data (indic) has been input as an object along with the function of scale. for the same to get a vector or matrix.

The K means clustering algorithm has been used for the cluster implementation. As KMeans being the most simple and quick clustering algorithm as it creates clusters with the mean

value close to the one feature to another. This helps in making a sophisticated form of clustering. Three clusters were formed as the KMean value was set at 3. Plotting the cluster created three clusters and the information about the clusters with the countries in the appropriate countries will be achieved.

## **6** Time Series and Models

```
1 from tqdm import tqdm
2 from time import time
1 import torch
2 import torch.nn as nn
3 import torch.optim as optim
4 import torch.nn.functional as f
1 from torch.utils.data import Dataset, DataLoader, TensorDataset
```

#### Figure 6: Libraries and utils

The libraries required to get the models using Pytorch. nn is the module for neural network models. Optim is the module which helps in optimizing the models deployed. Nn functional involves all functions which are applied on models. Tqdm library is used to show the progress bar.

```
1 def get valid range(country code, ECON COLS, CO2 COL, time step=5, verbose=False):
3
      try:
         x = COUNTRIES_DATA[country_code].loc[ECON_COLS + [CO2_COL], :]
4
5
      except KeyError:
        if verbose:
6
           8
9
         return [], None
10
     y_econ = (x.loc[ECON_cols, :].isna().sum(axis=0) == 0)['Value']
11
12
     y_co2 = ~x.loc[CO2_COL, :].isna()['Value']
13
14
      has co2 = [year for year in y co2[y co2].index]
     15
16
17
18
      y = y_econ & y_co2
19
     is_valid = [year if y[year] else 0 for year in y.index]
20
21
     res = continuous_subarrays(is_valid, step=1, min_len=time_step)
22
     if verbose and len(res) == 0:
         print('{} - {} has no valid interval.'.format(country_code, CODE_TO_NAMES[country code]))
23
24
25
     return res, x
```

Figure 6: Time series Data processing

In order to get the data ready for implementation of time series, a valid range of data need to be prepared which will withstand the factors while implementing the time series. Here, the country code (countries data), ECON\_COLS are the economic indicator columns, CO2\_COL are the CO2 emission indicator columns which are the target features for implementation. There has to be a set time step that will consider the time interval for the time series. Any null

values have been ignored. A continuous subarray created for validating the time step with the length of the same. The countries data and time step were returned.

```
1 def CO2TimeSeries(ECON_COLS, CO2_COL, time_step, valid_pct=0.2):
 2
        train data, val data = None, None
 4
        train_targets, val_targets = np.array([]), np.array([])
 5
 6
        for country in df_process['CountryCode'].unique():
 7
 8
           valid_ranges, df_countries = get_valid_range(country, ECON_COLS, CO2_COL, time_step)
 9
           if len(valid_ranges) == 0:
 10
               continue
 11
           for indicator in df_countries.index:
 12
               if indicator != CO2_COL:
 13
                   df_countries.loc[indicator, :] = \
 14
                       (df_countries.loc[indicator, :] - indicators_mean[indicator]) / indicators_std[indicator]
               else:
 17
                   df_countries.loc[indicator, :] = [np.log(num) if not np.isnan(num) else np.nan \
 18
                                                     for num in df_countries.loc[indicator, :]]
 19
 20
                   country_data, country_targets = None, np.array([])
           econ_df, co2_df = df_countries.loc[ECON_COLS, :], df_countries.loc[CO2_COL, :]
 21
 22
 23
           for interval in valid ranges:
 24
               for i in range(interval[0], interval[1] - time step + 1):
                   obs = econ_df.iloc[:, i:i+time_step].to_numpy()
 25
                   obs = obs[np.newaxis,:, :]
 26
 27
                   if country_data is not None:
 28
                       country_data = np.concatenate((country_data, obs), axis=0)
 29
                   else:
                      country_data = obs
 30
                   country_targets = np.append(country_targets, co2_df.iloc[i + time_step - 1])
 32
 33
           num_train = round((1 - valid_pct) * country_data.shape[0])
           country_train_data = country_data[:num_train,:,:]
 34
 35
            country_train_targets = country_targets[:num_train]
 36
            country_val_data = country_data[num_train:,:,:]
37
38
           country val targets = country targets[num train:]
39
             assert country_data.shape[0] == len(country_targets)
             assert len(country_val_targets) + len(country_train_targets) == country_data.shape[0]
40
             assert len(country val targets) == country val data.shape[0]
41
42
             assert len(country train targets) == country train data.shape[0]
43
44
             if train data is not None:
45
                  train_data = np.concatenate((train_data, country_train_data), axis=0)
46
                  val_data = np.concatenate((val_data, country_val_data), axis=0)
47
             else:
48
                 train_data = country_train_data
49
                  val data = country val data
50
             train_targets = np.append(train_targets, country_train_targets)
51
             val_targets = np.append(val_targets, country_val_targets)
52
53
        assert train data is not None,
         return TensorDataset(torch.tensor(train_data), torch.tensor(train_targets)),\
54
55
            TensorDataset(torch.tensor(val_data), torch.tensor(val_targets))
```

