code277241ml

May 17, 2024

[221]:

```
import torch
import random
import pandas as pd
import numpy as np
from torch import nn
from torch import optim
from torch.utils.data import Dataset
from torch.utils.data import DataLoader
from sklearn.neural_network import MLPRegressor
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler,MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import f1_score, accuracy_score, confusion_matrix,␣
↪ConfusionMatrixDisplay
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.decomposition import PCA
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import KFold
```

reading the data

[222]:

```
consumer = pd.read_csv('/content/Consumer prices indicators -␣
↪FAOSTAT_data_en_2-22-2024.csv')#used
crop_prod = pd.read_csv('/content/Crops production indicators -␣
↪FAOSTAT_data_en_2-22-2024.csv')#used
emissions = pd.read_csv('/content/Emissions - FAOSTAT_data_en_2-27-2024.
↪csv')#file4 used
employment = pd.read_csv('/content/Employment - FAOSTAT_data_en_2-27-2024.
↪csv')#used
exchange = pd.read_csv('/content/Exchange rate - FAOSTAT_data_en_2-22-2024.
↪csv')#used
fertilizers = pd.read_csv('/content/Fertilizers use - FAOSTAT_data_en_2-27-2024.
↪csv')#used
food_balance = pd.read_csv('/content/Food balances indicators -␣
↪FAOSTAT_data_en_2-22-2024.csv')#file5 used
```

```
food_security = pd.read_csv('/content/Food security indicators  -␣
 ↪FAOSTAT_data_en_2-22-2024.csv')# used
food_trade = pd.read_csv('/content/Food trade indicators -␣
 ↪FAOSTAT_data_en_2-22-2024.csv')#file1 used
foreign_invest = pd.read_csv('/content/Foreign direct investment -␣
 ↪FAOSTAT_data_en_2-27-2024.csv')# used
land_temp = pd.read_csv('/content/Land temperature change -␣
 ↪FAOSTAT_data_en_2-27-2024.csv')#used
land_use =pd.read_csv('/content/Land use - FAOSTAT_data_en_2-22-2024.
 ↪csv',low_memory=False)#used
pesticides = pd.read_csv('/content/Pesticides use - FAOSTAT_data_en_2-27-2024.
 ↪csv')# used
```

food_trade

[223]:
```
food_trade.head(5)
```

[223]:
```
  Domain Code                        Domain  Area Code (M49)         Area  \
0         TCL  Crops and livestock products                4  Afghanistan
1         TCL  Crops and livestock products                4  Afghanistan
2         TCL  Crops and livestock products                4  Afghanistan
3         TCL  Crops and livestock products                4  Afghanistan
4         TCL  Crops and livestock products                4  Afghanistan

   Element Code       Element Item Code (CPC)                      Item  \
0          5622  Import Value           F1888  Cereals and Preparations
1          5622  Import Value           F1888  Cereals and Preparations
2          5622  Import Value           F1888  Cereals and Preparations
3          5622  Import Value           F1888  Cereals and Preparations
4          5622  Import Value           F1888  Cereals and Preparations

   Year Code  Year      Unit    Value Flag Flag Description  Note
0       1991  1991  1000 USD  41600.0    A  Official figure   NaN
1       1992  1992  1000 USD  25600.0    E  Estimated value   NaN
2       1993  1993  1000 USD  40000.0    E  Estimated value   NaN
3       1994  1994  1000 USD  25700.0    E  Estimated value   NaN
4       1995  1995  1000 USD  37720.0    E  Estimated value   NaN
```

[224]:
```
itemcodes =␣
 ↪["F1844","F1846","F1847","F1848","F1888","F1889","F1890","F1896"]#filtering␣
 ↪the food_trade
food_trade = food_trade[food_trade["Item Code (CPC)"].isin(itemcodes)]
```

[225]:
```
export_frame = food_trade[food_trade["Element"] == "Export Value"]#creating␣
 ↪export and import dataframes
export_frame = export_frame.groupby(["Area","Year"], as_index=False).Value.sum()
export_frame.rename(columns={'Value': 'TotalExportValue'},inplace = True)
```

```
import_frame = food_trade[food_trade["Element"] == "Import Value"]
import_frame = import_frame.groupby(["Area","Year"], as_index=False).Value.sum()
import_frame.rename(columns={'Value': 'TotalImportValue'},inplace = True)
frame_merged = pd.merge(export_frame, import_frame, on=['Area',␣
↪'Year'],how='outer')#merging the export and import dfs
display(frame_merged.head())
nan_val = frame_merged[frame_merged.isna().any(axis=1)]#nan values
```

```
          Area  Year  TotalExportValue  TotalImportValue
0  Afghanistan  1991           98243.0          125520.0
1  Afghanistan  1992           42112.0          128605.0
2  Afghanistan  1993           44564.0          132076.0
3  Afghanistan  1994           50357.0          112377.0
4  Afghanistan  1995           49596.0          213741.0
```

consumer

[226]:
```
consumer.head(5)
```

[226]:
```
  Domain Code                  Domain  Area Code (M49)         Area  \
0          CP  Consumer Price Indices                4  Afghanistan
1          CP  Consumer Price Indices                4  Afghanistan
2          CP  Consumer Price Indices                4  Afghanistan
3          CP  Consumer Price Indices                4  Afghanistan
4          CP  Consumer Price Indices                4  Afghanistan

   Year Code  Year  Item Code                                     Item  \
0       2000  2000      23013  Consumer Prices, Food Indices (2015 = 100)
1       2000  2000      23013  Consumer Prices, Food Indices (2015 = 100)
2       2000  2000      23013  Consumer Prices, Food Indices (2015 = 100)
3       2000  2000      23013  Consumer Prices, Food Indices (2015 = 100)
4       2000  2000      23013  Consumer Prices, Food Indices (2015 = 100)

   Months Code    Months  Element Code Element Unit      Value Flag  \
0         7001   January          6125   Value  NaN  24.356332    I
1         7002  February          6125   Value  NaN  23.636242    I
2         7003     March          6125   Value  NaN  23.485345    I
3         7004     April          6125   Value  NaN  24.767194    I
4         7005       May          6125   Value  NaN  25.956912    I

  Flag Description                Note
0    Imputed value  base year is 2015
1    Imputed value  base year is 2015
2    Imputed value  base year is 2015
3    Imputed value  base year is 2015
4    Imputed value  base year is 2015
```

[227]:
```python
food_indices = consumer[consumer["Item"] == "Consumer Prices, Food Indices␣
↪(2015 = 100)"]#filtering the consumer df
food_indices = food_indices.groupby(["Area","Year"], as_index=False).Value.
↪mean()#creating food_indices and food_inflation dfs
food_indices.rename(columns={'Value': 'Food Indices'},inplace = True)
food_inflation = consumer[consumer["Item"] == "Food price inflation"]
food_inflation = food_inflation.groupby(["Area","Year"], as_index=False).Value.
↪mean()
food_inflation.rename(columns={'Value': 'Food Inflation'},inplace = True)
frame_merged2 = pd.merge(food_inflation, food_indices, on=['Area',␣
↪'Year'],how='outer')#merging them on outer join
display(frame_merged2.head())
nan_val = frame_merged2[frame_merged2.isna().any(axis=1)]#nan values
```

	Area	Year	Food Inflation	Food Indices
0	Afghanistan	2001	12.780692	29.893548
1	Afghanistan	2002	18.254516	35.344892
2	Afghanistan	2003	14.102244	40.203113
3	Afghanistan	2004	14.072172	45.840561
4	Afghanistan	2005	12.606240	51.605262

crop production dataframe

[228]:
```python
crop_prod.head(5)
```

[228]:

	Domain Code	Domain	Area Code (M49)	Area	\
0	QCL	Crops and livestock products	4	Afghanistan	
1	QCL	Crops and livestock products	4	Afghanistan	
2	QCL	Crops and livestock products	4	Afghanistan	
3	QCL	Crops and livestock products	4	Afghanistan	
4	QCL	Crops and livestock products	4	Afghanistan	