Figure 6.1: Time Series Implementation

Implementing the time series with the indicators and countries data. The train and validation were created using arrays in numpy library. The range of indicators if not null for economic and CO2 emissions will be considered. The second range of countries in valid range are not null then the data will concatenate else it will append. The same process was carried out to prepare the training and validation implementation.

```
1 train_ds, val_ds = CO2TimeSeries(ECON_cols, CO2_COL='EN.ATM.CO2E.KT', time_step=TIME_STEP, valid_pct=VALID_PCT)
2 print("Number of Training observations:", len(train_ds))
3 print("Number of Validation observations:", len(val_ds))
4
5 BATCH_SIZE = 32
6
7 train_dl = DataLoader(train_ds, batch_size=BATCH_SIZE, shuffle=True)
8 val_dl = DataLoader(val_ds, batch_size=BATCH_SIZE, shuffle=True)
```

Figure 6.2: Time series training and validation

The training and validation for the time series with the economic indicators and CO2 emissions metric ton per capita for CO2\_COL with the valid time step declared in the implementation phase. The batch size set to 32 shows the number of samples to be shown to the networks. The batch size used to fit the model and control the number of predictions made at a certain time.

```
1 class LSTM(nn.Module):
 2
             __init__(self, input_dims, hidden_dims, output_dims,):
 3
        def
            super().__init__()
self.rec_layer = nn.LSTM(input_dims, hidden_dims, batch_first=True)
 4
 5
            self.fc = nn.Linear(hidden_dims, output_dims)
 6
 7
 8
        def forward(self, x):
9
10
            x = self.reshape(x)
            outputs, (hidden_states, cell_states) = self.rec_layer(x)
11
            outputs = outputs[:, -1, :]
12
            outputs = outputs.squeeze(1)
13
            outputs = self.fc(outputs)
14
15
            return outputs.squeeze(1)
16
17
18
        def reshape(self, x):
19
            return torch.reshape(x, shape=(x.shape[0], x.shape[2], x.shape[1]))
```

#### Figure 6.3: 1Layer LSTM Model

```
1
   class GRU(nn.Module):
 2
             __init__(self, input_dims, hidden_dims, output_dims,):
 3
       def
            super().__init__()
 1
            self.rec layer = nn.GRU(input dims, hidden dims, batch first=True)
 5
            self.fc = nn.Linear(hidden_dims, output_dims)
 6
 7
8
       def forward(self, x):
9
            x = self.reshape(x)
10
11
            outputs, hidden_states = self.rec_layer(x)
            outputs = outputs[:, -1, :]
12
13
            outputs = outputs.squeeze(1)
           outputs = self.fc(outputs)
14
15
           return outputs.squeeze(1)
16
17
18
       def reshape(self, x):
            return torch.reshape(x, shape=(x.shape[0], x.shape[2], x.shape[1]))
19
```