	Element Code	Element	Item Code (CPC)	Item	Year Code	Year	\
0	5419	Yield	F1717	Cereals, primary	2000	2000	
1	5419	Yield	F1717	Cereals, primary	2001	2001	
2	5419	Yield	F1717	Cereals, primary	2002	2002	
3	5419	Yield	F1717	Cereals, primary	2003	2003	
4	5419	Yield	F1717	Cereals, primary	2004	2004	

	Unit	Value	Flag	Flag Description	Note
0	100 g/ha	8063	A	Official figure	NaN
1	100 g/ha	10067	A	Official figure	NaN
2	100 g/ha	16698	A	Official figure	NaN
3	100 g/ha	14580	A	Official figure	NaN
4	100 g/ha	13348	A	Official figure	NaN

[229]:
```
crop_prod = crop_prod.groupby(["Area","Year"], as_index=False).Value.
 ↪sum()#grouping through area and year columns
crop_prod.rename(columns={'Value': 'Yield'},inplace = True)# renaming the␣
 ↪column value to yield
crop_prod.head()
```

[229]:
```
          Area  Year   Yield
0  Afghanistan  2000  661957
1  Afghanistan  2001  667714
2  Afghanistan  2002  672489
3  Afghanistan  2003  673301
4  Afghanistan  2004  675944
```

emissions dataframe

[230]:
```
emissions.head(5)
```

[230]:
```
  Domain Code                Domain  Area Code (M49)         Area  \
0         GCE  Emissions from Crops                4  Afghanistan
1         GCE  Emissions from Crops                4  Afghanistan
2         GCE  Emissions from Crops                4  Afghanistan
3         GCE  Emissions from Crops                4  Afghanistan
4         GCE  Emissions from Crops                4  Afghanistan

   Element Code                      Element Item Code (CPC)       Item  \
0         72430  Crops total (Emissions N2O)           F1712  All Crops
1         72440  Crops total (Emissions CH4)           F1712  All Crops
2         72430  Crops total (Emissions N2O)           F1712  All Crops
3         72440  Crops total (Emissions CH4)           F1712  All Crops
4         72430  Crops total (Emissions N2O)           F1712  All Crops

   Year Code  Year  Source Code      Source Unit    Value Flag  \
0       2000  2000         3050  FAO TIER 1   kt   0.7056    E
1       2000  2000         3050  FAO TIER 1   kt  20.8471    E
2       2001  2001         3050  FAO TIER 1   kt   0.7054    E
3       2001  2001         3050  FAO TIER 1   kt  19.2605    E
4       2002  2002         3050  FAO TIER 1   kt   1.0656    E

  Flag Description  Note
0  Estimated value   NaN
1  Estimated value   NaN
2  Estimated value   NaN
3  Estimated value   NaN
4  Estimated value   NaN
```

[231]:
```
N2O_data = emissions[emissions["Element"] == "Emissions (N2O)"]#creating␣
 ↪n2O_data df
```

```python
Co2_data = emissions[emissions["Element"] == "Emissions (CO2)"]#creating co2_data df
N20_data = N20_data.groupby(["Area","Year"], as_index=False).Value.sum()
N20_data.rename(columns={'Value': 'Emission N20'},inplace = True)
Co2_data = Co2_data.groupby(["Area","Year"], as_index=False).Value.sum()
Co2_data.rename(columns={'Value': 'Emmision Co2'},inplace = True)
frame_merged4 = pd.merge(N20_data, Co2_data, on=['Area', 'Year'],how='outer')#merging them
frame_merged4.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5130 entries, 0 to 5129
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Area          5130 non-null   object
 1   Year          5130 non-null   int64
 2   Emission N20  5130 non-null   float64
 3   Emmision Co2  5130 non-null   float64
dtypes: float64(2), int64(1), object(1)
memory usage: 160.4+ KB
```

food balance dataframe

[232]:
```python
food_balance.head(5)
```

[232]:
```
  Domain Code                Domain  Area Code (M49)         Area  \
0         FBS  Food Balances (2010-)                4  Afghanistan
1         FBS  Food Balances (2010-)                4  Afghanistan
2         FBS  Food Balances (2010-)                4  Afghanistan
3         FBS  Food Balances (2010-)                4  Afghanistan
4         FBS  Food Balances (2010-)                4  Afghanistan

   Element Code          Element Item Code (FBS)                       Item  \
0          5611  Import Quantity           S2905  Cereals - Excluding Beer
1          5611  Import Quantity           S2905  Cereals - Excluding Beer
2          5611  Import Quantity           S2905  Cereals - Excluding Beer
3          5611  Import Quantity           S2905  Cereals - Excluding Beer
4          5611  Import Quantity           S2905  Cereals - Excluding Beer

   Year Code  Year    Unit   Value Flag Flag Description
0       2010  2010  1000 t  2000.0    E  Estimated value
1       2011  2011  1000 t  2448.0    E  Estimated value
2       2012  2012  1000 t  2001.0    E  Estimated value
3       2013  2013  1000 t  2155.0    E  Estimated value
4       2014  2014  1000 t  1840.0    E  Estimated value
```

[233]:
```
itemcodes =␣
↪["S2905","S2907","S2908","S2909","S2911","S2912","S2913","S2914","S2918","S2919","S2923"]
food_balance = food_balance[food_balance["Item Code (FBS)"].
↪isin(itemcodes)]#filtering the food_balance df
import_frame = food_balance[food_balance["Element Code"] == 5611]#creating␣
↪import and export dfs
export_frame = food_balance[food_balance["Element Code"] == 5911]
consumer_data = food_balance[food_balance["Element Code"] == 5142]#creating␣
↪consumer,loss and others uses dfs
loss_data = food_balance[food_balance["Element Code"] == 5123]
others_data = food_balance[food_balance["Element Code"] == 5154]
import_frame = import_frame.groupby(["Area","Year"], as_index=False).Value.sum()
import_frame.rename(columns={'Value': 'Import Value'},inplace = True)
export_frame = export_frame.groupby(["Area","Year"], as_index=False).Value.sum()
export_frame.rename(columns={'Value': 'Export Value'},inplace = True)
consumer_data = consumer_data.groupby(["Area","Year"], as_index=False).Value.
↪sum()
consumer_data.rename(columns={'Value': 'Whole Food Consumption'},inplace = True)
loss_data = loss_data.groupby(["Area","Year"], as_index=False).Value.sum()
loss_data.rename(columns={'Value': 'Total Food Loss'},inplace = True)
others_data = others_data.groupby(["Area","Year"], as_index=False).Value.sum()
others_data.rename(columns={'Value': 'Others'},inplace = True)
```

[234]:
```
frame_merged5 = pd.merge(import_frame, export_frame, on=['Area', 'Year'],␣
↪how='outer')
frame_merged5 = pd.merge(frame_merged5, consumer_data, on=['Area', 'Year'],␣
↪how='outer')
frame_merged5 = pd.merge(frame_merged5, loss_data, on=['Area', 'Year'],␣
↪how='outer')
frame_merged5 = pd.merge(frame_merged5, others_data, on=['Area', 'Year'],␣
↪how='outer')#merging all the 4 above dfs
frame_merged5.head()
```

[234]:
```
          Area  Year  Import Value  Export Value  Whole Food Consumption  \
0  Afghanistan  2010        2897.0         360.0                  8506.0
1  Afghanistan  2011        3505.0         277.0                  8624.0
2  Afghanistan  2012        3325.0         198.0                  9049.0
3  Afghanistan  2013        3507.0         281.0                  9554.0
4  Afghanistan  2014        3654.0         412.0                 10481.0

   Total Food Loss  Others
0           1090.0   215.0
1            866.0   257.0
2           1137.0   442.0
3           1168.0   428.0
4           1255.0    27.0
```

employment dataframe

[235]:
```
employment.head(5)
```

[235]:
```
  Domain Code                            Domain  Area Code (M49)  \
0         OEA  Employment Indicators: Agriculture                4
1         OEA  Employment Indicators: Agriculture                4
2         OEA  Employment Indicators: Agriculture                4
3         OEA  Employment Indicators: Agriculture                4
4         OEA  Employment Indicators: Agriculture                4

          Area  Indicator Code  \
0  Afghanistan           21150
1  Afghanistan           21150
2  Afghanistan           21144
3  Afghanistan           21144
4  Afghanistan           21144

                                           Indicator  Sex Code    Sex  \
0  Mean weekly hours actually worked per employed…         1  Total
1  Mean weekly hours actually worked per employed…         1  Total
2  Employment in agriculture, forestry and fishin…         1  Total
3  Employment in agriculture, forestry and fishin…         1  Total
4  Employment in agriculture, forestry and fishin…         1  Total

   Year Code  Year  Element Code Element  Source Code  \
0       2014  2014          6173   Value         3021
1       2017  2017          6173   Value         3021
2       2000  2000          6199   Value         3043
3       2001  2001          6199   Value         3043
4       2002  2002          6199   Value         3043

                                Source     Unit    Value Flag  \
0  Household income and expenditure survey       No    31.68    X
1  Household income and expenditure survey       No    29.66    X
2              ILO - ILO Modelled Estimates  1000 No  2765.95    X
3              ILO - ILO Modelled Estimates  1000 No  2805.54    X
4              ILO - ILO Modelled Estimates  1000 No  2897.51    X

                       Flag Description  \
0  Figure from international organizations
1  Figure from international organizations
2  Figure from international organizations
3  Figure from international organizations
4  Figure from international organizations

                                                Note
```

```
0  Job coverage: Main job currently held Reposito…
1  Job coverage: Main job currently held Reposito…
2                                                NaN
3                                                NaN
4                                                NaN
```

[236]:
```
model_est_data = employment[employment["Element Code"] == 6199]#creating df by␣
 ↪using the model estimated data
mean_work_data = employment[employment["Element Code"] == 6173]#creating df by␣
 ↪using average weekly hour work
model_est_data= model_est_data.groupby(["Area","Year"], as_index=False).Value.
 ↪sum()
model_est_data.rename(columns={'Value': 'Estimated Work'},inplace = True)
mean_work_data = mean_work_data.groupby(["Area","Year"], as_index=False).Value.
 ↪sum()
mean_work_data.rename(columns={'Value': 'Mean Hourly Work'},inplace = True)
frame_merged6 = pd.merge(model_est_data, mean_work_data, on=['Area',␣
 ↪'Year'],how='outer')#merging them
frame_merged6.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4217 entries, 0 to 4216
Data columns (total 4 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   Area              4217 non-null   object
 1   Year              4217 non-null   int64
 2   Estimated Work    4178 non-null   float64
 3   Mean Hourly Work  1735 non-null   float64
dtypes: float64(2), int64(1), object(1)
memory usage: 131.9+ KB
```

exchange dataframe

[237]:
```
exchange.head(5)
```

[237]:
```
  Domain Code           Domain  Area Code (M49)         Area  \
0          PE  Exchange rates                4  Afghanistan
1          PE  Exchange rates                4  Afghanistan
2          PE  Exchange rates                4  Afghanistan
3          PE  Exchange rates                4  Afghanistan
4          PE  Exchange rates                4  Afghanistan

  ISO Currency Code (FAO) Currency Element Code                       Element  \
0                     AFA  Afghani          LCU  Local currency units per USD
1                     AFA  Afghani          LCU  Local currency units per USD
2                     AFA  Afghani          LCU  Local currency units per USD
```

```
3                    AFA  Afghani          LCU  Local currency units per USD
4                    AFA  Afghani          LCU  Local currency units per USD

   Year Code  Year  Months Code    Months  Unit      Value Flag  \
0       1980  1980         7001   January   NaN  44.129167    X
1       1980  1980         7002  February   NaN  44.129167    X
2       1980  1980         7003     March   NaN  44.129167    X
3       1980  1980         7004     April   NaN  44.129167    X
4       1980  1980         7005       May   NaN  44.129167    X

                        Flag Description
0  Figure from international organizations
1  Figure from international organizations
2  Figure from international organizations
3  Figure from international organizations
4  Figure from international organizations
```

[238]:
```
exchange = exchange.groupby(["Area","Year"], as_index=False).Value.
 ↪mean()#grouping using area and year cols
exchange.rename(columns={'Value': 'RateOfExchange'},inplace = True)# value to␣
 ↪RateOfExchange
exchange.head()
```

[238]:
```
          Area  Year  RateOfExchange
0  Afghanistan  1980       44.129167
1  Afghanistan  1981       49.479902
2  Afghanistan  1982       50.599608
3  Afghanistan  1983       50.599608
4  Afghanistan  1984       50.599606
```

fertilizers dataframe

[239]:
```
fertilizers
```

[239]:
```
      Domain Code                  Domain  Area Code (M49)         Area  \
0             RFB  Fertilizers by Product                4  Afghanistan
1             RFB  Fertilizers by Product                4  Afghanistan
2             RFB  Fertilizers by Product                4  Afghanistan
3             RFB  Fertilizers by Product                4  Afghanistan
4             RFB  Fertilizers by Product                4  Afghanistan
…               …                       …              …            …
17802         RFB  Fertilizers by Product              716     Zimbabwe
17803         RFB  Fertilizers by Product              716     Zimbabwe
17804         RFB  Fertilizers by Product              716     Zimbabwe
17805         RFB  Fertilizers by Product              716     Zimbabwe
17806         RFB  Fertilizers by Product              716     Zimbabwe
```

```
       Element Code           Element  Item Code  \
0              5157  Agricultural Use       4021
1              5157  Agricultural Use       4021
2              5157  Agricultural Use       4021
3              5157  Agricultural Use       4001
4              5157  Agricultural Use       4001
…                 …                 …         …
17802          5157  Agricultural Use       4006
17803          5157  Agricultural Use       4006
17804          5157  Agricultural Use       4006
17805          5157  Agricultural Use       4006
17806          5157  Agricultural Use       4006

                                        Item  Year Code  Year Unit  \
0                             NPK fertilizers       2002  2002    t
1                             NPK fertilizers       2003  2003    t
2                             NPK fertilizers       2004  2004    t
3                                        Urea       2004  2004    t
4                                        Urea       2005  2005    t
…                                           …         …     …    …
17802  Urea and ammonium nitrate solutions (UAN)       2004  2004    t
17803  Urea and ammonium nitrate solutions (UAN)       2008  2008    t
17804  Urea and ammonium nitrate solutions (UAN)       2009  2009    t
17805  Urea and ammonium nitrate solutions (UAN)       2010  2010    t
17806  Urea and ammonium nitrate solutions (UAN)       2011  2011    t

          Value Flag Flag Description
0      17900.00    I    Imputed value
1      33200.00    I    Imputed value
2      47700.00    I    Imputed value
3      42300.00    I    Imputed value
4      20577.00    I    Imputed value
…           …   …                 …
17802      5.00    I    Imputed value
17803      2.13    I    Imputed value
17804      9.00    I    Imputed value
17805   4971.00    I    Imputed value
17806      7.00    I    Imputed value

[17807 rows x 14 columns]
```