Figure 6.3: Gated Recurring Unit (GRU)

```
1
     class FeedForward(nn.Module):
 2
            def __init__(self, input_dims, time_step):
    super().__init__()
 з
 4
 5
                   # 5 Layers
 6
                   self.flatten = nn.Flatten()
                   self.fc1 = nn.Linear(input_dims * time_step, 512)
self.fc2 = nn.Linear(512, 256)
self.fc3 = nn.Linear(256, 256)
self.fc4 = nn.Linear(256, 128)
self.fc5 = nn.Linear(128, 1)
 8
 9
10
11
12
13
14
                   self.activation = nn.ReLU()
15
16
            def forward(self, x):
17
                   x = self.flatten(x)
18
19
                   x = self.activation(self.fc1(x))
                   x = self.activation(self.fc2(x))
x = self.activation(self.fc3(x))
x = self.activation(self.fc4(x))
20
21
22
23
                   x = self.fc5(x)
24
25
                   return
```

Figure 6.4: Feed Forward Model creation

Three models were created 1 layer LSTM, GRU and Feed Forward. They were obtained form pytorch in the neural network. All models are recurrent models only LSTM and GRU were created a with the required dimensions for the models. The last hidden layer is grabbed and getting a reshaped form of hidden and output batch. The feed forward model was created with the linear functions consisting 4 layers and a dimension with inputs along with the time step. The observations are flattened.

```
import torch.backends.cudnn as cudnn
1
  torch.cuda.empty_cache()
2
  cudnn.benchmark = True
З
4
5 device = torch.device("cuda:0" if torch.cuda.is available() else "cpu")
1
  lstm_model = LSTM(input_dims=len(ECON_cols), hidden_dims=32, output_dims=1).to(device)
2 lstm_optimizer = optim.Adam(
З
       lstm_model.parameters()
4
      lr=0.001
  )
5
6 lstm loss fn = torch.nn.MSELoss()
  train_dl = train_dl
1
2 train_model = lstm_model
З
  train_optimizer = lstm_optimizer
4 train_loss_fn = lstm_loss_fn
1
1
  gru_model = GRU(input_dims=len(ECON_cols), hidden_dims=32, output_dims=1).to(device)
  gru_optimizer = optim.Adam(
2
З
       gru_model.parameters(),
4
       lr=0.001
5)
  gru_loss_fn = torch.nn.MSELoss()
6
1
  train_dl = train_dl
2
  train_model = gru_model
```

```
3 train_optimizer = gru_optimizer
```

```
4 train_loss_fn = gru_loss_fn
```

The cudnn benchmark is set true for hardware validation. CPU has been set for carrying out the models to run the dimensions.

The LSTM and GRU models are trained and optimized with calculating the loss using the loss function. The loss function will calculate the MSE loss.

```
1 train_np_x = np.stack([obs for obs, label in train_ds], axis=0)
2 train_np_y = np.stack([label for obs, label in train_ds])
3 val_np_x = np.stack([obs for obs, label in val_ds], axis=0)
4 val_np_y = np.stack([label for obs, label in val_ds])
5
6 train_np_x = train_np_x.reshape((train_np_x.shape[0], -1))
7 val_np_x = val_np_x.reshape((val_np_x.shape[0], -1))
1 xgb_model = XGBRegressor()
```

2 xgb\_model.fit(train\_np\_x, train\_np\_y)

Figure 6.5: Random Forest with Gradient Boosting

The training and validation dimensions have been converted to stacks using numpy. They are reshaped and fitted in the model. The results forecasted using matplotlib and box plots should give the validation loss for the number of observations.

### References

[1] Adi Bronstein (2017) *A quick introduction to the "Pandas" Python Library* Available at: <u>https://towardsdatascience.com/a-quick-introduction-to-the-pandas-python-library-f1b678f34673 [25</u> October 2021].

[2] Jason Brownlee (2017) *Stacked Long Short-term Memory Networks* Available at: <u>https://machinelearningmastery.com/stacked-long-short-term-memory-networks/</u> [10 November 2021]

[3] Tasnuva Zaman (2019) Building Neural Network Using Pytorch Available at: <u>https://towardsdatascience.com/building-neural-network-using-pytorch-84f6e75f9a</u> [12 November 2021]

[4] GeeksforGeeks (2020) *K-Means Clustering in R programming* Available at: <u>https://www.geeksforgeeks.org/k-means-clustering-in-r-programming/</u> [23 October 2021]