[240]:

```
fertilizers = fertilizers.groupby(["Area","Year"], as_index=False).Value.sum()
fertilizers.rename(columns={'Value': 'Fertilizers Quantity'},inplace =
 ↪True)#value to Fertilizers Quantity
fertilizers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 1933 entries, 0 to 1932
Data columns (total 3 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   Area                  1933 non-null   object
 1   Year                  1933 non-null   int64
 2   Fertilizers Quantity  1933 non-null   float64
dtypes: float64(1), int64(1), object(1)
memory usage: 45.4+ KB
```

land usage dataframe

[241]:
```
land_use
```

[241]:
```
      Domain Code    Domain  Area Code (M49)        Area  Element Code  \
0              RL  Land Use                4  Afghanistan          5110
1              RL  Land Use                4  Afghanistan          5110
2              RL  Land Use                4  Afghanistan          5110
3              RL  Land Use                4  Afghanistan          5110
4              RL  Land Use                4  Afghanistan          5110
…            …         …                …           …          …
97990          RL  Land Use              716     Zimbabwe          5110
97991          RL  Land Use              716     Zimbabwe          5110
97992          RL  Land Use              716     Zimbabwe          5110
97993          RL  Land Use              716     Zimbabwe          5110
97994          RL  Land Use              716     Zimbabwe          5110

      Element  Item Code                              Item  Year Code  Year  \
0        Area       6600                      Country area       1980  1980
1        Area       6600                      Country area       1981  1981
2        Area       6600                      Country area       1982  1982
3        Area       6600                      Country area       1983  1983
4        Area       6600                      Country area       1984  1984
…         …        …                               …        …   …
97990    Area       6690  Land area equipped for irrigation       2017  2017
97991    Area       6690  Land area equipped for irrigation       2018  2018
97992    Area       6690  Land area equipped for irrigation       2019  2019
97993    Area       6690  Land area equipped for irrigation       2020  2020
97994    Area       6690  Land area equipped for irrigation       2021  2021

          Unit    Value Flag                      Flag Description Note
0      1000 ha  65286.0    A                       Official figure  NaN
1      1000 ha  65286.0    A                       Official figure  NaN
2      1000 ha  65286.0    A                       Official figure  NaN
3      1000 ha  65286.0    A                       Official figure  NaN
4      1000 ha  65286.0    A                       Official figure  NaN
…         …       …  …                                   …   …
97990  1000 ha    186.6    X  Figure from international organizations  NaN
```

```
97991  1000 ha    186.6    I                            Imputed value  NaN
97992  1000 ha    186.6    I                            Imputed value  NaN
97993  1000 ha    186.6    I                            Imputed value  NaN
97994  1000 ha    186.6    I                            Imputed value  NaN

[97995 rows x 15 columns]
```

[242]:

```
land_use = land_use.groupby(["Area","Year"], as_index=False).Value.sum()
land_use.rename(columns={'Value': 'Land Used'},inplace = True)#value to Land␣
 ↪Used
land_use.head()
```

[242]:

```
          Area  Year  Land Used
0  Afghanistan  1980   255210.0
1  Afghanistan  1981   255241.0
2  Afghanistan  1982   255260.0
3  Afghanistan  1983   255275.0
4  Afghanistan  1984   255306.0
```

land temperature dataframe

[243]:

```
land_temp.head(5)
```

[243]:

```
  Domain Code                      Domain  Area Code (M49)         Area  \
0          ET  Temperature change on land                4  Afghanistan
1          ET  Temperature change on land                4  Afghanistan
2          ET  Temperature change on land                4  Afghanistan
3          ET  Temperature change on land                4  Afghanistan
4          ET  Temperature change on land                4  Afghanistan

   Element Code             Element  Months Code       Months  Year Code  \
0          7271  Temperature change         7016  Dec-Jan-Feb       2000
1          7271  Temperature change         7016  Dec-Jan-Feb       2001
2          7271  Temperature change         7016  Dec-Jan-Feb       2002
3          7271  Temperature change         7016  Dec-Jan-Feb       2003
4          7271  Temperature change         7016  Dec-Jan-Feb       2004

   Year Unit  Value Flag Flag Description
0  2000   °c  0.618    E  Estimated value
1  2001   °c  0.365    E  Estimated value
2  2002   °c  1.655    E  Estimated value
3  2003   °c  0.997    E  Estimated value
4  2004   °c  1.883    E  Estimated value
```

[244]:

```
land_temp = land_temp[land_temp["Element Code"] == 7271]#filtering using␣
 ↪element code,month code
land_temp= land_temp[land_temp["Months Code"] == 7020]
```

```
land_temp= land_temp.groupby(["Area","Year"], as_index=False).Value.mean()
land_temp.rename(columns={'Value': 'Temperature_change'},inplace = True)#value␣
 ↪to Temperature_change
land_temp.head()
```

[244]:

```
          Area  Year  Temperature_change
0  Afghanistan  2000               0.993
1  Afghanistan  2001               1.311
2  Afghanistan  2002               1.365
3  Afghanistan  2003               0.587
4  Afghanistan  2004               1.373
```

pesticides dataframe

[245]:
```
pesticides.head(5)
```

[245]:

```
  Domain Code          Domain  Area Code (M49)     Area  Element Code  \
0          RP  Pesticides Use                8  Albania          5157
1          RP  Pesticides Use                8  Albania          5159
2          RP  Pesticides Use                8  Albania          5173
3          RP  Pesticides Use                8  Albania          5157
4          RP  Pesticides Use                8  Albania          5159

                                   Element  Item Code               Item  \
0                         Agricultural Use       1357  Pesticides (total)
1                 Use per area of cropland       1357  Pesticides (total)
2  Use per value of agricultural production       1357  Pesticides (total)
3                         Agricultural Use       1357  Pesticides (total)
4                 Use per area of cropland       1357  Pesticides (total)

   Year Code  Year    Unit   Value Flag Flag Description Note
0       2000  2000       t  307.98    E  Estimated value  NaN
1       2000  2000   kg/ha    0.44    E  Estimated value  NaN
2       2000  2000  g/Int$    0.23    E  Estimated value  NaN
3       2001  2001       t  319.38    E  Estimated value  NaN
4       2001  2001   kg/ha    0.46    E  Estimated value  NaN
```

[246]:
```
pesticides = pesticides.groupby(["Area","Year"], as_index=False).Value.sum()
pesticides.rename(columns={'Value': 'Pesticides Quantity'},inplace =␣
 ↪True)#value to Pesticides Quantity
pesticides.head()
```

[246]:

```
      Area  Year  Pesticides Quantity
0  Albania  2000               608.57
1  Albania  2001               629.78
2  Albania  2002               650.98
3  Albania  2003               672.17
```

```
4  Albania  2004              693.36
```

foreign investiment dataframe

[247]:
```python
foreign_invest.head(5)
```

[247]:
```
  Domain Code                           Domain  Area Code (M49)         Area  \
0         FDI  Foreign Direct Investment (FDI)                4  Afghanistan
1         FDI  Foreign Direct Investment (FDI)                4  Afghanistan
2         FDI  Foreign Direct Investment (FDI)                4  Afghanistan
3         FDI  Foreign Direct Investment (FDI)                4  Afghanistan
4         FDI  Foreign Direct Investment (FDI)                4  Afghanistan

   Element Code    Element  Item Code               Item  Year Code  Year  \
0          6110  Value US$      23082  Total FDI inflows       2000  2000
1          6110  Value US$      23082  Total FDI inflows       2001  2001
2          6110  Value US$      23082  Total FDI inflows       2002  2002
3          6110  Value US$      23082  Total FDI inflows       2003  2003
4          6110  Value US$      23082  Total FDI inflows       2004  2004

          Unit   Value Flag                          Flag Description    Note
0  million USD    0.17    X  Figure from international organizations  UNCTAD
1  million USD    0.68    X  Figure from international organizations  UNCTAD
2  million USD   50.00    X  Figure from international organizations  UNCTAD
3  million USD   57.80    X  Figure from international organizations  UNCTAD
4  million USD  186.90    X  Figure from international organizations  UNCTAD
```

[248]:
```python
foreign_invest = foreign_invest.groupby(["Area","Year"], as_index=False).Value.
 ↪sum()
foreign_invest.rename(columns={'Value': 'Foreign Investment'},inplace =␣
 ↪True)#value to Foreign Investment
foreign_invest.head()
```

[248]:
```
          Area  Year  Foreign Investment
0  Afghanistan  2000                0.17
1  Afghanistan  2001                0.68
2  Afghanistan  2002               50.00
3  Afghanistan  2003               58.80
4  Afghanistan  2004              186.20
```

food security dataframe

[249]:
```python
food_security.head(5)
```

[249]:
```
  Domain Code                               Domain  Area Code (M49)  \
0          FS  Suite of Food Security Indicators                  4
1          FS  Suite of Food Security Indicators                  4
2          FS  Suite of Food Security Indicators                  4
```

```
3          FS   Suite of Food Security Indicators                4
4          FS   Suite of Food Security Indicators                4

          Area  Element Code Element  Item Code  \
0  Afghanistan          6121   Value      21010
1  Afghanistan          6121   Value      21010
2  Afghanistan          6121   Value      21010
3  Afghanistan          6121   Value      21010
4  Afghanistan          6121   Value      21010

                                                Item  Year Code       Year  \
0  Average dietary energy supply adequacy (percen…   20002002  2000-2002
1  Average dietary energy supply adequacy (percen…   20012003  2001-2003
2  Average dietary energy supply adequacy (percen…   20022004  2002-2004
3  Average dietary energy supply adequacy (percen…   20032005  2003-2005
4  Average dietary energy supply adequacy (percen…   20042006  2004-2006

  Unit  Value Flag Flag Description Note
0    %   88.0    E  Estimated value  NaN
1    %   89.0    E  Estimated value  NaN
2    %   92.0    E  Estimated value  NaN
3    %   93.0    E  Estimated value  NaN
4    %   94.0    E  Estimated value  NaN
```

[250]:
```
food_security = food_security.groupby(["Area","Year"], as_index=False).Value.
↪sum()
food_security.rename(columns={'Value': 'Dietary percentage'},inplace =␣
↪True)#value to Dietary percentage
food_security.head()
```

[250]:
```
          Area       Year  Dietary percentage
0  Afghanistan       2000               91.26
1  Afghanistan  2000-2002              418.40
2  Afghanistan       2001               98.90
3  Afghanistan  2001-2003              456.30
4  Afghanistan       2002              125.36
```

[251]:
```
final_data = pd.merge(frame_merged, frame_merged2, on=['Area',␣
↪'Year'],how='outer')
final_data = pd.merge(final_data, crop_prod, on=['Area', 'Year'],how='outer')
final_data = pd.merge(final_data, frame_merged4, on=['Area',␣
↪'Year'],how='outer')
final_data = pd.merge(final_data, frame_merged5, on=['Area',␣
↪'Year'],how='outer')
final_data = pd.merge(final_data, frame_merged6, on=['Area',␣
↪'Year'],how='outer')
final_data = pd.merge(final_data, exchange, on=['Area', 'Year'],how='outer')
```

```
final_data = pd.merge(final_data, fertilizers, on=['Area', 'Year'],how='outer')
final_data = pd.merge(final_data, land_use, on=['Area', 'Year'],how='outer')
final_data = pd.merge(final_data, land_temp, on=['Area', 'Year'],how='outer')
final_data = pd.merge(final_data, pesticides, on=['Area', 'Year'],how='outer')
#final_data = pd.merge(final_data, food_security, on=['Area',␣
↪'Year'],how='outer')
final_data = pd.merge(final_data, foreign_invest, on=['Area',␣
↪'Year'],how='outer')
#merging the all individuals dataframes to one with the name of final_data
final_data.info()
nan_val = final_data[final_data.isna().any(axis=1)]
print(len(nan_val))
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10445 entries, 0 to 10444
Data columns (total 22 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Area                    10445 non-null  object
 1   Year                    10445 non-null  int64
 2   TotalExportValue        6180 non-null   float64
 3   TotalImportValue        6205 non-null   float64
 4   Food Inflation          4653 non-null   float64
 5   Food Indices            4856 non-null   float64
 6   Yield                   4587 non-null   float64
 7   Emission N20            5130 non-null   float64
 8   Emmision Co2            5130 non-null   float64
 9   Import Value            2176 non-null   float64
 10  Export Value            2176 non-null   float64
 11  Whole Food Consumption  2176 non-null   float64
 12  Total Food Loss         2176 non-null   float64
 13  Others                  2149 non-null   float64
 14  Estimated Work          4178 non-null   float64
 15  Mean Hourly Work        1735 non-null   float64
 16  RateOfExchange          8639 non-null   float64
 17  Fertilizers Quantity    1933 non-null   float64
 18  Land Used               9519 non-null   float64
 19  Temperature_change      5219 non-null   float64
 20  Pesticides Quantity     4636 non-null   float64
 21  Foreign Investment      4580 non-null   float64
dtypes: float64(20), int64(1), object(1)
memory usage: 1.8+ MB
9888
```

[252]:
```
final_data.head()
```

```
[252]:          Area  Year  TotalExportValue  TotalImportValue  Food Inflation  \
0  Afghanistan  1991           98243.0          125520.0             NaN
1  Afghanistan  1992           42112.0          128605.0             NaN
2  Afghanistan  1993           44564.0          132076.0             NaN
3  Afghanistan  1994           50357.0          112377.0             NaN
4  Afghanistan  1995           49596.0          213741.0             NaN

   Food Indices  Yield  Emission N20  Emmision Co2  Import Value  …  \
0           NaN    NaN           NaN           NaN           NaN  …
1           NaN    NaN           NaN           NaN           NaN  …
2           NaN    NaN           NaN           NaN           NaN  …
3           NaN    NaN           NaN           NaN           NaN  …
4           NaN    NaN           NaN           NaN           NaN  …

   Total Food Loss  Others  Estimated Work  Mean Hourly Work  RateOfExchange  \
0              NaN     NaN             NaN               NaN       50.599605
1              NaN     NaN             NaN               NaN       50.599605
2              NaN     NaN             NaN               NaN       50.599605
3              NaN     NaN             NaN               NaN      425.099934
4              NaN     NaN             NaN               NaN      833.333333

   Fertilizers Quantity  Land Used  Temperature_change  Pesticides Quantity  \
0                   NaN   255629.0                 NaN                  NaN
1                   NaN   255809.0                 NaN                  NaN
2                   NaN   255425.0                 NaN                  NaN
3                   NaN   254941.0                 NaN                  NaN
4                   NaN   254741.0                 NaN                  NaN

   Foreign Investment
0                 NaN
1                 NaN
2                 NaN
3                 NaN
4                 NaN

[5 rows x 22 columns]
```

Nan VAlues Removal

```
[253]: final_data = final_data.dropna()#dropping the null values
final_data = final_data.reset_index()
final_data.head(20)
```

```
[253]:    index     Area  Year  TotalExportValue  TotalImportValue  \
0     51  Albania  2010          54423.00         716893.00
1     52  Albania  2011          79076.00         756193.00
2     53  Albania  2012          80834.00         725009.00
3     54  Albania  2013         104789.00         745689.00
```

```
4      55    Albania  2014          66876.64         476709.33
5      56    Albania  2015          98324.76         491757.49
6      57    Albania  2016         139745.37         586725.93
7      58    Albania  2017         104215.69         358604.63
8      59    Albania  2018         106920.01         411526.87
9      60    Albania  2019         107556.17         405811.65
10     87    Algeria  2014         358085.30        9634790.70
11    179  Argentina  2010       29481334.00        1445756.00
12    180  Argentina  2011       39144662.00        1763537.00
13    181  Argentina  2012       36905723.00        1764700.00
14    182  Argentina  2013       36026444.00        1741775.00
15    183  Argentina  2014       32138564.44        1615451.71
16    185  Argentina  2016       32312845.56        1900133.25
17    186  Argentina  2017       30242129.34        2519314.03
18    187  Argentina  2018       27279107.88        4195871.87
19    188  Argentina  2019       30928215.18        3022305.59

    Food Inflation  Food Indices      Yield  Emission N20  Emmision Co2  …  \
0         5.186276     84.268140  1130203.0        0.0750      109.7740  …
1         4.392094     87.961900  1194746.0        0.0750      109.8112  …
2         2.404087     90.036610  1238879.0        0.0750      109.8112  …
3         4.214607     93.837014  1181126.0        0.0745      109.1577  …
4         2.212385     95.911724  1245786.0        0.0745      109.1577  …
5         4.294233    100.032060  1331879.0        0.0744      109.0486  …
6         3.254620    103.270159  1457827.0        0.0743      108.8386  …
7         3.924577    107.312936  1510561.0        0.0744      108.8931  …
8         2.690945    110.202018  1659626.0        0.0735      107.5855  …
9         2.911840    113.411033  1629485.0        0.0726      105.7760  …
10        3.907136     95.517483   882319.0        0.0000        0.0000  …
11       14.379745     58.802023  1574014.0        2.1802     5214.6130  …
12        8.730300     63.906953  1643779.0        2.1822     5223.3922  …
13       10.211440     70.440465  1487954.0        2.1817     5222.7532  …
14        7.596194     75.784741  1616288.0        2.1796     5219.5236  …
15       17.616885     89.161945  1496918.0        2.1769     5213.9434  …
16       12.997803    113.063064  1449780.0        2.1830     5222.7425  …
17       17.671433    133.132900  1468335.0        2.1840     5224.2033  …
18       32.162954    176.712373  1479977.0        2.1955     5230.2713  …
19       58.685554    279.846802  1435903.0        2.2150     5231.4007  …

    Total Food Loss  Others  Estimated Work  Mean Hourly Work  RateOfExchange  \
0             216.0   410.0          459.46             36.97      103.936667
1             226.0   396.0          544.53             32.13      100.895833
2             227.0   379.0          527.87             35.27      108.184167
3             226.0   371.0          455.32             36.13      105.669167
4             228.0   158.0          434.23             31.03      105.480000
5             236.0   154.0          447.56             32.19      125.961667
6             249.0   142.0          459.68             34.12      124.142500
```

7	251.0	146.0	453.96	35.47	119.100000
8	249.0	124.0	462.57	36.09	107.989167
9	264.0	138.0	465.19	36.84	109.850833
10	3394.0	2126.0	943.10	45.70	80.579017
11	2747.0	3310.0	1617.47	35.09	3.896295
12	2888.0	3936.0	1598.24	47.31	4.110140
13	2715.0	4277.0	1566.35	44.85	4.536934
14	3034.0	4161.0	1543.55	46.86	5.459353
15	2789.0	5369.0	1523.63	44.17	8.075276
16	3034.0	5525.0	1459.61	45.48	14.758175
17	3201.0	6690.0	1425.14	40.61	16.562707
18	3045.0	6190.0	1426.78	44.44	28.094992
19	3416.0	6226.0	1432.01	43.24	48.147892

	Fertilizers Quantity	Land Used	Temperature_change \
0	114737.00	1.128090e+04	1.191
1	130334.00	1.127660e+04	1.055
2	125008.00	1.127540e+04	1.487
3	116890.28	1.124044e+04	1.333
4	116890.00	1.110494e+04	1.198
5	142691.00	1.112099e+04	1.569
6	168492.00	1.117820e+04	1.464
7	143681.90	1.137900e+04	1.121
8	89558.91	1.138769e+04	2.028
9	128114.19	1.139706e+04	1.675
10	424893.27	6.520377e+05	1.690
11	3281818.00	1.097034e+06	0.135
12	3580053.00	1.093966e+06	0.386
13	2891023.00	1.092656e+06	0.798
14	2992734.00	1.085207e+06	0.442
15	3120461.00	1.078037e+06	0.951
16	3595921.65	1.070188e+06	0.488
17	3727434.00	1.065014e+06	1.095
18	4257068.00	1.059497e+06	0.878
19	4865012.00	1.061106e+06	0.760

	Pesticides Quantity	Foreign Investment
0	1174.43	1056.482726
1	1157.39	906.128407
2	713.09	878.470090
3	890.69	1294.614516
4	905.06	1184.581510
5	1067.36	1025.452236
6	1159.01	1106.794293
7	1218.98	1048.542165
8	872.59	1286.594955
9	1412.04	1327.797915

```
10           9019.40       1488.436201
11         466943.17      12947.760680
12         436458.84      13564.000000
13         428938.41      19705.849578
14         421417.60      15294.972859
15         413896.99      14575.541433
16         396389.95       2229.530935
17         387636.42      12672.491678
18         341610.75      13442.565143
19         403897.51       8171.915990

[20 rows x 23 columns]
```

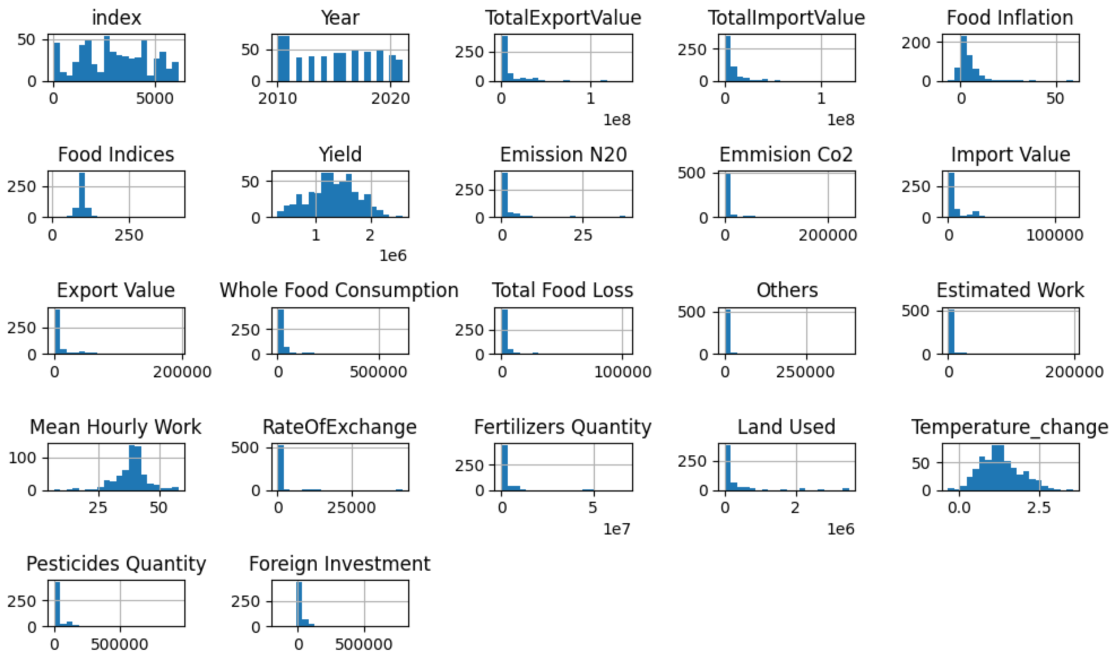
[254]:
```
final_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 557 entries, 0 to 556
Data columns (total 23 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   index                   557 non-null    int64
 1   Area                    557 non-null    object
 2   Year                    557 non-null    int64
 3   TotalExportValue        557 non-null    float64
 4   TotalImportValue        557 non-null    float64
 5   Food Inflation          557 non-null    float64
 6   Food Indices            557 non-null    float64
 7   Yield                   557 non-null    float64
 8   Emission N20            557 non-null    float64
 9   Emmision Co2            557 non-null    float64
 10  Import Value            557 non-null    float64
 11  Export Value            557 non-null    float64
 12  Whole Food Consumption  557 non-null    float64
 13  Total Food Loss         557 non-null    float64
 14  Others                  557 non-null    float64
 15  Estimated Work          557 non-null    float64
 16  Mean Hourly Work        557 non-null    float64
 17  RateOfExchange          557 non-null    float64
 18  Fertilizers Quantity    557 non-null    float64
 19  Land Used               557 non-null    float64
 20  Temperature_change      557 non-null    float64
 21  Pesticides Quantity     557 non-null    float64
 22  Foreign Investment      557 non-null    float64
dtypes: float64(20), int64(2), object(1)
memory usage: 100.2+ KB
```

[255]:
```
final_data.columns
```

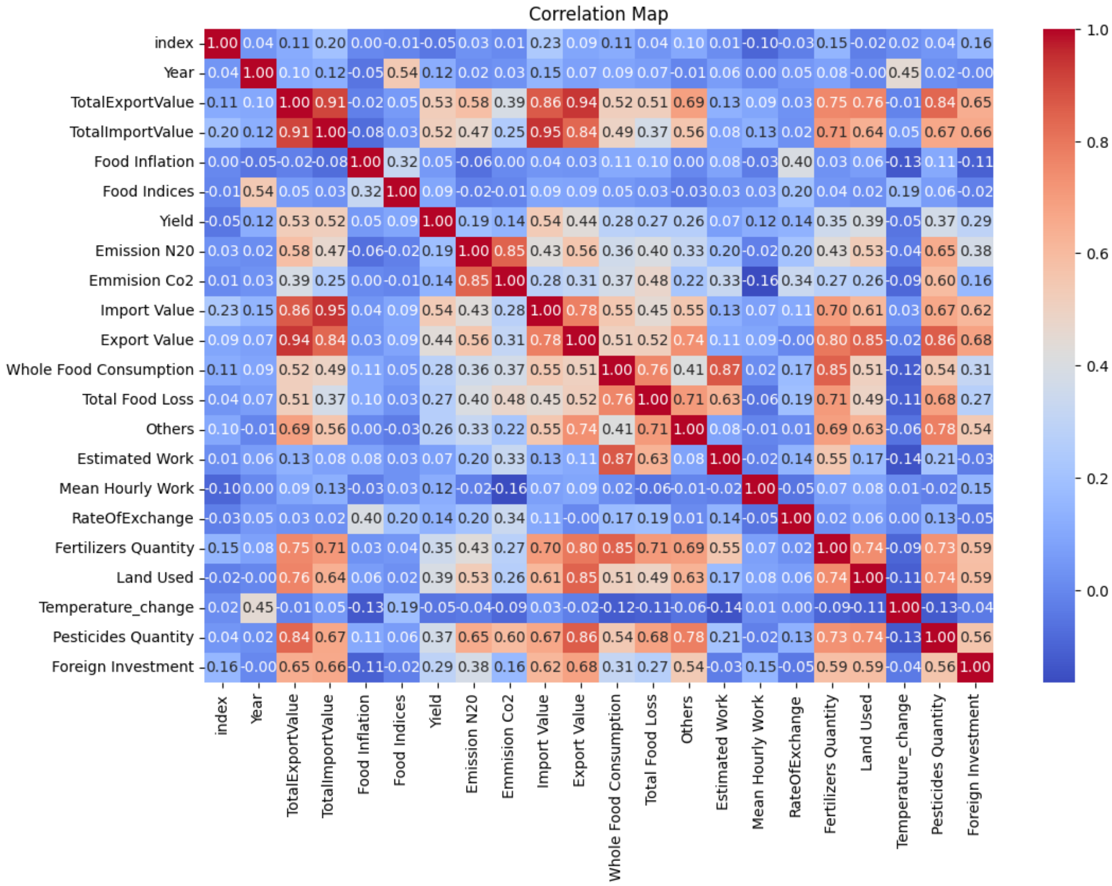
```
[255]: Index(['index', 'Area', 'Year', 'TotalExportValue', 'TotalImportValue',
       'Food Inflation', 'Food Indices', 'Yield', 'Emission N20',
       'Emmision Co2', 'Import Value', 'Export Value',
       'Whole Food Consumption', 'Total Food Loss', 'Others', 'Estimated Work',
       'Mean Hourly Work', 'RateOfExchange', 'Fertilizers Quantity',
       'Land Used', 'Temperature_change', 'Pesticides Quantity',
       'Foreign Investment'],
      dtype='object')
```

```
[256]: final_data.hist(figsize=(10, 6), bins=20)#hist plot
plt.tight_layout()
plt.show()
```




```
[257]: numeric_data = final_data.select_dtypes(include=['float64', 'int64'])
# Calculate correlation matrix
correlation_matrix = numeric_data.corr()

# Plot correlation map
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Correlation Map')
plt.show()
```



feature engineering

```
[258]: data_demo = final_data[[ 'TotalExportValue', 'TotalImportValue',
                'Yield', 'Emission N20',
                'Import Value', 'Export Value',
              'Whole Food Consumption',
              'Land Used']]#dropping the columms that we dont need for model feeding
```

```
[259]: data_demo = data_demo.dropna()
       data_demo= data_demo.reset_index()
       data_demo.info(20)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 557 entries, 0 to 556
Data columns (total 9 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   index                   557 non-null    int64
 1   TotalExportValue        557 non-null    float64
 2   TotalImportValue        557 non-null    float64
```

```
 3   Yield                    557 non-null    float64
 4   Emission N20             557 non-null    float64
 5   Import Value             557 non-null    float64
 6   Export Value             557 non-null    float64
 7   Whole Food Consumption   557 non-null    float64
 8   Land Used                557 non-null    float64
dtypes: float64(8), int64(1)
memory usage: 39.3 KB
```

[260]:
```
data_demo.head(5)
```

[260]:
```
   index  TotalExportValue  TotalImportValue      Yield  Emission N20  \
0      0          54423.00         716893.00  1130203.0        0.0750
1      1          79076.00         756193.00  1194746.0        0.0750
2      2          80834.00         725009.00  1238879.0        0.0750
3      3         104789.00         745689.00  1181126.0        0.0745
4      4          66876.64         476709.33  1245786.0        0.0745

   Import Value  Export Value  Whole Food Consumption  Land Used
0         930.0          41.0                  1932.0  11280.900
1         939.0          56.0                  2002.0  11276.600
2         906.0          72.0                  2023.0  11275.400
3         941.0          87.0                  2014.0  11240.438
4         853.0          92.0                  2006.0  11104.940
```

TotalExportValue TotalImportValue Import Value Export Value Whole Food Consumption Emission N2O Yield

[261]:
```
final_data.columns
```

[261]:
```
Index(['index', 'Area', 'Year', 'TotalExportValue', 'TotalImportValue',
       'Food Inflation', 'Food Indices', 'Yield', 'Emission N20',
       'Emmision Co2', 'Import Value', 'Export Value',
       'Whole Food Consumption', 'Total Food Loss', 'Others', 'Estimated Work',
       'Mean Hourly Work', 'RateOfExchange', 'Fertilizers Quantity',
       'Land Used', 'Temperature_change', 'Pesticides Quantity',
       'Foreign Investment'],
      dtype='object')
```

[262]:
```
final_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 557 entries, 0 to 556
Data columns (total 23 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   index                    557 non-null    int64
 1   Area                     557 non-null    object
```

```
 2   Year                     557 non-null    int64
 3   TotalExportValue         557 non-null    float64
 4   TotalImportValue         557 non-null    float64
 5   Food Inflation           557 non-null    float64
 6   Food Indices             557 non-null    float64
 7   Yield                    557 non-null    float64
 8   Emission N20             557 non-null    float64
 9   Emmision Co2             557 non-null    float64
 10  Import Value             557 non-null    float64
 11  Export Value             557 non-null    float64
 12  Whole Food Consumption   557 non-null    float64
 13  Total Food Loss          557 non-null    float64
 14  Others                   557 non-null    float64
 15  Estimated Work           557 non-null    float64
 16  Mean Hourly Work         557 non-null    float64
 17  RateOfExchange           557 non-null    float64
 18  Fertilizers Quantity     557 non-null    float64
 19  Land Used                557 non-null    float64
 20  Temperature_change       557 non-null    float64
 21  Pesticides Quantity      557 non-null    float64
 22  Foreign Investment       557 non-null    float64
dtypes: float64(20), int64(2), object(1)
memory usage: 100.2+ KB
```

model training and evaluation

[270]:
```
def preprocess_data(data, n_components):
    # Scale the numeric features using StandardScaler
    scaler_X = StandardScaler()
    scaler_y = StandardScaler()

    # Separate features and target
    X = data.drop(columns=['TotalExportValue']).values
    y = data['TotalExportValue'].values

    # Scale features
    X_scaled = scaler_X.fit_transform(X)
    y_scaled = scaler_y.fit_transform(y.reshape(-1, 1)).flatten()

    # Applying PCA
    pca = PCA(n_components=n_components)
    X_pca = pca.fit_transform(X_scaled)

    return X_pca, y_scaled, scaler_y, pca

# 2. Building the MLP model
class MLPModel(nn.Module):
    def __init__(self, input_dim, hidden_dim1, hidden_dim2):
```

```
        super(MLPModel, self).__init__()
        self.hidden1 = nn.Linear(input_dim, hidden_dim1)
        self.hidden2 = nn.Linear(hidden_dim1, hidden_dim2)
        self.output = nn.Linear(hidden_dim2, 1)
        self.relu = nn.ReLU()
        self.dropout = nn.Dropout(p=0.5)

    def forward(self, x):
        x = self.hidden1(x)
        x = self.relu(x)
        x = self.dropout(x)
        x = self.hidden2(x)
        x = self.relu(x)
        x = self.dropout(x)
        x = self.output(x)
        return x

# 3. Training and evaluating the model with early stopping
def train_model(X_train, y_train, X_val, y_val, hidden_dim1, hidden_dim2,␣
 ↪learning_rate, num_epochs, batch_size, weight_decay, patience):
    # Converting data to PyTorch tensors
    X_train = torch.tensor(X_train, dtype=torch.float32)
    y_train = torch.tensor(y_train, dtype=torch.float32)
    X_val = torch.tensor(X_val, dtype=torch.float32)
    y_val = torch.tensor(y_val, dtype=torch.float32)

    # Creating DataLoader for mini-batch training
    train_dataset = torch.utils.data.TensorDataset(X_train, y_train)
    train_loader = torch.utils.data.DataLoader(train_dataset,␣
 ↪batch_size=batch_size, shuffle=True)

    # Initializing the model, loss function, and optimizer
    input_dim = X_train.shape[1]
    model = MLPModel(input_dim, hidden_dim1, hidden_dim2)
    criterion = nn.MSELoss()
    optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate,␣
 ↪weight_decay=weight_decay)

    # Early stopping parameters
    best_val_loss = float('inf')
    epochs_no_improve = 0


    for epoch in range(num_epochs):
        model.train()
        for X_batch, y_batch in train_loader:
            # Forward pass
```

```
            y_pred = model(X_batch)
            loss = criterion(y_pred, y_batch.view(-1, 1))

            # Backward pass and optimization
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

        # Evaluate on the validation set
        model.eval()
        with torch.no_grad():
            y_val_pred = model(X_val)
            val_loss = criterion(y_val_pred, y_val.view(-1, 1))

        print(f"Epoch [{epoch+1}/{num_epochs}], Training Loss: {loss.item():.
 ↪4f}, Validation Loss: {val_loss.item():.4f}")

        # Early stopping check
        if val_loss.item() < best_val_loss:
            best_val_loss = val_loss.item()
            epochs_no_improve = 0
        else:
            epochs_no_improve += 1
            if epochs_no_improve == patience:
                print("Early stopping triggered")
                break

    return model

# 4. Making predictions on the test set
def predict(model, X_test):
    X_test = torch.tensor(X_test, dtype=torch.float32)
    with torch.no_grad():
        y_pred = model(X_test)
    return y_pred.numpy()

# 5. Evaluate performance
def evaluate_performance(y_test, y_pred, scaler_y):
    y_test = scaler_y.inverse_transform(y_test.reshape(-1, 1)).flatten()
    y_pred = scaler_y.inverse_transform(y_pred.reshape(-1, 1)).flatten()
    mse = mean_squared_error(y_test, y_pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)
    print(f"Test RMSE: {rmse:.4f}, Test MAE: {mae:.4f}, Test R^2: {r2:.4f}")
# 6. Save predictions to CSV
```

```
def cross_validate(data, n_components, k, hidden_dim1, hidden_dim2, learning_rate, num_epochs, batch_size, weight_decay, patience):
    X, y, scaler_y, _ = preprocess_data(data, n_components)
    kf = KFold(n_splits=k, shuffle=True, random_state=42)

    metrics = {'rmse': [], 'mae': [], 'r2': []}

    for train_index, val_index in kf.split(X):
        X_train, X_val = X[train_index], X[val[train_index]]
        y_train, y_val = y[train_index], y[val_index]

        model = train_model(X_train, y_train, X_val, y_val, hidden_dim1, hidden_dim2, learning_rate, num_epochs, batch_size, weight_decay, patience)

        y_val_pred = predict(model, X_val)
        rmse, mae, r2 = evaluate_performance(y_val, y_val_pred, scaler_y)

        metrics['rmse'].append(rmse)
        metrics['mae'].append(mae)
        metrics['r2'].append(r2)

    avg_rmse = np.mean(metrics['rmse'])
    avg_mae = np.mean(metrics['mae'])
    avg_r2 = np.mean(metrics['r2'])

    print(f"Average RMSE: {avg_rmse:.4f}")
    print(f"Average MAE: {avg_mae:.4f}")
    print(f"Average R^2: {avg_r2:.4f}")

def save_predictions_to_csv(instance_ids, y_true, y_pred, filename='predictions.csv'):
    df = pd.DataFrame({
        'InstanceID': instance_ids,
        'TrueValue': y_true,
        'PredictedValue': y_pred.flatten()
    })
    df.to_csv(filename, index=False)

# Main function to run the complete process
def main():
    # Use your existing data
    data = data_demo

    # Preprocess the data with PCA
    n_components = 2 # Number of principal components to keep
    X, y, scaler_y, pca = preprocess_data(data, n_components)
```

```
    # Split the data into train, validation, and test sets
    X_train_val, X_test, y_train_val, y_test = train_test_split(X, y,␣
 ↪test_size=0.2, random_state=42)
    X_train, X_val, y_train, y_val = train_test_split(X_train_val, y_train_val,␣
 ↪test_size=0.2, random_state=42)

    # Set the hyperparameters
    hidden_dim1 = 1000
    hidden_dim2 = 500
    learning_rate = 0.001
    num_epochs = 300
    k = 5
    batch_size = 25 # Adjust as needed500
    weight_decay = 0.01  # L2 regularization term
    patience = 10  # Early stopping patience

    # Train the model
    model = train_model(X_train, y_train, X_val, y_val, hidden_dim1,␣
 ↪hidden_dim2, learning_rate, num_epochs, batch_size, weight_decay, patience)

    # Make predictions on the test set
    y_pred = predict(model, X_test)

    # Evaluate performance
    evaluate_performance(y_test, y_pred, scaler_y)

    # Save predictions to CSV
    instance_ids = np.arange(len(y_test))  # Generate instance IDs
    save_predictions_to_csv(instance_ids, y_test, y_pred)
# Run the main function
main()
```

```
Epoch [1/300], Training Loss: 1.3123, Validation Loss: 0.6086
Epoch [2/300], Training Loss: 0.0041, Validation Loss: 0.2147
Epoch [3/300], Training Loss: 0.1600, Validation Loss: 0.1060
Epoch [4/300], Training Loss: 0.4054, Validation Loss: 0.1444
Epoch [5/300], Training Loss: 0.1935, Validation Loss: 0.1293
Epoch [6/300], Training Loss: 0.2565, Validation Loss: 0.1109
Epoch [7/300], Training Loss: 0.7753, Validation Loss: 0.1962
Epoch [8/300], Training Loss: 0.0117, Validation Loss: 0.0912
Epoch [9/300], Training Loss: 0.0404, Validation Loss: 0.1746
Epoch [10/300], Training Loss: 0.0144, Validation Loss: 0.0975
Epoch [11/300], Training Loss: 0.0238, Validation Loss: 0.1344
Epoch [12/300], Training Loss: 0.4956, Validation Loss: 0.1023
Epoch [13/300], Training Loss: 0.0075, Validation Loss: 0.1853
Epoch [14/300], Training Loss: 0.0762, Validation Loss: 0.1277
Epoch [15/300], Training Loss: 0.0306, Validation Loss: 0.1103
```

```
Epoch [16/300], Training Loss: 0.8881, Validation Loss: 0.1518
Epoch [17/300], Training Loss: 0.0544, Validation Loss: 0.1176
Epoch [18/300], Training Loss: 1.0351, Validation Loss: 0.0964
Early stopping triggered
Test RMSE: 6250084.4849, Test MAE: 3313760.9344, Test R^2: 0.9137
```

[ ]:

[263]:

[263]:

[263]:

[263]:

[263]: