```
import pandas as pd

consumer_prices =
pd.read_csv('/Users/sude_umac/PycharmProjects/Machine Learning
/files/Consumer prices indicators - FAOSTAT_data_en_2-22-
2024(in).csv', encoding='ISO-8859-1')
crops_production =
pd.read_csv('/Users/sude_umac/PycharmProjects/Machine Learning
/files/Crops production indicators - FAOSTAT_data_en_2-22-
2024(in).csv', encoding='ISO-8859-1')

employment = pd.read_csv('/Users/sude_umac/PycharmProjects/Machine
Learning /files/Employment - FAOSTAT_data_en_2-27-2024(in).csv',
encoding='ISO-8859-1')

emissions = pd.read_csv('/Users/sude_umac/PycharmProjects/Machine
Learning /files/Emissions - FAOSTAT_data_en_2-27-2024(in).csv',
encoding='ISO-8859-1')
exchange_rate = pd.read_csv('/Users/sude_umac/PycharmProjects/Machine
Learning /files/Exchange rate - FAOSTAT_data_en_2-22-2024(in).csv',
encoding='ISO-8859-1')
food_balances = pd.read_csv('/Users/sude_umac/PycharmProjects/Machine
Learning /files/Food balances indicators - FAOSTAT_data_en_2-22-
2024(in).csv', encoding='ISO-8859-1')
food_security = pd.read_csv('/Users/sude_umac/PycharmProjects/Machine
Learning /files/Food security indicators  - FAOSTAT_data_en_2-22-
2024(in).csv', encoding='ISO-8859-1')
food_trade = pd.read_csv('/Users/sude_umac/PycharmProjects/Machine
Learning /files/Food trade indicators - FAOSTAT_data_en_2-22-
2024(in).csv', encoding='ISO-8859-1')
land_temp_change =
pd.read_csv('/Users/sude_umac/PycharmProjects/Machine Learning
/files/Land temperature change - FAOSTAT_data_en_2-27-2024(in).csv',
encoding='ISO-8859-1')
land_use = pd.read_csv('/Users/sude_umac/PycharmProjects/Machine
Learning /files/Land use - FAOSTAT_data_en_2-22-2024(in).csv',
encoding='ISO-8859-1', low_memory=False)
fertilizers_use =
pd.read_csv('/Users/sude_umac/PycharmProjects/Machine Learning
/files/Fertilizers use - FAOSTAT_data_en_2-27-2024(in).csv',
encoding='ISO-8859-1')
Pesticides_use = pd.read_csv('/Users/sude_umac/PycharmProjects/Machine
Learning /files/Pesticides use - FAOSTAT_data_en_2-27-2024(in).csv',
encoding='ISO-8859-1')
foreign_investment =
pd.read_csv('/Users/sude_umac/PycharmProjects/Machine Learning
/files/Foreign direct investment - FAOSTAT_data_en_2-27-2024(in).csv',
encoding='ISO-8859-1')

consumer_prices.head(consumer_prices.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 112890 entries, 0 to 112889
Data columns (total 17 columns):
 #   Column            Non-Null Count   Dtype  
---  ------            --------------   -----  
 0   Domain Code       112890 non-null  object 
 1   Domain            112890 non-null  object 
 2   Area Code (M49)   112890 non-null  int64  
 3   Area              112890 non-null  object 
 4   Year Code         112890 non-null  int64  
 5   Year              112890 non-null  int64  
 6   Item Code         112890 non-null  int64  
 7   Item              112890 non-null  object 
 8   Months Code       112890 non-null  int64  
 9   Months            112890 non-null  object 
 10  Element Code      112890 non-null  int64  
 11  Element           112890 non-null  object 
 12  Unit              55227 non-null   object 
 13  Value             112890 non-null  float64
 14  Flag              112890 non-null  object 
 15  Flag Description  112890 non-null  object 
 16  Note              57663 non-null   object 
dtypes: float64(1), int64(6), object(10)
memory usage: 14.6+ MB

       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
...            ...                     ...              ...           .
..
112885          CP  Consumer Price Indices              716   
Zimbabwe
112886          CP  Consumer Price Indices              716   
Zimbabwe
112887          CP  Consumer Price Indices              716   
Zimbabwe
112888          CP  Consumer Price Indices              716   
Zimbabwe
112889          CP  Consumer Price Indices              716   
Zimbabwe
```

```
        Year Code  Year  Item Code  \
0            2000  2000      23013
1            2000  2000      23013
2            2000  2000      23013
3            2000  2000      23013
4            2000  2000      23013
...           ...   ...        ...
112885       2023  2023      23014
112886       2023  2023      23014
112887       2023  2023      23014
112888       2023  2023      23014
112889       2023  2023      23014

                                                 Item  Months Code
Months  \
0       Consumer Prices, Food Indices (2015 = 100)         7001
January
1       Consumer Prices, Food Indices (2015 = 100)         7002
February
2       Consumer Prices, Food Indices (2015 = 100)         7003
March
3       Consumer Prices, Food Indices (2015 = 100)         7004
April
4       Consumer Prices, Food Indices (2015 = 100)         7005
May
...                                                 ...          ...
...
112885                        Food price inflation         7005
May
112886                        Food price inflation         7006
June
112887                        Food price inflation         7007
July
112888                        Food price inflation         7008
August
112889                        Food price inflation         7009
September

        Element Code Element Unit       Value Flag Flag Description  \
0               6125   Value  NaN   24.356332    I    Imputed value
1               6125   Value  NaN   23.636242    I    Imputed value
2               6125   Value  NaN   23.485345    I    Imputed value
3               6125   Value  NaN   24.767194    I    Imputed value
4               6125   Value  NaN   25.956912    I    Imputed value
...              ...     ...  ...         ...  ...              ...
112885          6121   Value    %  116.960656    E  Estimated value
112886          6121   Value    %  255.596454    E  Estimated value
112887          6121   Value    %  103.098144    E  Estimated value
112888          6121   Value    %   70.758637    E  Estimated value
```

```
112889          6121   Value    %   71.437761    E  Estimated value

                      Note
0       base year is 2015
1       base year is 2015
2       base year is 2015
3       base year is 2015
4       base year is 2015
...                   ...
112885                NaN
112886                NaN
112887                NaN
112888                NaN
112889                NaN

[112890 rows x 17 columns]
```

```
consumer_prices.head()
```

```
  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
```

```
# List of columns to drop
columns_to_drop = ['Domain Code', 'Domain', 'Area', 'Item Code',
'Element Code', 'Year', 'Flag', 'Flag Description', 'Note', 'Months']

# Drop the columns if they exist in the DataFrame
consumer_prices_clear = consumer_prices.drop(columns=[col for col in
columns_to_drop if col in consumer_prices.columns])



# Checking for duplicate rows and removing them
consumer_prices_clear = consumer_prices_clear.drop_duplicates()

# Converting 'Item Code' column to categorical data type
consumer_prices_clear['Item'] =
consumer_prices_clear['Item'].astype('category')

# Filling missing values in 'Unit' column with 'Unknown'
consumer_prices_clear['Unit'] =
consumer_prices_clear['Unit'].fillna('Unknown')



#check if item is categorical
print(consumer_prices_clear['Item'].dtype)
```

```
category
```

```
#check if there are missing values
print(consumer_prices_clear.isnull().sum())
```

```
Area Code (M49)    0
Year Code          0
Item               0
Months Code        0
Element            0
Unit               0
Value              0
dtype: int64
```

So both Item code and Element code are categorical data types. they only have 2 unique values in which both item and element code refer to 2 different labels in 'Item' (Consumer Prices, Food Indices (2015 = 100)) and (Food price inflation). So we can drop both of them and only refer to 'Item' column.

```python
# Define a function to convert decimal to percentage
def convert_decimal_to_percentage(row):
    if isinstance(row['Value'], float) and row['Unit'] == '%':
        return row['Value'] * 100
    return row['Value']

# Apply the function to the DataFrame
consumer_prices_clear['Value'] =
consumer_prices_clear.apply(convert_decimal_to_percentage, axis=1)

# Display the cleaned DataFrame
print(consumer_prices_clear.head())
```

```
   Area Code (M49)  Year Code
Item  \
0                4       2000  Consumer Prices, Food Indices (2015 =
100)
1                4       2000  Consumer Prices, Food Indices (2015 =
100)
2                4       2000  Consumer Prices, Food Indices (2015 =
100)
3                4       2000  Consumer Prices, Food Indices (2015 =
100)
4                4       2000  Consumer Prices, Food Indices (2015 =
100)

   Months Code Element     Unit      Value
0         7001   Value  Unknown  24.356332
1         7002   Value  Unknown  23.636242
2         7003   Value  Unknown  23.485345
3         7004   Value  Unknown  24.767194
4         7005   Value  Unknown  25.956912
```

```python
print(consumer_prices_clear.iloc[322])  # Indexing starts from 0, so
we use 322 to get the 323rd row6
```

```
Area Code (M49)                          4
Year Code                             2004
Item               Food price inflation
Months Code                           7002
Element                              Value
Unit                                     %
Value                            862.1524
Name: 322, dtype: object
```

```python
# Group the DataFrame by 'Area', 'Year', and 'Element', and calculate
the mean of 'Value'
group_consumer_prices = consumer_prices_clear.groupby(['Area Code
(M49)', 'Year Code', 'Element'])['Value'].mean().reset_index()
```

```python
# Pivot the DataFrame
pivot_consumer_prices = group_consumer_prices.pivot(index=['Area Code (M49)', 'Year Code'], columns='Element', values='Value')

# Reset the index of the DataFrame
pivot_consumer_prices = pivot_consumer_prices.reset_index()

pivot_consumer_prices
```

```
Element  Area Code (M49)  Year Code        Value
0                      4       2000    26.629848
1                      4       2001   653.981386
2                      4       2002   930.398250
3                      4       2003   725.213777
4                      4       2004   726.528864
...                  ...        ...          ...
4851                 894       2019   596.243903
4852                 894       2020   899.170700
4853                 894       2021  1502.023896
4854                 894       2022   787.094206
4855                 894       2023   415.040265

[4856 rows x 3 columns]
```

```
crops_production
```

```
      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
...           ...                             ...              ...
...
41644         QCL  Crops and livestock products              716
Zimbabwe
41645         QCL  Crops and livestock products              716
Zimbabwe
41646         QCL  Crops and livestock products              716
Zimbabwe
41647         QCL  Crops and livestock products              716
Zimbabwe
41648         QCL  Crops and livestock products              716
Zimbabwe
```

```
       Element Code Element Item Code (CPC)                Item  Year
Code  \
0              5419   Yield           F1717    Cereals, primary
2000
1              5419   Yield           F1717    Cereals, primary
2001
2              5419   Yield           F1717    Cereals, primary
2002
3              5419   Yield           F1717    Cereals, primary
2003
4              5419   Yield           F1717    Cereals, primary
2004
...             ...     ...             ...                 ...
...
41644          5419   Yield           F1735  Vegetables Primary
2018
41645          5419   Yield           F1735  Vegetables Primary
2019
41646          5419   Yield           F1735  Vegetables Primary
2020
41647          5419   Yield           F1735  Vegetables Primary
2021
41648          5419   Yield           F1735  Vegetables Primary
2022

       Year      Unit  Value Flag Flag Description  Note
0      2000  100 g/ha   8063    A  Official figure   NaN
1      2001  100 g/ha  10067    A  Official figure   NaN
2      2002  100 g/ha  16698    A  Official figure   NaN
3      2003  100 g/ha  14580    A  Official figure   NaN
4      2004  100 g/ha  13348    A  Official figure   NaN
...     ...       ...    ...  ...              ...   ...
41644  2018  100 g/ha  66518    E  Estimated value   NaN
41645  2019  100 g/ha  64830    E  Estimated value   NaN
41646  2020  100 g/ha  65628    E  Estimated value   NaN
41647  2021  100 g/ha  66126    E  Estimated value   NaN
41648  2022  100 g/ha  65856    E  Estimated value   NaN

[41649 rows x 15 columns]
```

```
# List of columns to drop
columns_to_drop = ['Domain Code', 'Domain', 'Area', 'Item Code (CPC)',
'Element Code', 'Year', 'Flag', 'Flag Description', 'Note', 'Unit']

# Drop the columns if they exist in the DataFrame
crops_production_clear = crops_production.drop(columns=[col for col in
columns_to_drop if col in crops_production.columns])

# Checking for duplicate rows and removing them
```

```
crops_production_clear = crops_production_clear.drop_duplicates()

# Converting necessary columns to categorical data type
crops_production_clear['Item'] =
crops_production_clear['Item'].astype('category')


# Display the cleaned DataFrame
print(crops_production_clear)
```

```
       Area Code (M49) Element                Item  Year Code  Value
0                   4   Yield    Cereals, primary       2000   8063
1                   4   Yield    Cereals, primary       2001  10067
2                   4   Yield    Cereals, primary       2002  16698
3                   4   Yield    Cereals, primary       2003  14580
4                   4   Yield    Cereals, primary       2004  13348
...               ...     ...                 ...        ...    ...
41644             716   Yield  Vegetables Primary       2018  66518
41645             716   Yield  Vegetables Primary       2019  64830
41646             716   Yield  Vegetables Primary       2020  65628
41647             716   Yield  Vegetables Primary       2021  66126
41648             716   Yield  Vegetables Primary       2022  65856

[41649 rows x 5 columns]
```

```
# Group the DataFrame by 'Area Code (M49)', 'Year Code', and
'Element', and calculate the mean of 'Value'
group_crops_production = crops_production_clear.groupby(['Area Code
(M49)', 'Year Code', 'Element'])['Value'].mean().reset_index()

# Pivot the DataFrame
pivot_crops_production = group_crops_production.pivot(index=['Area
Code (M49)', 'Year Code'], columns='Element', values='Value')

# Reset the index of the DataFrame
pivot_crops_production = pivot_crops_production.reset_index()

pivot_crops_production
```

```
Element  Area Code (M49)  Year Code          Yield
0                      4       2000   60177.909091
1                      4       2001   60701.272727
2                      4       2002   61135.363636
3                      4       2003   61209.181818
4                      4       2004   61449.454545
...                  ...        ...            ...
4582                 894       2018  148768.200000
4583                 894       2019  151648.900000
4584                 894       2020  147976.600000
4585                 894       2021  148215.800000
4586                 894       2022  136029.900000
```

```
[4587 rows x 3 columns]

emissions

      Domain Code                                  Domain  Area Code
(M49)  \
0              GCE                    Emissions from Crops
4
1              GCE                    Emissions from Crops
4
2              GCE                    Emissions from Crops
4
3              GCE                    Emissions from Crops
4
4              GCE                    Emissions from Crops
4
...            ...                                    ...               .
..
28905           GV  Emissions from Drained organic soils
716
28906           GV  Emissions from Drained organic soils
716
28907           GV  Emissions from Drained organic soils
716
28908           GV  Emissions from Drained organic soils
716
28909           GV  Emissions from Drained organic soils
716

               Area  Element Code                        Element Item
Code (CPC)  \
0      Afghanistan         72430  Crops total (Emissions N2O)
F1712
1      Afghanistan         72440  Crops total (Emissions CH4)
F1712
2      Afghanistan         72430  Crops total (Emissions N2O)
F1712
3      Afghanistan         72440  Crops total (Emissions CH4)
F1712
4      Afghanistan         72430  Crops total (Emissions N2O)
F1712
...             ...           ...                           ...
...
28905     Zimbabwe          7273               Emissions (CO2)
6728
28906     Zimbabwe          7230               Emissions (N2O)
6728
28907     Zimbabwe          7273               Emissions (CO2)
6728
```

```
28908     Zimbabwe          7230               Emissions (N2O)
6728
28909     Zimbabwe          7273               Emissions (CO2)
6728

                            Item  Year Code  Year  Source Code
Source Unit  \
0                     All Crops       2000  2000         3050  FAO TIER
1   kt
1                     All Crops       2000  2000         3050  FAO TIER
1   kt
2                     All Crops       2001  2001         3050  FAO TIER
1   kt
3                     All Crops       2001  2001         3050  FAO TIER
1   kt
4                     All Crops       2002  2002         3050  FAO TIER
1   kt
...                         ...        ...   ...          ...         .
..  ...
28905  Grassland organic soils       2019  2019         3050  FAO TIER
1   kt
28906  Grassland organic soils       2020  2020         3050  FAO TIER
1   kt
28907  Grassland organic soils       2020  2020         3050  FAO TIER
1   kt
28908  Grassland organic soils       2021  2021         3050  FAO TIER
1   kt
28909  Grassland organic soils       2021  2021         3050  FAO TIER
1   kt

          Value Flag Flag Description  Note
0        0.7056    E  Estimated value   NaN
1       20.8471    E  Estimated value   NaN
2        0.7054    E  Estimated value   NaN
3       19.2605    E  Estimated value   NaN
4        1.0656    E  Estimated value   NaN
...         ...  ...              ...   ...
28905    0.0000    E  Estimated value   NaN
28906    0.0000    E  Estimated value   NaN
28907    0.0000    E  Estimated value   NaN
28908    0.0000    E  Estimated value   NaN
28909    0.0000    E  Estimated value   NaN

[28910 rows x 17 columns]
```

```
# List of columns to drop
columns_to_drop = ['Domain Code', 'Domain', 'Area', 'Element Code',
'Item Code', 'Element Code', 'Year', 'Flag', 'Flag Description',
'Note', 'Months', 'Source', 'Source Code', 'Unit']
```

```
# Drop the columns if they exist in the DataFrame
emissions_clear = emissions.drop(columns=[col for col in
columns_to_drop if col in emissions.columns])

# Checking for duplicate rows and removing them
emissions_clear = emissions_clear.drop_duplicates()

# Converting necessary columns to categorical data type
emissions_clear['Item'] = emissions_clear['Item'].astype('category')



# Display the cleaned DataFrame
print(emissions_clear)
```

```
       Area Code (M49)                       Element Item Code (CPC)  \
0                    4  Crops total (Emissions N2O)           F1712
1                    4  Crops total (Emissions CH4)           F1712
2                    4  Crops total (Emissions N2O)           F1712
3                    4  Crops total (Emissions CH4)           F1712
4                    4  Crops total (Emissions N2O)           F1712
...                ...                          ...             ...
28905              716              Emissions (CO2)            6728
28906              716              Emissions (N2O)            6728
28907              716              Emissions (CO2)            6728
28908              716              Emissions (N2O)            6728
28909              716              Emissions (CO2)            6728

                           Item  Year Code    Value
0                     All Crops       2000   0.7056
1                     All Crops       2000  20.8471
2                     All Crops       2001   0.7054
3                     All Crops       2001  19.2605
4                     All Crops       2002   1.0656
...                         ...        ...      ...
28905  Grassland organic soils       2019   0.0000
28906  Grassland organic soils       2020   0.0000
28907  Grassland organic soils       2020   0.0000
28908  Grassland organic soils       2021   0.0000
28909  Grassland organic soils       2021   0.0000

[28910 rows x 6 columns]
```

```
# Group the DataFrame by 'Area', 'Year', and 'Element', and calculate
the mean of 'Value'
group_emission = emissions_clear.groupby(['Area Code (M49)', 'Year
Code', 'Element'])['Value'].mean().reset_index()

# Pivot the DataFrame
pivot_emission = group_emission.pivot(index=['Area Code (M49)', 'Year
```

```
Code'], columns='Element', values='Value')

# Reset the index of the DataFrame
pivot_emission = pivot_emission.reset_index()

pivot_emission
```

Element	Area Code (M49)	Year Code	Crops total (Emissions CH4)	\
0	4	2000	20.8471	
1	4	2001	19.2605	
2	4	2002	21.2553	
3	4	2003	23.7017	
4	4	2004	30.3089	
...	...	...	...	
5125	894	2017	6.0887	
5126	894	2018	5.1998	
5127	894	2019	4.1332	
5128	894	2020	5.4800	
5129	894	2021	7.0885	

Element	Crops total (Emissions N2O)	Emissions (CO2)	Emissions (N2O)
0	0.7056	0.00000	0.0000
1	0.7054	0.00000	0.0000
2	1.0656	0.00000	0.0000
3	1.3117	0.00000	0.0000
4	1.0856	0.00000	0.0000
...	...	...	...
5125	0.9435	7228.65865	1.3770
5126	0.6835	7232.02850	1.3775
5127	0.5891	7277.36255	1.3848
5128	0.8914	7283.33290	1.3858
5129	0.9896	7283.33290	1.3858

[5130 rows x 6 columns]

```
# Calculate the mean of the 'Crops total (Emissions CH4)' column
mean_crops_total_emissions = pivot_emission['Crops total (Emissions
CH4)'].mean()
```

```
# Fill the missing values in the 'Crops total (Emissions CH4)' column
with the mean
pivot_emission['Crops total (Emissions CH4)'] = pivot_emission['Crops
total (Emissions CH4)'].fillna(mean_crops_total_emissions)

# Calculate the mean of the 'Crops total (Emissions N2O)' column
mean_crops_total_emissions_n2o = pivot_emission['Crops total
(Emissions N2O)'].mean()

# Fill the missing values in the 'Crops total (Emissions N2O)' column
with the mean
pivot_emission['Crops total (Emissions N2O)'] = pivot_emission['Crops
total (Emissions N2O)'].fillna(mean_crops_total_emissions_n2o)

# Check if there are any missing values in the 'Crops total (Emissions
CH4)' column
missing_values = pivot_emission['Crops total (Emissions
CH4)'].isnull().sum()
print(f"Number of missing values in 'Crops total (Emissions CH4)':
{missing_values}")

# Check if there are any missing values in the 'Crops total (Emissions
N2O)' column
missing_values_n2o = pivot_emission['Crops total (Emissions
N2O)'].isnull().sum()
print(f"Number of missing values in 'Crops total (Emissions N2O)':
{missing_values_n2o}")
```

```
Number of missing values in 'Crops total (Emissions CH4)': 0
Number of missing values in 'Crops total (Emissions N2O)': 0
```

employment

	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
...	...	...	...
5912	OEA	Employment Indicators: Agriculture	716
5913	OEA	Employment Indicators: Agriculture	716

```
5914         OEA  Employment Indicators: Agriculture              716

5915         OEA  Employment Indicators: Agriculture              716

5916         OEA  Employment Indicators: Agriculture              716


              Area  Indicator Code  \
0     Afghanistan           21150
1     Afghanistan           21150
2     Afghanistan           21144
3     Afghanistan           21144
4     Afghanistan           21144
...           ...             ...
5912     Zimbabwe           21144
5913     Zimbabwe           21144
5914     Zimbabwe           21150
5915     Zimbabwe           21150
5916     Zimbabwe           21150

                                                 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
...                                                 ...        ...    .
..
5912  Employment in agriculture, forestry and fishin...          1
Total
5913  Employment in agriculture, forestry and fishin...          1
Total
5914  Mean weekly hours actually worked per employed...          1
Total
5915  Mean weekly hours actually worked per employed...          1
Total
5916  Mean weekly hours actually worked per employed...          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
...         ...   ...           ...     ...          ...
5912       2020  2020          6199   Value         3043
5913       2021  2021          6199   Value         3043
5914       2004  2004          6173   Value         3023
5915       2014  2014          6173   Value         3023
5916       2019  2019          6173   Value         3023

                                        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

...                                       ...      ...      ...  ...

5912             ILO - ILO Modelled Estimates  1000 No  3443.50    X

5913             ILO - ILO Modelled Estimates  1000 No  3512.15    X

5914                     Labour force survey       No    44.14    X

5915                     Labour force survey       No    20.01    X

5916                     Labour force survey       No    30.20    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
...                                       ...
5912  Figure from international organizations
5913  Figure from international organizations
5914  Figure from international organizations
5915  Figure from international organizations
5916  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
...                                                  ...
5912                                                 NaN
5913                                                 NaN
5914  Working time arrangement coverage: Full-time w...
5915  Job coverage: Main job currently held Reposito...
5916  Job coverage: Main job currently held | Break ...

[5917 rows x 19 columns]
```

```
# List of columns to drop
columns_to_drop = ['Domain Code', 'Domain', 'Area', 'Indicator Code',
'Element Code', 'Year', 'Flag', 'Flag Description', 'Note', 'Months',
'Source Code', 'Unit','Sex', 'Sex Code']

# Drop the columns if they exist in the DataFrame
employment_clear = employment.drop(columns=[col for col in
columns_to_drop if col in employment.columns])

# Checking for duplicate rows and removing them
employment_clear = employment_clear.drop_duplicates()

# Converting necessary columns to categorical data type
employment_clear['Indicator'] =
employment_clear['Indicator'].astype('category')
employment_clear['Source'] =
employment_clear['Source'].astype('category')

# Display the cleaned DataFrame
print(employment_clear)
```

```
      Area Code (M49)
Indicator  \
0                   4  Mean weekly hours actually worked per
employed...
1                   4  Mean weekly hours actually worked per
employed...
2                   4  Employment in agriculture, forestry and
fishin...
3                   4  Employment in agriculture, forestry and
fishin...
4                   4  Employment in agriculture, forestry and
fishin...
...               ...                                       .
..
5912              716  Employment in agriculture, forestry and
fishin...
5913              716  Employment in agriculture, forestry and
fishin...
5914              716  Mean weekly hours actually worked per
```

```
employed...
5915              716  Mean weekly hours actually worked per
employed...
5916              716  Mean weekly hours actually worked per
employed...

      Year Code Element                                    Source
Value
0          2014   Value  Household income and expenditure survey
31.68
1          2017   Value  Household income and expenditure survey
29.66
2          2000   Value              ILO - ILO Modelled Estimates
2765.95
3          2001   Value              ILO - ILO Modelled Estimates
2805.54
4          2002   Value              ILO - ILO Modelled Estimates
2897.51
...         ...     ...                                      ...
...
5912       2020   Value              ILO - ILO Modelled Estimates
3443.50
5913       2021   Value              ILO - ILO Modelled Estimates
3512.15
5914       2004   Value                        Labour force survey
44.14
5915       2014   Value                        Labour force survey
20.01
5916       2019   Value                        Labour force survey
30.20

[5917 rows x 6 columns]
```

```
# Group the DataFrame by 'Area Code (M49)', 'Year Code', and
'Element', and calculate the mean of 'Value'
group_employment = employment_clear.groupby(['Area Code (M49)', 'Year
Code', 'Element'])['Value'].mean().reset_index()

# Pivot the DataFrame
pivot_employment = group_employment.pivot(index=['Area Code (M49)',
'Year Code'], columns='Element', values='Value')

# Reset the index of the DataFrame
pivot_employment = pivot_employment.reset_index()

pivot_employment
```

```
Element  Area Code (M49)  Year Code     Value
0                      4       2000  2765.950
1                      4       2001  2805.540
```

```
2                      4       2002  2897.510
3                      4       2003  3093.270
4                      4       2004  3212.460
...                  ...        ...       ...
4212                 894       2017  1517.785
4213                 894       2018  1656.255
4214                 894       2019  1673.465
4215                 894       2020  1818.285
4216                 894       2021  1859.005

[4217 rows x 3 columns]
```

exchange_rate

```
       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
...            ...             ...             ...          ...
103271          PE  Exchange rates             716     Zimbabwe
103272          PE  Exchange rates             716     Zimbabwe
103273          PE  Exchange rates             716     Zimbabwe
103274          PE  Exchange rates             716     Zimbabwe
103275          PE  Exchange rates             716     Zimbabwe

       ISO Currency Code (FAO)               Currency Element Code  \
0                          AFA                Afghani          LCU
1                          AFA                Afghani          LCU
2                          AFA                Afghani          LCU
3                          AFA                Afghani          LCU
4                          AFA                Afghani          LCU
...                        ...                    ...          ...
103271                     ZWD  Zimbabwe Dollar (old)          LCU
103272                     ZWD  Zimbabwe Dollar (old)          LCU
103273                     ZWD  Zimbabwe Dollar (old)          LCU
103274                     ZWD  Zimbabwe Dollar (old)          LCU
103275                     ZWD  Zimbabwe Dollar (old)          LCU

                            Element  Year Code  Year  Months Code
Months  \
0       Local currency units per USD       1980  1980         7001
January
1       Local currency units per USD       1980  1980         7002
February
2       Local currency units per USD       1980  1980         7003
March
3       Local currency units per USD       1980  1980         7004
April
```

```
4       Local currency units per USD       1980  1980         7005
May
...                                  ...        ...   ...          ...
...
103271  Local currency units per USD       2022  2022         7009
September
103272  Local currency units per USD       2022  2022         7010
October
103273  Local currency units per USD       2023  2023         7004
April
103274  Local currency units per USD       2023  2023         7005
May
103275  Local currency units per USD       2023  2023         7006
June

        Unit        Value Flag                          Flag
Description
0        NaN    44.129167    X  Figure from international
organizations
1        NaN    44.129167    X  Figure from international
organizations
2        NaN    44.129167    X  Figure from international
organizations
3        NaN    44.129167    X  Figure from international
organizations
4        NaN    44.129167    X  Figure from international
organizations
...      ...          ...  ...                                   ..
.
103271   NaN   597.970000    X  Figure from international
organizations
103272   NaN   628.716452    X  Figure from international
organizations
103273   NaN   981.370229    X  Figure from international
organizations
103274   NaN  1439.613438    X  Figure from international
organizations
103275   NaN  5482.310800    X  Figure from international
organizations

[103276 rows x 16 columns]
```

```
# List of columns to drop
columns_to_drop = ['Domain Code', 'Domain', 'Area', 'Element Code',
'Year', 'Flag', 'Flag Description', 'Note', 'Months', 'Currency',
'Unit']

# Drop the columns if they exist in the DataFrame
exchange_rate_clear = exchange_rate.drop(columns=[col for col in
columns_to_drop if col in exchange_rate.columns])
```

```
# Checking for duplicate rows and removing them
exchange_rate_clear = exchange_rate_clear.drop_duplicates()

# Converting necessary columns to categorical data type
exchange_rate_clear['ISO Currency Code (FAO)'] =
exchange_rate_clear['ISO Currency Code (FAO)'].astype('category')

# Group the DataFrame by 'Area', 'Year', and 'Element', and calculate
the mean of 'Value'
group_exchange_rate = exchange_rate_clear.groupby(['Area Code (M49)',
'Year Code', 'Element'])['Value'].mean().reset_index()

# Pivot the DataFrame
pivot_exchange_rate = group_exchange_rate.pivot(index=['Area Code
(M49)', 'Year Code'], columns='Element', values='Value')

# Reset the index of the DataFrame
pivot_exchange_rate = pivot_exchange_rate.reset_index()

pivot_exchange_rate
```

```
Element  Area Code (M49)  Year Code  Local currency units per USD
0                      4       1980                     44.129167
1                      4       1981                     49.479902
2                      4       1982                     50.599608
3                      4       1983                     50.599608
4                      4       1984                     50.599606
...                  ...        ...                           ...
8634                 894       2019                     12.890000
8635                 894       2020                     18.344093
8636                 894       2021                     20.018487
8637                 894       2022                     16.937594
8638                 894       2023                     19.799163

[8639 rows x 3 columns]
```

```
fertilizers_use
```

```
      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]
```

```
# List of columns to drop
columns_to_drop = ['Domain Code', 'Domain', 'Area', 'Element Code',
'Year', 'Flag', 'Flag Description', 'Note', 'Months', 'Unit']

# Drop the columns if they exist in the DataFrame
fertilizers_use_clear = fertilizers_use.drop(columns=[col for col in
columns_to_drop if col in fertilizers_use.columns])

# Checking for duplicate rows and removing them
fertilizers_use_clear = fertilizers_use_clear.drop_duplicates()

# Converting necessary columns to categorical data type
fertilizers_use_clear['Item'] =
fertilizers_use_clear['Item'].astype('category')

# Group the DataFrame by 'Area', 'Year', and 'Element', and calculate
the mean of 'Value'
group_fertilizers_use = fertilizers_use_clear.groupby(['Area Code
(M49)', 'Year Code', 'Element'])['Value'].mean().reset_index()

# Pivot the DataFrame
pivot_fertilizers_use = group_fertilizers_use.pivot(index=['Area Code
(M49)', 'Year Code'], columns='Element', values='Value')

# Reset the index of the DataFrame
pivot_fertilizers_use = pivot_fertilizers_use.reset_index()

pivot_fertilizers_use
```

```
Element  Area Code (M49)  Year Code  Agricultural Use
0                      4       2002         17900.000
1                      4       2003         33200.000
2                      4       2004         45000.000
3                      4       2005         20577.000
4                      4       2006         68253.000
...                  ...        ...               ...
```

```
1928                  894       2016        136190.000
1929                  894       2017        180547.500
1930                  894       2018        130232.925
1931                  894       2019        184999.795
1932                  894       2021        194259.350

[1933 rows x 3 columns]
```

food_balances

```
       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
...             ...                    ...              ...           ..
.
148036          FBS  Food Balances (2010-)              716
Zimbabwe
148037          FBS  Food Balances (2010-)              716
Zimbabwe
148038          FBS  Food Balances (2010-)              716
Zimbabwe
148039          FBS  Food Balances (2010-)              716
Zimbabwe
148040          FBS  Food Balances (2010-)              716
Zimbabwe

        Element Code          Element Item Code (FBS)  \
0               5611  Import Quantity           S2905
1               5611  Import Quantity           S2905
2               5611  Import Quantity           S2905
3               5611  Import Quantity           S2905
4               5611  Import Quantity           S2905
...              ...              ...             ...
148036          5142             Food           S2960
148037          5142             Food           S2960
148038          5142             Food           S2960
148039          5142             Food           S2960
148040          5142             Food           S2960

                           Item  Year Code  Year    Unit    Value
Flag  \
```

```
0       Cereals - Excluding Beer       2010  2010  1000 t  2000.00
E
1       Cereals - Excluding Beer       2011  2011  1000 t  2448.00
E
2       Cereals - Excluding Beer       2012  2012  1000 t  2001.00
E
3       Cereals - Excluding Beer       2013  2013  1000 t  2155.00
E
4       Cereals - Excluding Beer       2014  2014  1000 t  1840.00
E
...                          ...        ...   ...     ...      ...  ..
.
148036             Fish, Seafood       2017  2017  1000 t    57.96
E
148037             Fish, Seafood       2018  2018  1000 t    46.91
E
148038             Fish, Seafood       2019  2019  1000 t    31.08
E
148039             Fish, Seafood       2020  2020  1000 t    31.08
E
148040             Fish, Seafood       2021  2021  1000 t    31.08
E

       Flag Description
0       Estimated value
1       Estimated value
2       Estimated value
3       Estimated value
4       Estimated value
...                 ...
148036  Estimated value
148037  Estimated value
148038  Estimated value
148039  Estimated value
148040  Estimated value

[148041 rows x 14 columns]
```

```
# List of columns to drop
columns_to_drop = ['Domain Code', 'Domain', 'Area', 'Element Code',
'Year', 'Flag', 'Flag Description', 'Note', 'Months', 'Unit']

# Drop the columns if they exist in the DataFrame
food_balances_clear = food_balances.drop(columns=[col for col in
columns_to_drop if col in food_balances.columns])

# Checking for duplicate rows and removing them
food_balances_clear = food_balances_clear.drop_duplicates()

# Converting necessary columns to categorical data type
```

```
food_balances_clear['Item'] =
food_balances_clear['Item'].astype('category')

# Group the DataFrame by 'Area', 'Year', and 'Element', and calculate
the mean of 'Value'
group_food_balances = food_balances_clear.groupby(['Area Code (M49)',
'Year Code', 'Element'])['Value'].mean().reset_index()

# Pivot the DataFrame
pivot_food_balances = group_food_balances.pivot(index=['Area Code
(M49)', 'Year Code'], columns='Element', values='Value')

# Reset the index of the DataFrame
pivot_food_balances = pivot_food_balances.reset_index()

pivot_food_balances
```

Element	Area Code (M49)	Year Code	Export Quantity	Food	\
0	4	2010	32.727273	635.203750	
1	4	2011	25.181818	644.437500	
2	4	2012	18.000000	685.173750	
3	4	2013	25.545455	712.472500	
4	4	2014	25.750000	866.460000	
...	...	...	...	...	
2172	894	2017	54.221765	416.381765	
2173	894	2018	34.518235	471.152500	
2174	894	2019	37.767647	529.385000	
2175	894	2020	48.628125	543.072500	
2176	894	2021	67.708824	539.760000	

Element	Import Quantity	Losses	Other uses (non-food)
0	207.800667	96.166667	107.500000
1	267.642857	77.833333	128.500000
2	267.071429	100.500000	221.000000
3	281.785714	103.333333	214.000000
4	304.764706	111.500000	9.000000
...	...	...	...
2172	35.315882	68.416667	171.200000
2173	35.672941	68.500000	181.222222
2174	37.814118	64.250000	24.875000
2175	34.637647	59.250000	45.000000
2176	37.108235	65.500000	23.777778

```
[2177 rows x 7 columns]

# Calculate the mean of the 'Losses' column
mean_losses = pivot_food_balances['Losses'].mean()

# Fill the missing value in the 'Losses' column with the calculated
mean
```

```
pivot_food_balances['Losses'] =
pivot_food_balances['Losses'].fillna(mean_losses)

# Check if there are any missing values in the 'Losses' column
missing_values_losses = pivot_food_balances['Losses'].isnull().sum()
print(f"Number of missing values in 'Losses':
{missing_values_losses}")
```

```
Number of missing values in 'Losses': 0
```

```
food_security
```

```
      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

...           ...                                 ...              ...

36507          FS  Suite of Food Security Indicators              716

36508          FS  Suite of Food Security Indicators              716

36509          FS  Suite of Food Security Indicators              716

36510          FS  Suite of Food Security Indicators              716

36511          FS  Suite of Food Security Indicators              716


               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
...             ...           ...     ...        ...
36507      Zimbabwe          6121   Value      21049
36508      Zimbabwe          6121   Value      21049
36509      Zimbabwe          6121   Value      21049
36510      Zimbabwe          6121   Value      21049
36511      Zimbabwe          6121   Value      21049

                                                  Item  Year Code  \
```

```
0      Average dietary energy supply adequacy (percen...   20002002
1      Average dietary energy supply adequacy (percen...   20012003
2      Average dietary energy supply adequacy (percen...   20022004
3      Average dietary energy supply adequacy (percen...   20032005
4      Average dietary energy supply adequacy (percen...   20042006
...                                                  ...        ...
36507            Prevalence of low birthweight (percent)       2016
36508            Prevalence of low birthweight (percent)       2017
36509            Prevalence of low birthweight (percent)       2018
36510            Prevalence of low birthweight (percent)       2019
36511            Prevalence of low birthweight (percent)       2020

             Year Unit  Value Flag                            Flag
Description  \
0      2000-2002    %   88.0    E                            Estimated
value
1      2001-2003    %   89.0    E                            Estimated
value
2      2002-2004    %   92.0    E                            Estimated
value
3      2003-2005    %   93.0    E                            Estimated
value
4      2004-2006    %   94.0    E                            Estimated
value
...           ...  ...    ...  ...
...
36507       2016    %   12.1    X  Figure from international
organizations
36508       2017    %   12.0    X  Figure from international
organizations
36509       2018    %   12.0    X  Figure from international
organizations
36510       2019    %   11.9    X  Figure from international
organizations
36511       2020    %   11.8    X  Figure from international
organizations

      Note
0      NaN
1      NaN
2      NaN
3      NaN
4      NaN
...    ...
36507  NaN
36508  NaN
36509  NaN
36510  NaN
36511  NaN
```

```
[36512 rows x 15 columns]
```

```
# List of columns to drop
columns_to_drop = ['Domain Code', 'Domain', 'Area', 'Element Code',
'Year', 'Flag', 'Flag Description', 'Note', 'Months', 'Unit']

# Drop the columns if they exist in the DataFrame
food_security_clear = food_security.drop(columns=[col for col in
columns_to_drop if col in food_security.columns])

# Checking for duplicate rows and removing them
food_security_clear = food_security_clear.drop_duplicates()

# Converting necessary columns to categorical data type
food_security_clear['Item'] =
food_security_clear['Item'].astype('category')

# Group the DataFrame by 'Area', 'Year', and 'Element', and calculate
the mean of 'Value'
group_food_security = food_security_clear.groupby(['Area Code (M49)',
'Year Code', 'Element'])['Value'].mean().reset_index()

# Pivot the DataFrame
pivot_food_security = group_food_security.pivot(index=['Area Code
(M49)', 'Year Code'], columns='Element', values='Value')

# Reset the index of the DataFrame
pivot_food_security = pivot_food_security.reset_index()

pivot_food_security
```

```
Element  Area Code (M49)  Year Code       Value
0                      4       2000   30.420000
1                      4       2001   32.966667
2                      4       2002   31.340000
3                      4       2003   31.500000
4                      4       2004   25.125000
...                  ...        ...         ...
8575                 894   20162018   31.160000
8576                 894   20172019   32.180000
8577                 894   20182020   32.280000
8578                 894   20192021   35.733333
8579                 894   20202022  101.000000

[8580 rows x 3 columns]
```

```
food_trade
```

```
      Domain Code                       Domain  Area Code (M49)  \
0             TCL  Crops and livestock products                4
```

```
1              TCL  Crops and livestock products                4
2              TCL  Crops and livestock products                4
3              TCL  Crops and livestock products                4
4              TCL  Crops and livestock products                4
...            ...                              ...            ...
141733         TCL  Crops and livestock products              716
141734         TCL  Crops and livestock products              716
141735         TCL  Crops and livestock products              716
141736         TCL  Crops and livestock products              716
141737         TCL  Crops and livestock products              716

                Area  Element Code       Element Item Code (CPC)  \
0       Afghanistan          5622  Import Value           F1888
1       Afghanistan          5622  Import Value           F1888
2       Afghanistan          5622  Import Value           F1888
3       Afghanistan          5622  Import Value           F1888
4       Afghanistan          5622  Import Value           F1888
...             ...           ...           ...             ...
141733     Zimbabwe          5922  Export Value           F1896
141734     Zimbabwe          5622  Import Value           F1896
141735     Zimbabwe          5922  Export Value           F1896
141736     Zimbabwe          5622  Import Value           F1896
141737     Zimbabwe          5922  Export Value           F1896

                            Item  Year Code  Year      Unit      Value
Flag  \
0       Cereals and Preparations       1991  1991  1000 USD   41600.00
A
1       Cereals and Preparations       1992  1992  1000 USD   25600.00
E
2       Cereals and Preparations       1993  1993  1000 USD   40000.00
E
3       Cereals and Preparations       1994  1994  1000 USD   25700.00
E
4       Cereals and Preparations       1995  1995  1000 USD   37720.00
E
...                          ...        ...   ...       ...        ...
...
141733                  Tobacco       2020  2020  1000 USD  794956.99
A
141734                  Tobacco       2021  2021  1000 USD   18265.04
A
141735                  Tobacco       2021  2021  1000 USD  836533.69
A
141736                  Tobacco       2022  2022  1000 USD   27138.09
A
141737                  Tobacco       2022  2022  1000 USD  998057.60
A

       Flag Description  Note
```

```
0       Official figure   NaN
1       Estimated value   NaN
2       Estimated value   NaN
3       Estimated value   NaN
4       Estimated value   NaN
...                 ...   ...
141733  Official figure   NaN
141734  Official figure   NaN
141735  Official figure   NaN
141736  Official figure   NaN
141737  Official figure   NaN

[141738 rows x 15 columns]
```

```
# List of columns to drop
columns_to_drop = ['Domain Code', 'Domain', 'Area', 'Element Code',
'Year', 'Flag', 'Flag Description', 'Note', 'Months', 'Unit']

# Drop the columns if they exist in the DataFrame
food_trade_clear = food_trade.drop(columns=[col for col in
columns_to_drop if col in food_trade.columns])

# Checking for duplicate rows and removing them
food_trade_clear = food_trade_clear.drop_duplicates()

# Converting necessary columns to categorical data type
food_trade_clear['Item'] = food_trade_clear['Item'].astype('category')

# Group the DataFrame by 'Area', 'Year', and 'Element', and calculate
the mean of 'Value'
group_food_trade = food_trade_clear.groupby(['Area Code (M49)', 'Year
Code', 'Element'])['Value'].mean().reset_index()

# Pivot the DataFrame
pivot_food_trade = group_food_trade.pivot(index=['Area Code (M49)',
'Year Code'], columns='Element', values='Value')

# Reset the index of the DataFrame
pivot_food_trade = pivot_food_trade.reset_index()

pivot_food_trade
```

```
Element  Area Code (M49)  Year Code   Export Value  Import Value
0                      4       1991   19648.600000  10859.416667
1                      4       1992    8422.400000  11130.416667
2                      4       1993    8912.800000  11349.666667
3                      4       1994   10071.400000   9683.083333
4                      4       1995    9919.200000  17950.916667
...                  ...        ...            ...           ...
6200                 894       2018   61485.388333  33055.448333
6201                 894       2019   54332.815000  34154.626667
```

```
6202                  894       2020   62173.202500  31061.486667
6203                  894       2021   85032.921667  42894.053333
6204                  894       2022  102246.105833  46604.015833

[6205 rows x 4 columns]
```

```
# Calculate the mean of the 'Export Value' column
mean_export_value = pivot_food_trade['Export Value'].mean()

# Fill the missing values in the 'Export Value' column with the mean
pivot_food_trade['Export Value'] = pivot_food_trade['Export
Value'].fillna(mean_export_value)


# Check if there are any missing values in the 'Export Value' column
missing_values_export_value = pivot_food_trade['Export
Value'].isnull().sum()
print(f"Number of missing values in 'Export Value':
{missing_values_export_value}")
```

```
Number of missing values in 'Export Value': 0
```

```
foreign_investment
```

```
      Domain Code                            Domain  Area Code (M49)  \
0             FDI  Foreign Direct Investment (FDI)                4
1             FDI  Foreign Direct Investment (FDI)                4
2             FDI  Foreign Direct Investment (FDI)                4
3             FDI  Foreign Direct Investment (FDI)                4
4             FDI  Foreign Direct Investment (FDI)                4
...           ...                              ...              ...
12271         FDI  Foreign Direct Investment (FDI)              716
12272         FDI  Foreign Direct Investment (FDI)              716
12273         FDI  Foreign Direct Investment (FDI)              716
12274         FDI  Foreign Direct Investment (FDI)              716
12275         FDI  Foreign Direct Investment (FDI)              716

               Area  Element Code    Element  Item Code
Item  \
0      Afghanistan          6110  Value US$      23082   Total FDI
inflows
1      Afghanistan          6110  Value US$      23082   Total FDI
inflows
2      Afghanistan          6110  Value US$      23082   Total FDI
inflows
3      Afghanistan          6110  Value US$      23082   Total FDI
inflows
4      Afghanistan          6110  Value US$      23082   Total FDI
inflows
...            ...           ...        ...        ...
...
```

```
12271     Zimbabwe          6110  Value US$      23085  Total FDI
outflows
12272     Zimbabwe          6110  Value US$      23085  Total FDI
outflows
12273     Zimbabwe          6110  Value US$      23085  Total FDI
outflows
12274     Zimbabwe          6110  Value US$      23085  Total FDI
outflows
12275     Zimbabwe          6110  Value US$      23085  Total FDI
outflows

       Year Code  Year         Unit       Value Flag  \
0           2000  2000  million USD    0.170000    X
1           2001  2001  million USD    0.680000    X
2           2002  2002  million USD   50.000000    X
3           2003  2003  million USD   57.800000    X
4           2004  2004  million USD  186.900000    X
...          ...   ...          ...         ...  ...
12271       2018  2018  million USD   26.771877    X
12272       2019  2019  million USD   32.000000    X
12273       2020  2020  million USD   33.000000    X
12274       2021  2021  million USD   32.000000    X
12275       2022  2022  million USD   17.000000    X

                              Flag Description    Note
0      Figure from international organizations  UNCTAD
1      Figure from international organizations  UNCTAD
2      Figure from international organizations  UNCTAD
3      Figure from international organizations  UNCTAD
4      Figure from international organizations  UNCTAD
...                                        ...     ...
12271  Figure from international organizations  UNCTAD
12272  Figure from international organizations  UNCTAD
12273  Figure from international organizations  UNCTAD
12274  Figure from international organizations  UNCTAD
12275  Figure from international organizations  UNCTAD

[12276 rows x 15 columns]
```

```
# List of columns to drop
columns_to_drop = ['Domain Code', 'Domain', 'Area', 'Element Code',
'Year', 'Flag', 'Flag Description', 'Note', 'Months', 'Unit']

# Drop the columns if they exist in the DataFrame
foreign_investment_clear = foreign_investment.drop(columns=[col for
col in columns_to_drop if col in foreign_investment.columns])

# Checking for duplicate rows and removing them
foreign_investment_clear = foreign_investment_clear.drop_duplicates()
```

```
# Converting necessary columns to categorical data type
foreign_investment_clear['Item'] =
foreign_investment_clear['Item'].astype('category')


# Group the DataFrame by 'Area', 'Year', and 'Element', and calculate
the mean of 'Value'
group_foreign_investment = foreign_investment_clear.groupby(['Area
Code (M49)', 'Year Code', 'Element'])['Value'].mean().reset_index()

# Pivot the DataFrame
pivot_foreign_investment = group_foreign_investment.pivot(index=['Area
Code (M49)', 'Year Code'], columns='Element', values='Value')

# Reset the index of the DataFrame
pivot_foreign_investment = pivot_foreign_investment.reset_index()

pivot_foreign_investment
```

```
Element  Area Code (M49)  Year Code   Value US$
0                      4       2000    0.170000
1                      4       2001    0.680000
2                      4       2002   50.000000
3                      4       2003   29.400000
4                      4       2004   93.100000
...                  ...        ...         ...
4575                 894       2018  117.007475
4576                 894       2019  495.333333
4577                 894       2020  113.466667
4578                 894       2021  -92.100000
4579                 894       2022 -135.934725

[4580 rows x 3 columns]
```

```
land_temp_change
```

```
      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
...           ...                         ...              ...
...
54805          ET  Temperature change on land              716
```

```
Zimbabwe
54806          ET  Temperature change on land              716
Zimbabwe
54807          ET  Temperature change on land              716
Zimbabwe
54808          ET  Temperature change on land              716
Zimbabwe
54809          ET  Temperature change on land              716
Zimbabwe

       Element Code              Element  Months Code
Months  \
0              7271  Temperature change         7016
Dec–Jan–Feb
1              7271  Temperature change         7016
Dec–Jan–Feb
2              7271  Temperature change         7016
Dec–Jan–Feb
3              7271  Temperature change         7016
Dec–Jan–Feb
4              7271  Temperature change         7016
Dec–Jan–Feb
...             ...                 ...          ...
...
54805          6078  Standard Deviation         7020  Meteorological
year
54806          6078  Standard Deviation         7020  Meteorological
year
54807          6078  Standard Deviation         7020  Meteorological
year
54808          6078  Standard Deviation         7020  Meteorological
year
54809          6078  Standard Deviation         7020  Meteorological
year

       Year Code  Year Unit  Value Flag Flag Description
0            2000  2000   °c  0.618    E  Estimated value
1            2001  2001   °c  0.365    E  Estimated value
2            2002  2002   °c  1.655    E  Estimated value
3            2003  2003   °c  0.997    E  Estimated value
4            2004  2004   °c  1.883    E  Estimated value
...           ...   ...  ...    ...  ...              ...
54805        2018  2018   °c  0.311    E  Estimated value
54806        2019  2019   °c  0.311    E  Estimated value
54807        2020  2020   °c  0.311    E  Estimated value
54808        2021  2021   °c  0.311    E  Estimated value
54809        2022  2022   °c  0.311    E  Estimated value

[54810 rows x 14 columns]
```

```python
# List of columns to drop
columns_to_drop = ['Domain Code', 'Domain', 'Area', 'Element Code',
'Year', 'Flag', 'Flag Description', 'Note', 'Months Code', 'Unit']

# Drop the columns if they exist in the DataFrame
land_temp_change_clear = land_temp_change.drop(columns=[col for col in
columns_to_drop if col in land_temp_change.columns])

# Checking for duplicate rows and removing them
land_temp_change_clear = land_temp_change_clear.drop_duplicates()

# Converting necessary columns to categorical data type
land_temp_change_clear['Months'] =
land_temp_change_clear['Months'].astype('category')

# Display the cleaned DataFrame
print(land_temp_change_clear)
```

```
       Area Code (M49)             Element                Months  Year
Code  \
0                    4  Temperature change          Dec–Jan–Feb
2000
1                    4  Temperature change          Dec–Jan–Feb
2001
2                    4  Temperature change          Dec–Jan–Feb
2002
3                    4  Temperature change          Dec–Jan–Feb
2003
4                    4  Temperature change          Dec–Jan–Feb
2004
...                ...                ...                  ...
...
54805              716  Standard Deviation  Meteorological year
2018
54806              716  Standard Deviation  Meteorological year
2019
54807              716  Standard Deviation  Meteorological year
2020
54808              716  Standard Deviation  Meteorological year
2021
54809              716  Standard Deviation  Meteorological year
2022

       Value
0      0.618
1      0.365
2      1.655
3      0.997
4      1.883
```

```
...      ...
54805  0.311
54806  0.311
54807  0.311
54808  0.311
54809  0.311

[54810 rows x 5 columns]
```

```
# Group the DataFrame by 'Area', 'Year', and 'Element', and calculate
the mean of 'Value'
group_land_temp_change = land_temp_change_clear.groupby(['Area Code
(M49)', 'Year Code', 'Element'])['Value'].mean().reset_index()

# Pivot the DataFrame
pivot_land_temp_change = group_land_temp_change.pivot(index=['Area
Code (M49)', 'Year Code'], columns='Element', values='Value')

# Reset the index of the DataFrame
pivot_land_temp_change = pivot_land_temp_change.reset_index()

pivot_land_temp_change
```

```
Element  Area Code (M49)  Year Code  Standard Deviation  Temperature
change
0                      4       2000              0.8326
0.9930
1                      4       2001              0.8326
1.3110
2                      4       2002              0.8326
1.3650
3                      4       2003              0.8326
0.5870
4                      4       2004              0.8326
1.3732
...                  ...        ...                 ...
...
5476                 894       2018              0.3636
0.6482
5477                 894       2019              0.3636
0.8548
5478                 894       2020              0.3636
0.8912
5479                 894       2021              0.3636
0.8218
5480                 894       2022              0.3636
0.6864

[5481 rows x 4 columns]
```

```
print(pivot_land_temp_change.isnull().sum())
```

```
Element
Area Code (M49)           0
Year Code                 0
Standard Deviation     1022
Temperature change      213
dtype: int64
```

```
# Calculate the mean of the 'Standard Deviation' column
mean_std_dev = pivot_land_temp_change['Standard Deviation'].mean()

# Fill the missing values in the 'Standard Deviation' column with the
mean
pivot_land_temp_change['Standard Deviation'] =
pivot_land_temp_change['Standard Deviation'].fillna(mean_std_dev)

# Calculate the mean of the 'Temperature change' column
mean_temp_change = pivot_land_temp_change['Temperature change'].mean()

# Fill the missing values in the 'Temperature change' column with the
mean
pivot_land_temp_change['Temperature change'] =
pivot_land_temp_change['Temperature change'].fillna(mean_temp_change)

# Check if there are any missing values in the 'Standard Deviation'
column
missing_values = pivot_land_temp_change['Standard
Deviation'].isnull().sum()
print(f"Number of missing values in 'Standard Deviation':
{missing_values}")

# Check if there are any missing values in the 'Temperature change'
column
missing_values_temp_change = pivot_land_temp_change['Temperature
change'].isnull().sum()
print(f"Number of missing values in 'Temperature change':
{missing_values_temp_change}")
```

```
Number of missing values in 'Standard Deviation': 0
Number of missing values in 'Temperature change': 0
```

```
land_use
```

```
      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]
```

```python
# List of columns to drop
columns_to_drop = ['Domain Code', 'Domain', 'Area', 'Element Code',
'Year', 'Flag', 'Flag Description', 'Note', 'Months Code', 'Unit',
'Item Code']

# Drop the columns if they exist in the DataFrame
land_use_clear = land_use.drop(columns=[col for col in columns_to_drop
if col in land_use.columns])

# Checking for duplicate rows and removing them
land_use_clear = land_use_clear.drop_duplicates()

# Converting necessary columns to categorical data type
land_use_clear['Item'] = land_use_clear['Item'].astype('category')

# Display the cleaned DataFrame
print(land_use_clear)
```

```
       Area Code (M49) Element                                 Item
Year Code  \
0                    4    Area                         Country area
1980
1                    4    Area                         Country area
1981
2                    4    Area                         Country area
1982
3                    4    Area                         Country area
1983
4                    4    Area                         Country area
1984
...                ...     ...                                  ...
```

```
...
97990               716    Area  Land area equipped for irrigation
2017
97991               716    Area  Land area equipped for irrigation
2018
97992               716    Area  Land area equipped for irrigation
2019
97993               716    Area  Land area equipped for irrigation
2020
97994               716    Area  Land area equipped for irrigation
2021

          Value
0      65286.0
1      65286.0
2      65286.0
3      65286.0
4      65286.0
...        ...
97990    186.6
97991    186.6
97992    186.6
97993    186.6
97994    186.6

[97995 rows x 5 columns]
```

```
# Group the DataFrame by 'Area', 'Year', and 'Element', and calculate the mean of 'Value'
group_land_use = land_use_clear.groupby(['Area Code (M49)', 'Year Code', 'Element'])['Value'].mean().reset_index()

# Pivot the DataFrame
pivot_land_use = group_land_use.pivot(index=['Area Code (M49)', 'Year Code'], columns='Element', values='Value')

# Reset the index of the DataFrame
pivot_land_use = pivot_land_use.reset_index()

pivot_land_use
```

```
Element  Area Code (M49)  Year Code          Area
0                      4       1980  28356.666667
1                      4       1981  28360.111111
2                      4       1982  28362.222222
3                      4       1983  28363.888889
4                      4       1984  28367.333333
...                  ...        ...           ...
9514                 894       2017  19075.999992
9515                 894       2018  19076.000000
```

```
9516                  894       2019  19076.000000
9517                  894       2020  19076.000000
9518                  894       2021  19076.000000

[9519 rows x 3 columns]

Pesticides_use

      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
...           ...             ...             ...       ...
...
35197          RP  Pesticides Use             716  Zimbabwe
5157
35198          RP  Pesticides Use             716  Zimbabwe
5157
35199          RP  Pesticides Use             716  Zimbabwe
5157
35200          RP  Pesticides Use             716  Zimbabwe
5157
35201          RP  Pesticides Use             716  Zimbabwe
5157

                                       Element  Item Code  \
0                           Agricultural Use       1357
1                     Use per area of cropland       1357
2      Use per value of agricultural production       1357
3                           Agricultural Use       1357
4                     Use per area of cropland       1357
...                                        ...        ...
35197                       Agricultural Use       1345
35198                       Agricultural Use       1345
35199                       Agricultural Use       1345
35200                       Agricultural Use       1345
35201                       Agricultural Use       1345

                     Item  Year Code  Year    Unit   Value Flag  \
0      Pesticides (total)       2000  2000       t  307.98    E
1      Pesticides (total)       2000  2000   kg/ha    0.44    E
2      Pesticides (total)       2000  2000  g/Int$    0.23    E
```

```
3      Pesticides (total)       2001  2001       t  319.38    E
4      Pesticides (total)       2001  2001   kg/ha    0.46    E
...                   ...        ...   ...     ...     ...  ...
35197        Rodenticides       2017  2017       t    0.00    I
35198        Rodenticides       2018  2018       t    0.00    I
35199        Rodenticides       2019  2019       t    0.00    I
35200        Rodenticides       2020  2020       t    0.00    I
35201        Rodenticides       2021  2021       t    0.00    I

      Flag Description Note
0      Estimated value  NaN
1      Estimated value  NaN
2      Estimated value  NaN
3      Estimated value  NaN
4      Estimated value  NaN
...                ...  ...
35197    Imputed value  NaN
35198    Imputed value  NaN
35199    Imputed value  NaN
35200    Imputed value  NaN
35201    Imputed value  NaN

[35202 rows x 15 columns]
```

```
# List of columns to drop
columns_to_drop = ['Domain Code', 'Domain', 'Area', 'Element Code',
'Year', 'Flag', 'Flag Description', 'Note', 'Months', 'Unit']

# Drop the columns if they exist in the DataFrame
Pesticides_use_clear = Pesticides_use.drop(columns=[col for col in
columns_to_drop if col in Pesticides_use.columns])

# Checking for duplicate rows and removing them
Pesticides_use_clear = Pesticides_use_clear.drop_duplicates()

# Converting necessary columns to categorical data type
Pesticides_use_clear['Item'] =
Pesticides_use_clear['Item'].astype('category')

# Display the cleaned DataFrame
print(Pesticides_use_clear)
```

```
       Area Code (M49)                                  Element  Item
Code  \
0                   8                        Agricultural Use
1357
1                   8                Use per area of cropland
1357
2                   8  Use per value of agricultural production
1357
```

```
3                      8                            Agricultural Use
1357
4                      8                  Use per area of cropland
1357
...                  ...                                        ...
...
35197                716                            Agricultural Use
1345
35198                716                            Agricultural Use
1345
35199                716                            Agricultural Use
1345
35200                716                            Agricultural Use
1345
35201                716                            Agricultural Use
1345

                       Item  Year Code   Value
0      Pesticides (total)       2000  307.98
1      Pesticides (total)       2000    0.44
2      Pesticides (total)       2000    0.23
3      Pesticides (total)       2001  319.38
4      Pesticides (total)       2001    0.46
...                    ...        ...     ...
35197        Rodenticides       2017    0.00
35198        Rodenticides       2018    0.00
35199        Rodenticides       2019    0.00
35200        Rodenticides       2020    0.00
35201        Rodenticides       2021    0.00

[35202 rows x 6 columns]
```

```
# Group the DataFrame by 'Area', 'Year', and 'Element', and calculate
the mean of 'Value'
group_Pesticides_use = Pesticides_use_clear.groupby(['Area Code
(M49)', 'Year Code', 'Element'])['Value'].mean().reset_index()

# Pivot the DataFrame
pivot_Pesticides_use = group_Pesticides_use.pivot(index=['Area Code
(M49)', 'Year Code'], columns='Element', values='Value')

# Reset the index of the DataFrame
pivot_Pesticides_use = pivot_Pesticides_use.reset_index()

pivot_Pesticides_use
```

```
Element  Area Code (M49)  Year Code  Agricultural Use  \
0                      8       2000         86.842857
1                      8       2001         89.870000
2                      8       2002         92.895714
```

```
3                       8       2003          95.920000
4                       8       2004          98.945714
...                   ...        ...                ...
4631                  894       2017       1739.112000
4632                  894       2018       1678.656000
4633                  894       2019       1678.656000
4634                  894       2020       1678.656000
4635                  894       2021       1678.656000

Element  Use per area of cropland  Use per value of agricultural
production
0                              0.44
0.23
1                              0.46
0.23
2                              0.47
0.24
3                              0.49
0.24
4                              0.51
0.23
...                             ...
...
4631                           1.13
1.16
4632                           1.09
1.18
4633                           1.09
1.23
4634                           1.09
1.14
4635                           1.09
1.09

[4636 rows x 5 columns]
```

```
# Calculate the mean of the 'Use per area of cropland' column
mean_use_per_area = pivot_Pesticides_use['Use per area of
cropland'].mean()

# Fill the missing values in the 'Use per area of cropland' column
with the mean
pivot_Pesticides_use['Use per area of cropland'] =
pivot_Pesticides_use['Use per area of
cropland'].fillna(mean_use_per_area)

#Calculate the mean of the 'Use per value of agricultural production'
column
mean_use_per_value = pivot_Pesticides_use['Use per value of
agricultural production'].mean()
```

```python
# Fill the missing values in the 'Use per value of agricultural production' column with the mean
pivot_Pesticides_use['Use per value of agricultural production'] = pivot_Pesticides_use['Use per value of agricultural production'].fillna(mean_use_per_value)


# Check if there are any missing values in the 'Use per area of cropland' column
missing_values_use_per_area = pivot_Pesticides_use['Use per area of cropland'].isnull().sum()
print(f"Number of missing values in 'Use per area of cropland': {missing_values_use_per_area}")

# Check if there are any missing values in the 'Use per value of agricultural production' column
missing_values_use_per_value = pivot_Pesticides_use['Use per value of agricultural production'].isnull().sum()
print(f"Number of missing values in 'Use per value of agricultural production': {missing_values_use_per_value}")
```

```
Number of missing values in 'Use per area of cropland': 0
Number of missing values in 'Use per value of agricultural production': 0
```

```python
#perfrom Exploratory Data Analysis

import matplotlib.pyplot as plt
import seaborn as sns

dataframes = [pivot_consumer_prices, pivot_crops_production, pivot_emission, pivot_employment, pivot_exchange_rate, pivot_fertilizers_use, pivot_food_balances, pivot_food_security, pivot_food_trade, pivot_foreign_investment, pivot_land_temp_change, pivot_land_use, pivot_Pesticides_use]

for df in dataframes:
    # Understand the data
    print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4856 entries, 0 to 4855
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Area Code (M49)  4856 non-null   int64  
 1   Year Code        4856 non-null   int64  
 2   Value            4856 non-null   float64
```

```
dtypes: float64(1), int64(2)
memory usage: 113.9 KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4587 entries, 0 to 4586
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Area Code (M49)  4587 non-null   int64  
 1   Year Code        4587 non-null   int64  
 2   Yield            4587 non-null   float64
dtypes: float64(1), int64(2)
memory usage: 107.6 KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5130 entries, 0 to 5129
Data columns (total 6 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Area Code (M49)               5130 non-null   int64  
 1   Year Code                     5130 non-null   int64  
 2   Crops total (Emissions CH4)  5130 non-null   float64
 3   Crops total (Emissions N2O)  5130 non-null   float64
 4   Emissions (CO2)               5130 non-null   float64
 5   Emissions (N2O)               5130 non-null   float64
dtypes: float64(4), int64(2)
memory usage: 240.6 KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4217 entries, 0 to 4216
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Area Code (M49)  4217 non-null   int64  
 1   Year Code        4217 non-null   int64  
 2   Value            4217 non-null   float64
dtypes: float64(1), int64(2)
memory usage: 99.0 KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8639 entries, 0 to 8638
Data columns (total 3 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Area Code (M49)               8639 non-null   int64  
 1   Year Code                     8639 non-null   int64  
 2   Local currency units per USD  8639 non-null   float64
dtypes: float64(1), int64(2)
memory usage: 202.6 KB
```

```
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1933 entries, 0 to 1932
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Area Code (M49)   1933 non-null   int64  
 1   Year Code         1933 non-null   int64  
 2   Agricultural Use  1933 non-null   float64
dtypes: float64(1), int64(2)
memory usage: 45.4 KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2177 entries, 0 to 2176
Data columns (total 7 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Area Code (M49)        2177 non-null   int64  
 1   Year Code              2177 non-null   int64  
 2   Export Quantity        2177 non-null   float64
 3   Food                   2177 non-null   float64
 4   Import Quantity        2177 non-null   float64
 5   Losses                 2177 non-null   float64
 6   Other uses (non-food)  2177 non-null   float64
dtypes: float64(5), int64(2)
memory usage: 119.2 KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8580 entries, 0 to 8579
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Area Code (M49)  8580 non-null   int64  
 1   Year Code        8580 non-null   int64  
 2   Value            8580 non-null   float64
dtypes: float64(1), int64(2)
memory usage: 201.2 KB
None
<class 'pandas.core.frame.DataFrame'>
Index: 1933 entries, 0 to 1932
Data columns (total 5 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   Area Code (M49)      1933 non-null   int64  
 1   Year Code            1933 non-null   int64  
 2   Export Value         1933 non-null   float64
 3   Import Value         1933 non-null   float64
 4   Export Value Lagged  1933 non-null   float64
dtypes: float64(3), int64(2)
```

```
memory usage: 90.6 KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4580 entries, 0 to 4579
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Area Code (M49)  4580 non-null   int64  
 1   Year Code        4580 non-null   int64  
 2   Value US$        4580 non-null   float64
dtypes: float64(1), int64(2)
memory usage: 107.5 KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5481 entries, 0 to 5480
Data columns (total 4 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Area Code (M49)     5481 non-null   int64  
 1   Year Code           5481 non-null   int64  
 2   Standard Deviation  5481 non-null   float64
 3   Temperature change  5481 non-null   float64
dtypes: float64(2), int64(2)
memory usage: 171.4 KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9519 entries, 0 to 9518
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Area Code (M49)  9519 non-null   int64  
 1   Year Code        9519 non-null   int64  
 2   Area             9519 non-null   float64
dtypes: float64(1), int64(2)
memory usage: 223.2 KB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4636 entries, 0 to 4635
Data columns (total 5 columns):
 #   Column                                  Non-Null Count  Dtype  
---  ------                                  --------------  -----  
 0   Area Code (M49)                         4636 non-null   int64  
 1   Year Code                               4636 non-null   int64  
 2   Agricultural Use                        4636 non-null   float64
 3   Use per area of cropland                4636 non-null   float64
 4   Use per value of agricultural production  4636 non-null   float64
dtypes: float64(3), int64(2)
memory usage: 181.2 KB
None
```

```python
for df in dataframes:
    # Understand the data
    print(df.describe())
```

```
Element  Area Code (M49)    Year Code         Value
count        4856.000000  4856.000000  4.856000e+03
mean          424.771005  2011.526359  2.452992e+08
std           249.703399     6.917770  1.316394e+10
min             4.000000  2000.000000 -8.823557e+02
25%           212.000000  2006.000000  1.171832e+02
50%           426.000000  2012.000000  2.394935e+02
75%           638.000000  2018.000000  4.643297e+02
max           894.000000  2023.000000  8.877778e+11
Element  Area Code (M49)    Year Code          Yield
count        4587.000000  4587.000000    4587.000000
mean          429.189448  2010.976673  105164.360316
std           252.293780     6.642259   59284.147617
min             4.000000  2000.000000     535.500000
25%           208.000000  2005.000000   63748.188889
50%           422.000000  2011.000000   99118.777778
75%           643.000000  2017.000000  136174.344444
max           894.000000  2022.000000  718138.000000
Element  Area Code (M49)    Year Code  Crops total (Emissions CH4)  \
count        5130.000000  5130.000000                   5130.000000
mean          435.660039  2010.521053                    159.913930
std           252.413017     6.343028                    629.620664
min             4.000000  2000.000000                      0.000000
25%           218.000000  2005.000000                      0.240525
50%           434.000000  2011.000000                      6.857050
75%           654.000000  2016.000000                    159.913930
max           894.000000  2021.000000                   5649.183400

Element  Crops total (Emissions N2O)  Emissions (CO2)  Emissions (N2O)

count                    5130.000000      5130.000000      5130.000000

mean                        3.979807      1726.793480         0.741248

std                        13.512806      8042.168867         2.570614

min                         0.000000         0.000000         0.000000

25%                         0.065575         0.000000         0.000000

50%                         0.739450         0.000000         0.000000

75%                         3.979807       337.775600         0.093150

max                       134.165600    120512.534850        19.041600
```

```
Element  Area Code (M49)    Year Code          Value
count        4217.000000  4217.000000    4217.000000
mean          433.225516  2010.538535    4846.300277
std           254.707776     6.344924   24192.430485
min             4.000000  2000.000000       0.170000
25%           208.000000  2005.000000      67.845000
50%           426.000000  2011.000000     389.110000
75%           662.000000  2016.000000    1887.035000
max           894.000000  2022.000000  358154.430000
Element  Area Code (M49)    Year Code  Local currency units per USD
count        8639.000000  8639.000000                 8.639000e+03
mean          428.825674  2002.591619                 7.809742e+05
std           250.066687    12.448408                 7.233271e+07
min             4.000000  1980.000000                 6.452907e-04
25%           218.000000  1992.000000                 1.541914e+00
50%           426.000000  2003.000000                 7.621292e+00
75%           642.000000  2013.000000                 1.137772e+02
max           894.000000  2023.000000                 6.723052e+09
Element  Area Code (M49)    Year Code  Agricultural Use
count        1933.000000  1933.000000      1.933000e+03
mean          420.620279  2010.070874      2.159906e+05
std           253.647217     5.299190      6.553975e+05
min             4.000000  2002.000000      0.000000e+00
25%           204.000000  2006.000000      3.511733e+03
50%           410.000000  2009.000000      2.074242e+04
75%           616.000000  2014.000000      1.072280e+05
max           894.000000  2021.000000      6.388340e+06
Element  Area Code (M49)    Year Code  Export Quantity
Food  \
count        2177.000000  2177.000000      2177.000000   2177.000000

mean          425.478640  2015.539274       508.225862   2215.011361

std           250.755052     3.462834      1280.369478   9126.273787

min             4.000000  2010.000000         0.000000      0.471875

25%           204.000000  2013.000000        11.680000    115.345294

50%           418.000000  2016.000000        60.528235    384.508125

75%           642.000000  2019.000000       368.063750   1205.096471

max           894.000000  2021.000000     12759.800000  91074.883750


Element  Import Quantity       Losses  Other uses (non-food)
count        2177.000000  2177.000000            2177.000000
mean          642.038727   282.484419             575.864971
std          1595.877808  1027.437934            3416.481621
```

```
min              0.250000     0.000000               0.000000
25%             44.974118    11.875000               5.428571
50%            156.169412    46.583333              24.714286
75%            494.277059   151.692308             144.916667
max          19329.064118  8927.933333           49922.894444
Element  Area Code (M49)     Year Code        Value
count         8580.000000  8.580000e+03  8580.000000
mean           425.041375  9.630100e+06    41.101216
std            253.548887  1.004158e+07    95.322037
min              4.000000  2.000000e+03   -83.900000
25%            203.750000  2.010000e+03    16.644250
50%            418.000000  2.021000e+03    29.222750
75%            643.000000  2.009201e+07    51.210000
max            894.000000  2.020202e+07  5735.000000
Element  Area Code (M49)    Year Code  Export Value  Import Value  \
count         1933.000000  1933.000000  1.933000e+03  1.933000e+03
mean           128.575272  2006.503880  4.136485e+05  4.254417e+05
std             72.132313     9.190677  1.066353e+06  1.590308e+06
min              4.000000  1991.000000  5.666667e+00  4.985833e+02
25%             64.000000  1999.000000  3.971583e+03  1.577117e+04
50%            132.000000  2006.000000  3.191032e+04  5.873025e+04
75%            192.000000  2014.000000  1.939529e+05  2.104958e+05
max            246.000000  2022.000000  1.158613e+07  2.175788e+07

Element  Export Value Lagged
count            1.933000e+03
mean             4.139294e+05
std              1.066271e+06
min              5.666667e+00
25%              3.971583e+03
50%              3.237908e+04
75%              1.945799e+05
max              1.158613e+07
Element  Area Code (M49)    Year Code      Value US$
count         4580.000000  4580.000000    4580.000000
mean           427.732314  2011.036026    4399.514995
std            251.633840     6.621206   16476.070827
min              4.000000  2000.000000 -293503.007500
25%            208.000000  2005.000000      53.949829
50%            426.000000  2011.000000     311.059974
75%            643.000000  2017.000000    1706.588250
max            894.000000  2022.000000  329026.500000
Element  Area Code (M49)    Year Code  Standard Deviation  Temperature
change
count         5481.000000  5481.000000         5481.000000
5481.000000
mean           434.977194  2011.021346            0.505723
1.051048
std            253.999159     6.630340            0.221895
```

```
0.547298
min             4.000000  2000.000000            0.197600           -
1.305200
25%           214.000000  2005.000000            0.335000
0.689600
50%           434.000000  2011.000000            0.505723
1.017200
75%           654.000000  2017.000000            0.596800
1.349000
max           894.000000  2022.000000            1.828000
5.327200
Element  Area Code (M49)    Year Code           Area
count        9519.000000  9519.000000    9519.000000
mean          435.903561  2001.067129   19649.805821
std           252.752005    12.027339   59059.363581
min             4.000000  1980.000000       0.044000
25%           218.000000  1991.000000      81.265714
50%           434.000000  2001.000000    2442.456923
75%           654.000000  2011.500000   13044.833333
max           894.000000  2021.000000  706804.222222
Element  Area Code (M49)    Year Code  Agricultural Use  \
count        4636.000000  4636.000000       4636.000000
mean          425.071182  2010.510354       5462.949959
std           249.822875     6.342244      20184.737029
min             8.000000  2000.000000          0.002857
25%           204.000000  2005.000000         34.032143
50%           422.000000  2011.000000        291.040952
75%           634.000000  2016.000000       1819.812000
max           894.000000  2021.000000     236281.010000

Element  Use per area of cropland  Use per value of agricultural
production
count                  4636.000000
4636.000000
mean                      3.410599
0.961465
std                       4.846379
1.009887
min                       0.000000
0.000000
25%                       0.420000
0.280000
50%                       1.765000
0.800000
75%                       3.442500
1.090000
max                      37.610000
7.430000
```

```python
# Check for missing values
for df in dataframes:
    print(df.isnull().sum())
```

```
Element
Area Code (M49)    0
Year Code          0
Value              0
dtype: int64
Element
Area Code (M49)    0
Year Code          0
Yield              0
dtype: int64
Element
Area Code (M49)                  0
Year Code                        0
Crops total (Emissions CH4)      0
Crops total (Emissions N2O)      0
Emissions (CO2)                  0
Emissions (N2O)                  0
dtype: int64
Element
Area Code (M49)    0
Year Code          0
Value              0
dtype: int64
Element
Area Code (M49)                   0
Year Code                         0
Local currency units per USD      0
dtype: int64
Element
Area Code (M49)     0
Year Code           0
Agricultural Use    0
dtype: int64
Element
Area Code (M49)           0
Year Code                 0
Export Quantity           0
Food                      0
Import Quantity           0
Losses                    0
Other uses (non-food)     0
dtype: int64
Element
Area Code (M49)    0
Year Code          0
Value              0
```

```
dtype: int64
Element
Area Code (M49)         0
Year Code               0
Export Value            0
Import Value            0
Export Value Lagged     0
dtype: int64
Element
Area Code (M49)    0
Year Code          0
Value US$          0
dtype: int64
Element
Area Code (M49)        0
Year Code              0
Standard Deviation     0
Temperature change     0
dtype: int64
Element
Area Code (M49)    0
Year Code          0
Area               0
dtype: int64
Element
Area Code (M49)                                  0
Year Code                                        0
Agricultural Use                                 0
Use per area of cropland                         0
Use per value of agricultural production         0
dtype: int64

EDITTTTTTTTTTTTTT

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[394], line 1
----> 1 EDITTTTTTTTTTTTTT

NameError: name 'EDITTTTTTTTTTTTTT' is not defined
```

```
import numpy as np

# Create an empty list to store the new DataFrames
new_dataframes = []
```

```
# Analyse relationship between variables
for df in dataframes:
    # Calculate the absolute correlation matrix
    corr_matrix = df.corr().abs()

    # Select upper triangle of correlation matrix
    upper = corr_matrix.where(np.triu(np.ones(corr_matrix.shape),
k=1).astype(bool))

    # List the feature pairs with correlation greater than 0.7
    to_drop = [column for column in upper.columns if any(upper[column]
> 0.7)]

    # Drop the highly correlated features
    df_new = df.drop(to_drop, axis=1)

    # Add the new DataFrame to the list
    new_dataframes.append(df_new)

# Now, new_dataframes list contains all the modified DataFrames
for df in new_dataframes:
    print(df)
```

```
Element  Area Code (M49)  Year Code         Value
0                      4       2000     26.629848
1                      4       2001    653.981386
2                      4       2002    930.398250
3                      4       2003    725.213777
4                      4       2004    726.528864
...                  ...        ...           ...
4851                 894       2019    596.243903
4852                 894       2020    899.170700
4853                 894       2021   1502.023896
4854                 894       2022    787.094206
4855                 894       2023    415.040265

[4856 rows x 3 columns]
Element  Area Code (M49)  Year Code          Yield
0                      4       2000   60177.909091
1                      4       2001   60701.272727
2                      4       2002   61135.363636
3                      4       2003   61209.181818
4                      4       2004   61449.454545
...                  ...        ...            ...
4582                 894       2018  148768.200000
4583                 894       2019  151648.900000
4584                 894       2020  147976.600000
4585                 894       2021  148215.800000
4586                 894       2022  136029.900000
```

```
[4587 rows x 3 columns]
Element  Area Code (M49)  Year Code  Crops total (Emissions CH4)  \
0                      4       2000                      20.8471
1                      4       2001                      19.2605
2                      4       2002                      21.2553
3                      4       2003                      23.7017
4                      4       2004                      30.3089
...                  ...        ...                          ...
5125                 894       2017                       6.0887
5126                 894       2018                       5.1998
5127                 894       2019                       4.1332
5128                 894       2020                       5.4800
5129                 894       2021                       7.0885

Element  Emissions (CO2)
0                0.00000
1                0.00000
2                0.00000
3                0.00000
4                0.00000
...                  ...
5125          7228.65865
5126          7232.02850
5127          7277.36255
5128          7283.33290
5129          7283.33290

[5130 rows x 4 columns]
Element  Area Code (M49)  Year Code     Value
0                      4       2000  2765.950
1                      4       2001  2805.540
2                      4       2002  2897.510
3                      4       2003  3093.270
4                      4       2004  3212.460
...                  ...        ...       ...
4212                 894       2017  1517.785
4213                 894       2018  1656.255
4214                 894       2019  1673.465
4215                 894       2020  1818.285
4216                 894       2021  1859.005

[4217 rows x 3 columns]
Element  Area Code (M49)  Year Code  Local currency units per USD
0                      4       1980                     44.129167
1                      4       1981                     49.479902
2                      4       1982                     50.599608
3                      4       1983                     50.599608
4                      4       1984                     50.599606
...                  ...        ...                          ...
8634                 894       2019                     12.890000
```

```
8635                  894       2020                      18.344093
8636                  894       2021                      20.018487
8637                  894       2022                      16.937594
8638                  894       2023                      19.799163

[8639 rows x 3 columns]
Element  Area Code (M49)  Year Code  Agricultural Use
0                      4       2002         17900.000
1                      4       2003         33200.000
2                      4       2004         45000.000
3                      4       2005         20577.000
4                      4       2006         68253.000
...                  ...        ...               ...
1928                 894       2016        136190.000
1929                 894       2017        180547.500
1930                 894       2018        130232.925
1931                 894       2019        184999.795
1932                 894       2021        194259.350

[1933 rows x 3 columns]
Element  Area Code (M49)  Year Code  Export Quantity        Food
0                      4       2010        32.727273  635.203750
1                      4       2011        25.181818  644.437500
2                      4       2012        18.000000  685.173750
3                      4       2013        25.545455  712.472500
4                      4       2014        25.750000  866.460000
...                  ...        ...              ...         ...
2172                 894       2017        54.221765  416.381765
2173                 894       2018        34.518235  471.152500
2174                 894       2019        37.767647  529.385000
2175                 894       2020        48.628125  543.072500
2176                 894       2021        67.708824  539.760000

[2177 rows x 4 columns]
Element  Area Code (M49)  Year Code       Value
0                      4       2000   30.420000
1                      4       2001   32.966667
2                      4       2002   31.340000
3                      4       2003   31.500000
4                      4       2004   25.125000
...                  ...        ...         ...
8575                 894   20162018   31.160000
8576                 894   20172019   32.180000
8577                 894   20182020   32.280000
8578                 894   20192021   35.733333
8579                 894   20202022  101.000000

[8580 rows x 3 columns]
Element  Area Code (M49)  Year Code   Export Value   Import Value
0                      4       1991   19648.600000   10859.416667
```

```
1                       4       1992    8422.400000   11130.416667
2                       4       1993    8912.800000   11349.666667
3                       4       1994   10071.400000    9683.083333
4                       4       1995    9919.200000   17950.916667
...                   ...        ...             ...            ...
1928                  246       2007  144449.500000  346656.250000
1929                  246       2008  167628.750000  421922.250000
1930                  246       2009  131014.083333  373540.583333
1931                  246       2010  142707.000000  384444.750000
1932                  246       2011  179557.916667  469686.500000

[1933 rows x 4 columns]
Element  Area Code (M49)  Year Code   Value US$
0                       4       2000    0.170000
1                       4       2001    0.680000
2                       4       2002   50.000000
3                       4       2003   29.400000
4                       4       2004   93.100000
...                   ...        ...         ...
4575                  894       2018  117.007475
4576                  894       2019  495.333333
4577                  894       2020  113.466667
4578                  894       2021  -92.100000
4579                  894       2022 -135.934725

[4580 rows x 3 columns]
Element  Area Code (M49)  Year Code  Standard Deviation  Temperature
change
0                       4       2000              0.8326
0.9930
1                       4       2001              0.8326
1.3110
2                       4       2002              0.8326
1.3650
3                       4       2003              0.8326
0.5870
4                       4       2004              0.8326
1.3732
...                   ...        ...                 ...
...
5476                  894       2018              0.3636
0.6482
5477                  894       2019              0.3636
0.8548
5478                  894       2020              0.3636
0.8912
5479                  894       2021              0.3636
0.8218
5480                  894       2022              0.3636
```

```
0.6864

[5481 rows x 4 columns]
Element  Area Code (M49)  Year Code          Area
0                      4       1980  28356.666667
1                      4       1981  28360.111111
2                      4       1982  28362.222222
3                      4       1983  28363.888889
4                      4       1984  28367.333333
...                  ...        ...           ...
9514                 894       2017  19075.999992
9515                 894       2018  19076.000000
9516                 894       2019  19076.000000
9517                 894       2020  19076.000000
9518                 894       2021  19076.000000

[9519 rows x 3 columns]
Element  Area Code (M49)  Year Code  Agricultural Use  \
0                      8       2000         86.842857
1                      8       2001         89.870000
2                      8       2002         92.895714
3                      8       2003         95.920000
4                      8       2004         98.945714
...                  ...        ...               ...
4631                 894       2017       1739.112000
4632                 894       2018       1678.656000
4633                 894       2019       1678.656000
4634                 894       2020       1678.656000
4635                 894       2021       1678.656000

Element  Use per area of cropland  Use per value of agricultural
production
0                             0.44
0.23
1                             0.46
0.23
2                             0.47
0.24
3                             0.49
0.24
4                             0.51
0.23
...                            ...
...
4631                          1.13
1.16
4632                          1.09
1.18
4633                          1.09
1.23
```

```
4634                            1.09
1.14
4635                            1.09
1.09

[4636 rows x 5 columns]
```

```
# Initialize an empty list
df_list = []

# Now, new_dataframes list contains all the modified DataFrames
for df in new_dataframes:
    df_list.append(df)

# Now, df_list contains all the dataframes
print(df_list)
```

```
[Element  Area Code (M49)  Year Code        Value
0                       4       2000    26.629848
1                       4       2001   653.981386
2                       4       2002   930.398250
3                       4       2003   725.213777
4                       4       2004   726.528864
...                   ...        ...          ...
4851                  894       2019   596.243903
4852                  894       2020   899.170700
4853                  894       2021  1502.023896
4854                  894       2022   787.094206
4855                  894       2023   415.040265

[4856 rows x 3 columns], Element  Area Code (M49)  Year Code
Yield
0                       4       2000   60177.909091
1                       4       2001   60701.272727
2                       4       2002   61135.363636
3                       4       2003   61209.181818
4                       4       2004   61449.454545
...                   ...        ...            ...
4582                  894       2018  148768.200000
4583                  894       2019  151648.900000
4584                  894       2020  147976.600000
4585                  894       2021  148215.800000
4586                  894       2022  136029.900000

[4587 rows x 3 columns], Element  Area Code (M49)  Year Code  Crops
total (Emissions CH4)  \
0                       4       2000                       20.8471
1                       4       2001                       19.2605
2                       4       2002                       21.2553
3                       4       2003                       23.7017
```

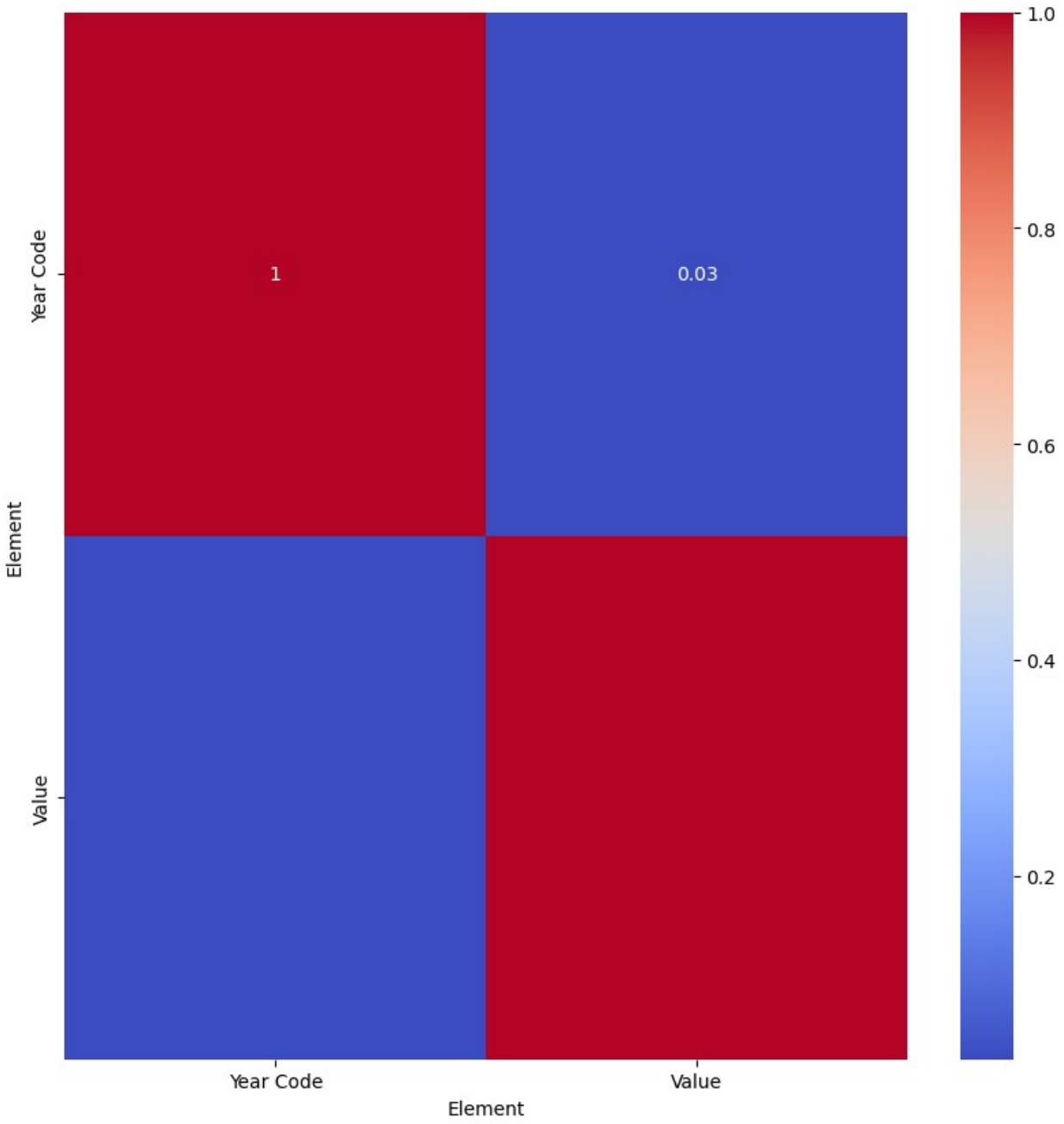
```
4                       4       2004                        30.3089
...                   ...        ...                           ...
5125                  894       2017                         6.0887
5126                  894       2018                         5.1998
5127                  894       2019                         4.1332
5128                  894       2020                         5.4800
5129                  894       2021                         7.0885

Element  Emissions (CO2)
0                0.00000
1                0.00000
2                0.00000
3                0.00000
4                0.00000
...                  ...
5125          7228.65865
5126          7232.02850
5127          7277.36255
5128          7283.33290
5129          7283.33290

[5130 rows x 4 columns], Element  Area Code (M49)  Year Code     Value
0                       4       2000  2765.950
1                       4       2001  2805.540
2                       4       2002  2897.510
3                       4       2003  3093.270
4                       4       2004  3212.460
...                   ...        ...       ...
4212                  894       2017  1517.785
4213                  894       2018  1656.255
4214                  894       2019  1673.465
4215                  894       2020  1818.285
4216                  894       2021  1859.005

[4217 rows x 3 columns], Element  Area Code (M49)  Year Code  Local
currency units per USD
0                       4       1980                       44.129167
1                       4       1981                       49.479902
2                       4       1982                       50.599608
3                       4       1983                       50.599608
4                       4       1984                       50.599606
...                   ...        ...                             ...
8634                  894       2019                       12.890000
8635                  894       2020                       18.344093
8636                  894       2021                       20.018487
8637                  894       2022                       16.937594
8638                  894       2023                       19.799163

[8639 rows x 3 columns], Element  Area Code (M49)  Year Code
Agricultural Use
```

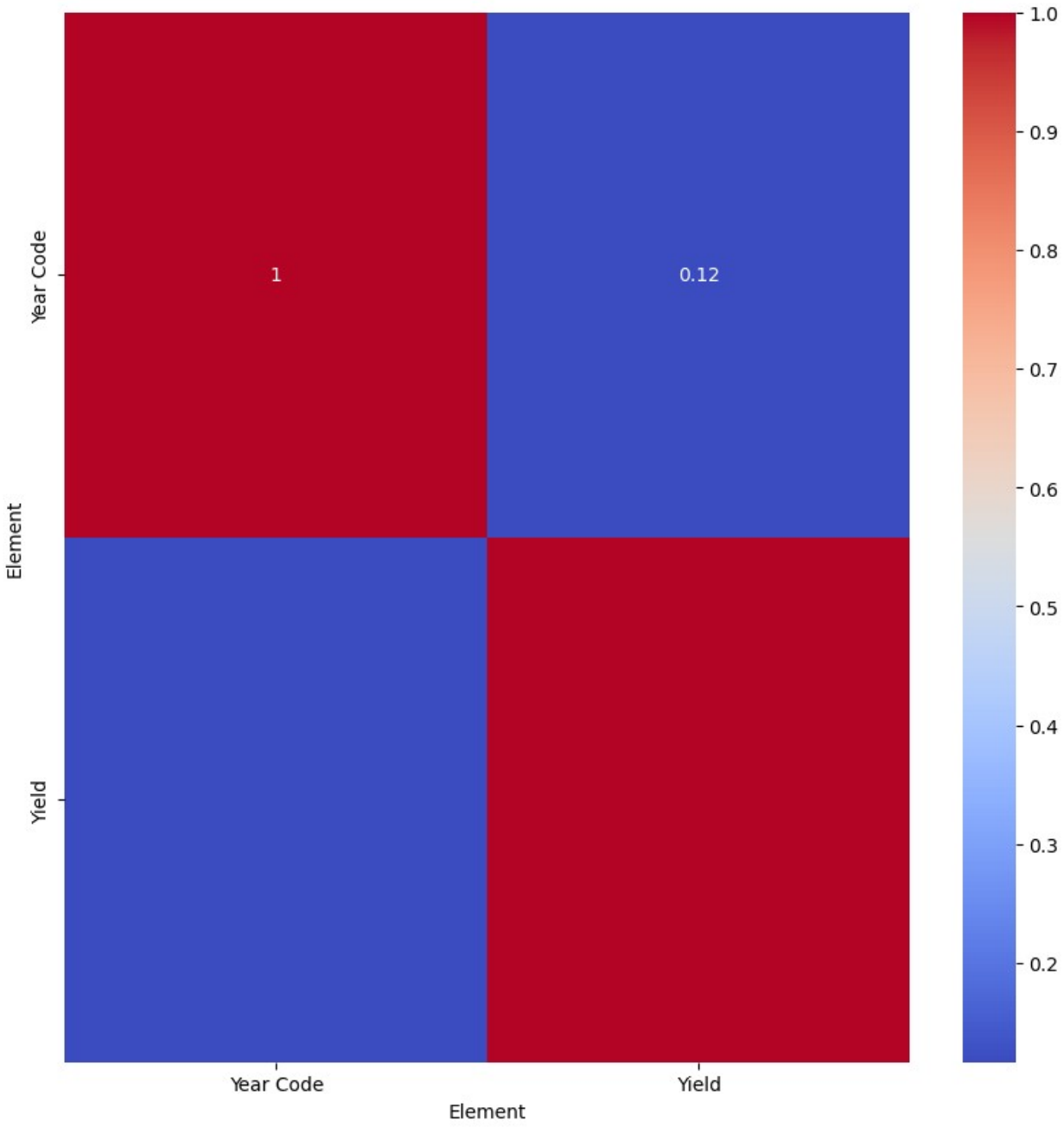
```
0                      4       2002         17900.000
1                      4       2003         33200.000
2                      4       2004         45000.000
3                      4       2005         20577.000
4                      4       2006         68253.000
...                  ...        ...               ...
1928                 894       2016        136190.000
1929                 894       2017        180547.500
1930                 894       2018        130232.925
1931                 894       2019        184999.795
1932                 894       2021        194259.350

[1933 rows x 3 columns], Element  Area Code (M49)  Year Code  Export
Quantity        Food
0                      4       2010        32.727273  635.203750
1                      4       2011        25.181818  644.437500
2                      4       2012        18.000000  685.173750
3                      4       2013        25.545455  712.472500
4                      4       2014        25.750000  866.460000
...                  ...        ...              ...         ...
2172                 894       2017        54.221765  416.381765
2173                 894       2018        34.518235  471.152500
2174                 894       2019        37.767647  529.385000
2175                 894       2020        48.628125  543.072500
2176                 894       2021        67.708824  539.760000

[2177 rows x 4 columns], Element  Area Code (M49)  Year Code
Value
0                      4       2000   30.420000
1                      4       2001   32.966667
2                      4       2002   31.340000
3                      4       2003   31.500000
4                      4       2004   25.125000
...                  ...        ...         ...
8575                 894   20162018   31.160000
8576                 894   20172019   32.180000
8577                 894   20182020   32.280000
8578                 894   20192021   35.733333
8579                 894   20202022  101.000000

[8580 rows x 3 columns], Element  Area Code (M49)  Year Code   Export
Value   Import Value
0                      4       1991   19648.600000   10859.416667
1                      4       1992    8422.400000   11130.416667
2                      4       1993    8912.800000   11349.666667
3                      4       1994   10071.400000    9683.083333
4                      4       1995    9919.200000   17950.916667
...                  ...        ...             ...             ...
1928                 246       2007  144449.500000  346656.250000
1929                 246       2008  167628.750000  421922.250000
```

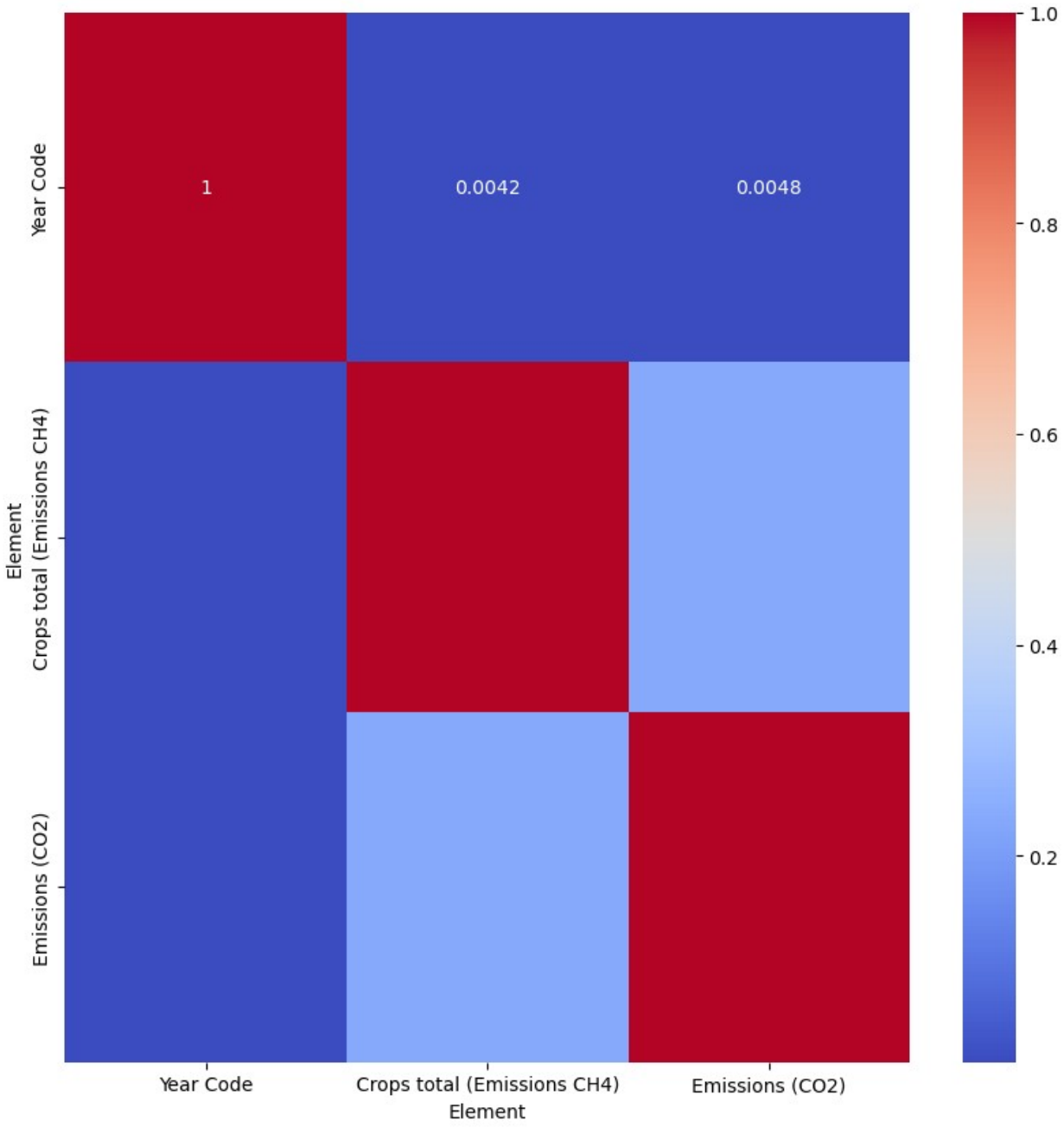
```
1930                  246       2009  131014.083333  373540.583333
1931                  246       2010  142707.000000  384444.750000
1932                  246       2011  179557.916667  469686.500000

[1933 rows x 4 columns], Element  Area Code (M49)  Year Code   Value
US$
0                       4       2000    0.170000
1                       4       2001    0.680000
2                       4       2002   50.000000
3                       4       2003   29.400000
4                       4       2004   93.100000
...                   ...        ...         ...
4575                  894       2018  117.007475
4576                  894       2019  495.333333
4577                  894       2020  113.466667
4578                  894       2021  -92.100000
4579                  894       2022 -135.934725

[4580 rows x 3 columns], Element  Area Code (M49)  Year Code  Standard
Deviation  Temperature change
0                       4       2000              0.8326
0.9930
1                       4       2001              0.8326
1.3110
2                       4       2002              0.8326
1.3650
3                       4       2003              0.8326
0.5870
4                       4       2004              0.8326
1.3732
...                   ...        ...                 ...
...
5476                  894       2018              0.3636
0.6482
5477                  894       2019              0.3636
0.8548
5478                  894       2020              0.3636
0.8912
5479                  894       2021              0.3636
0.8218
5480                  894       2022              0.3636
0.6864

[5481 rows x 4 columns], Element  Area Code (M49)  Year Code
Area
0                       4       1980  28356.666667
1                       4       1981  28360.111111
2                       4       1982  28362.222222
3                       4       1983  28363.888889
4                       4       1984  28367.333333
```

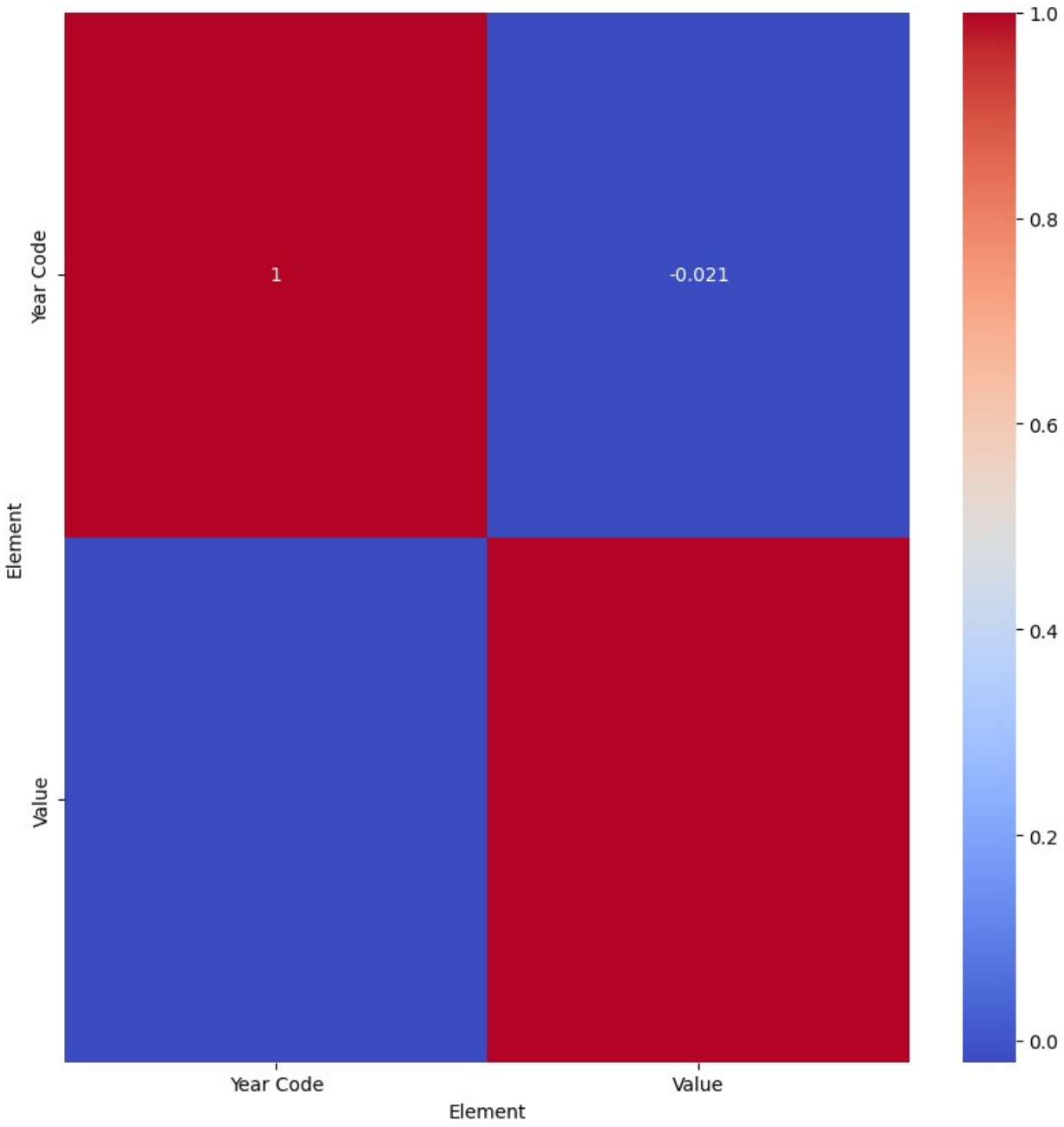
```
...                   ...         ...            ...
9514                  894        2017  19075.999992
9515                  894        2018  19076.000000
9516                  894        2019  19076.000000
9517                  894        2020  19076.000000
9518                  894        2021  19076.000000

[9519 rows x 3 columns], Element  Area Code (M49)  Year Code
Agricultural Use  \
0                       8        2000         86.842857
1                       8        2001         89.870000
2                       8        2002         92.895714
3                       8        2003         95.920000
4                       8        2004         98.945714
...                   ...         ...               ...
4631                  894        2017       1739.112000
4632                  894        2018       1678.656000
4633                  894        2019       1678.656000
4634                  894        2020       1678.656000
4635                  894        2021       1678.656000

Element  Use per area of cropland  Use per value of agricultural
production
0                              0.44
0.23
1                              0.46
0.23
2                              0.47
0.24
3                              0.49
0.24
4                              0.51
0.23
...                             ...
...
4631                           1.13
1.16
4632                           1.09
1.18
4633                           1.09
1.23
4634                           1.09
1.14
4635                           1.09
1.09

[4636 rows x 5 columns]]

import seaborn as sns
from sklearn.linear_model import Lasso
```

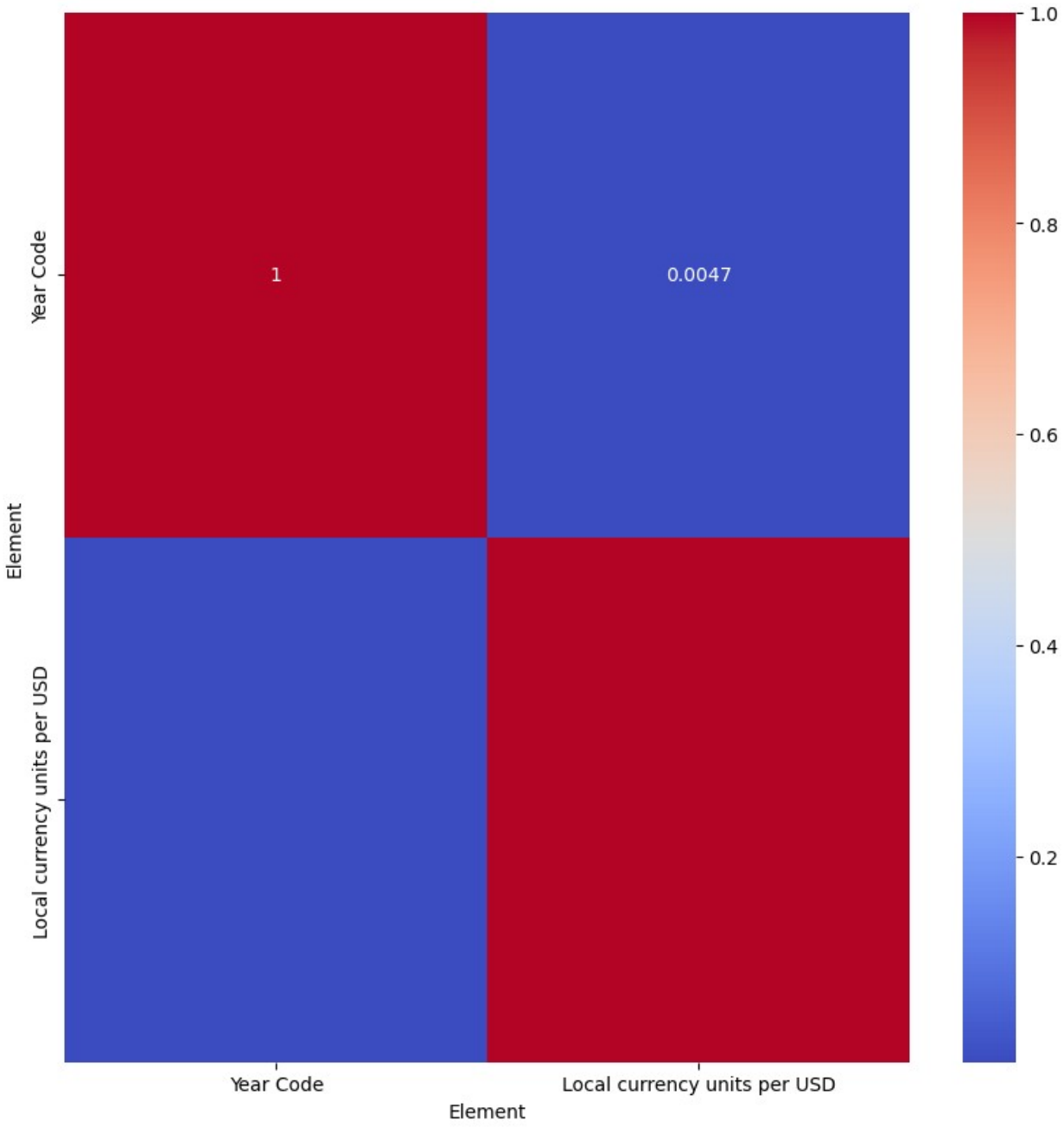
```
# Initialize an empty list to store all selected features
all_selected_features = []

# Perform Lasso regression analysis for each DataFrame
for df in new_dataframes:
    # Check if DataFrame has at least two columns to perform
regression
    if len(df.columns) >= 2:
        # Use the first column as dependent variable and the rest as
independent variables
        X = df[df.columns[1:]]
        y = df[df.columns[0]]

        # Create a correlation heatmap for the DataFrame
        plt.figure(figsize=(10,10))
        sns.heatmap(X.corr(), annot=True, cmap='coolwarm')
        plt.show()

        # Create a Lasso regression model
        model = Lasso(alpha=0.1)  # You can adjust the alpha parameter
as needed

        # Fit the model
        model.fit(X, y)

        # Get the feature names
        feature_names = np.array(X.columns)

        # Get the selected features
        selected_features = feature_names[model.coef_ != 0]

        # Remove 'Export Value' from selected features if it exists
        selected_features = [feature for feature in selected_features
if feature != 'Export Value']


        # Add the selected features to the list of all selected
features
        all_selected_features.extend(selected_features)

# Convert the list to a set to remove duplicates
unique_selected_features = set(all_selected_features)

# Convert the set back to a list (optional)
unique_selected_features = list(unique_selected_features)

print("Unique selected features:")
print(unique_selected_features)
```

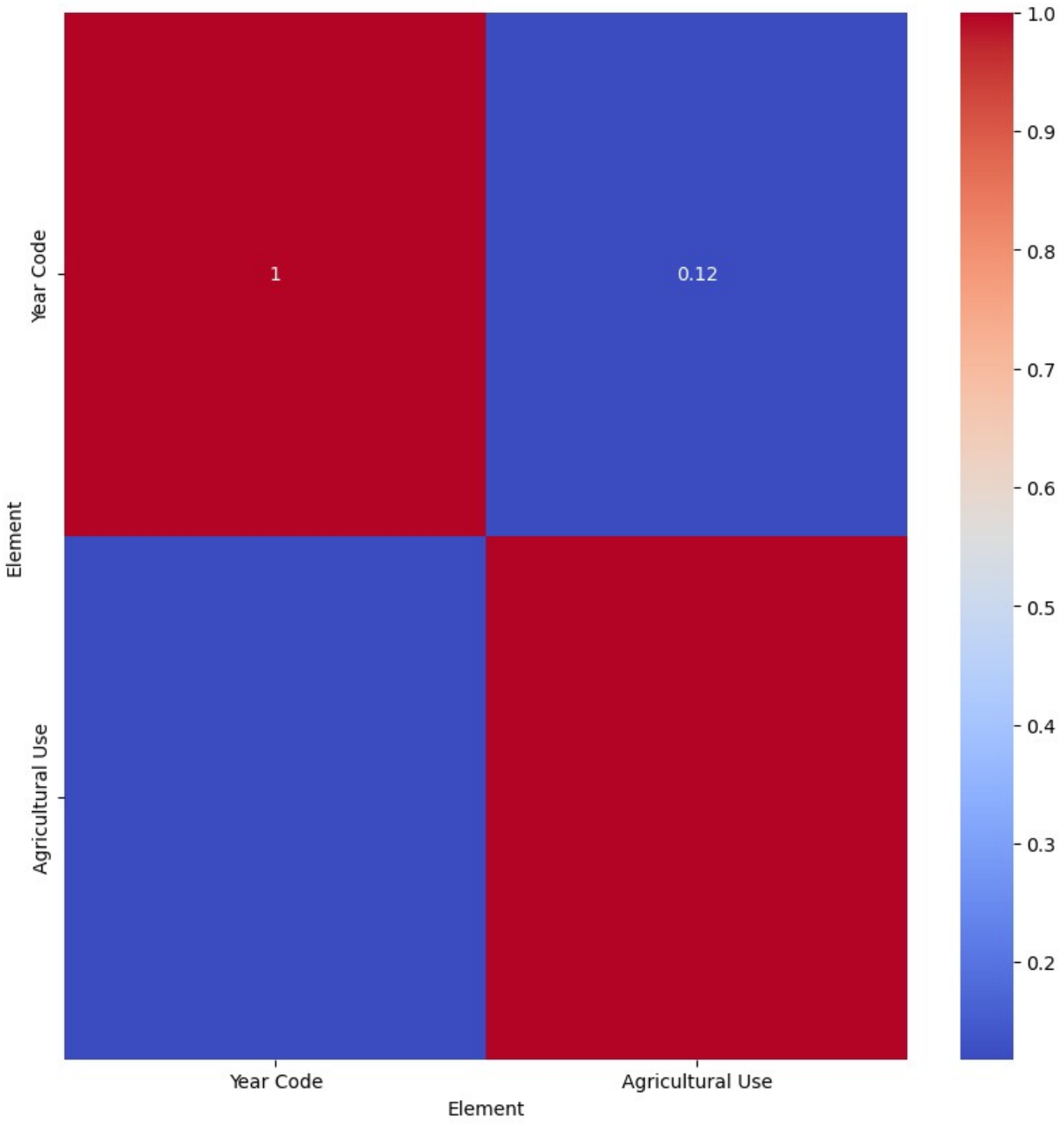

Year Code
Value
Element
1
0.03
Year Code
Value
Element
1.0
0.8
0.6
0.4
0.2

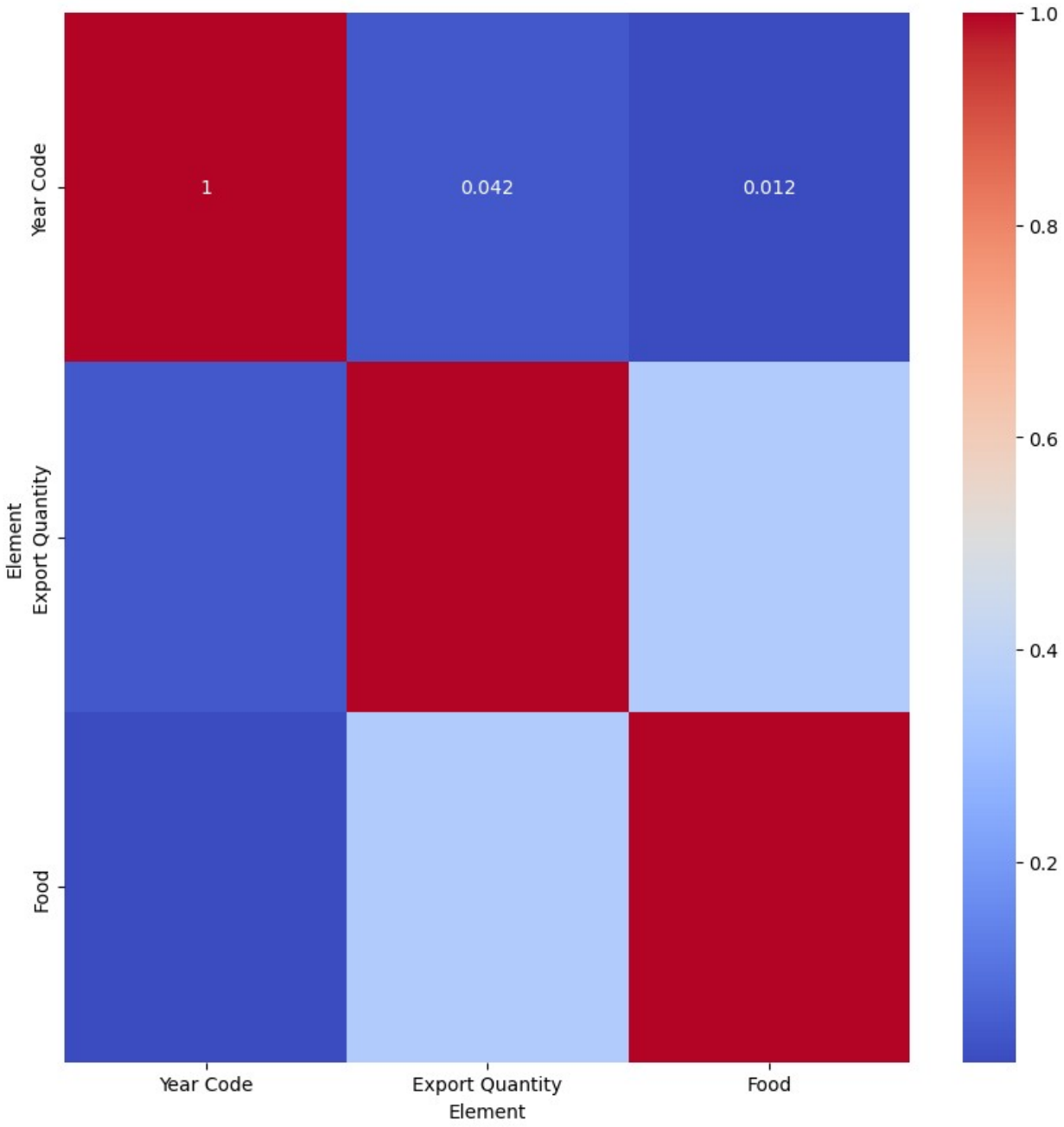

Year Code
Yield
Element
1
0.12
Year Code
Yield
Element
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2

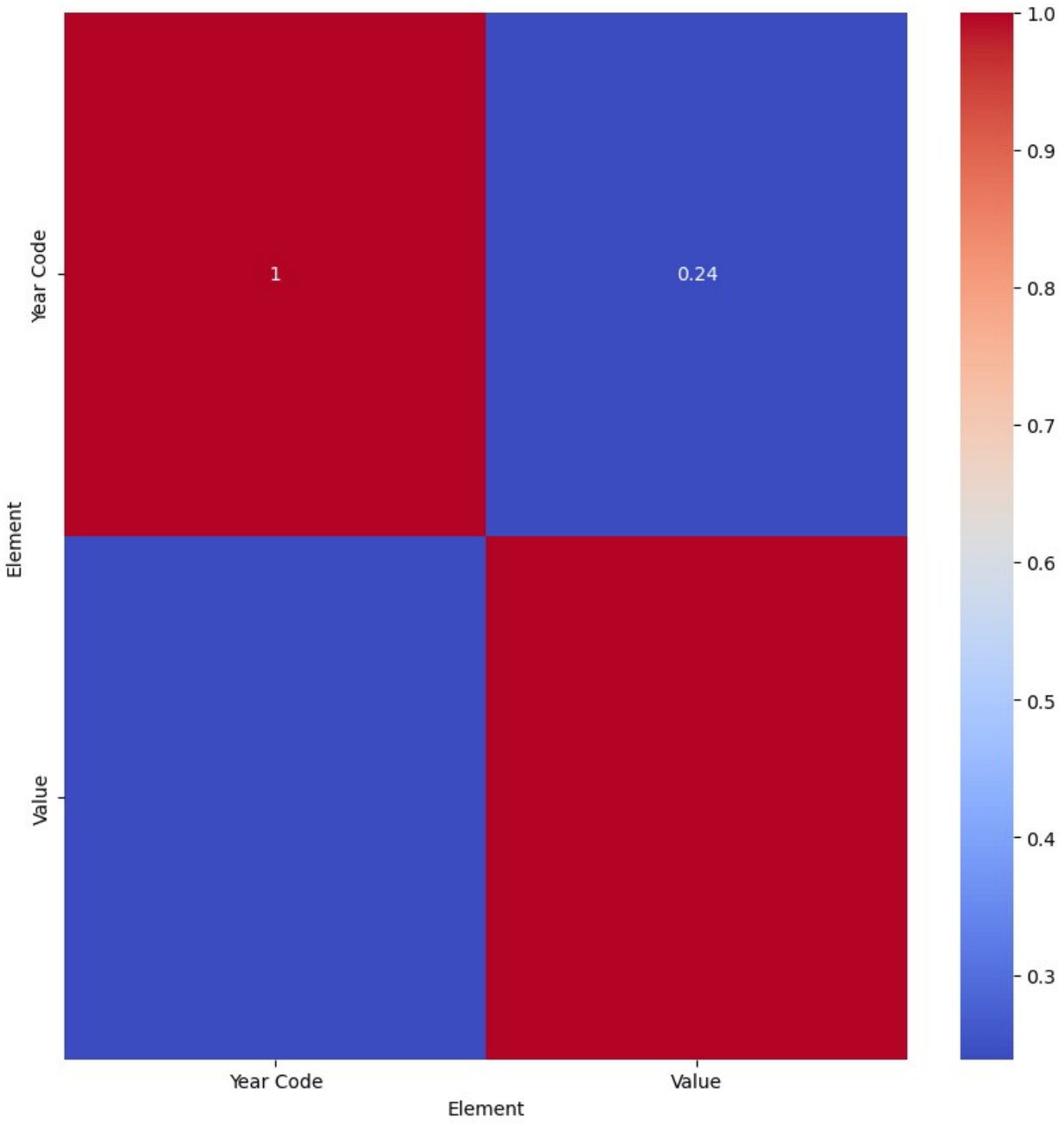

1
0.0042
0.0048
Year Code
Crops total (Emissions CH4)
Emissions (CO2)
Element
Year Code
Crops total (Emissions CH4)
Emissions (CO2)
Element
1.0
0.8
0.6
0.4
0.2

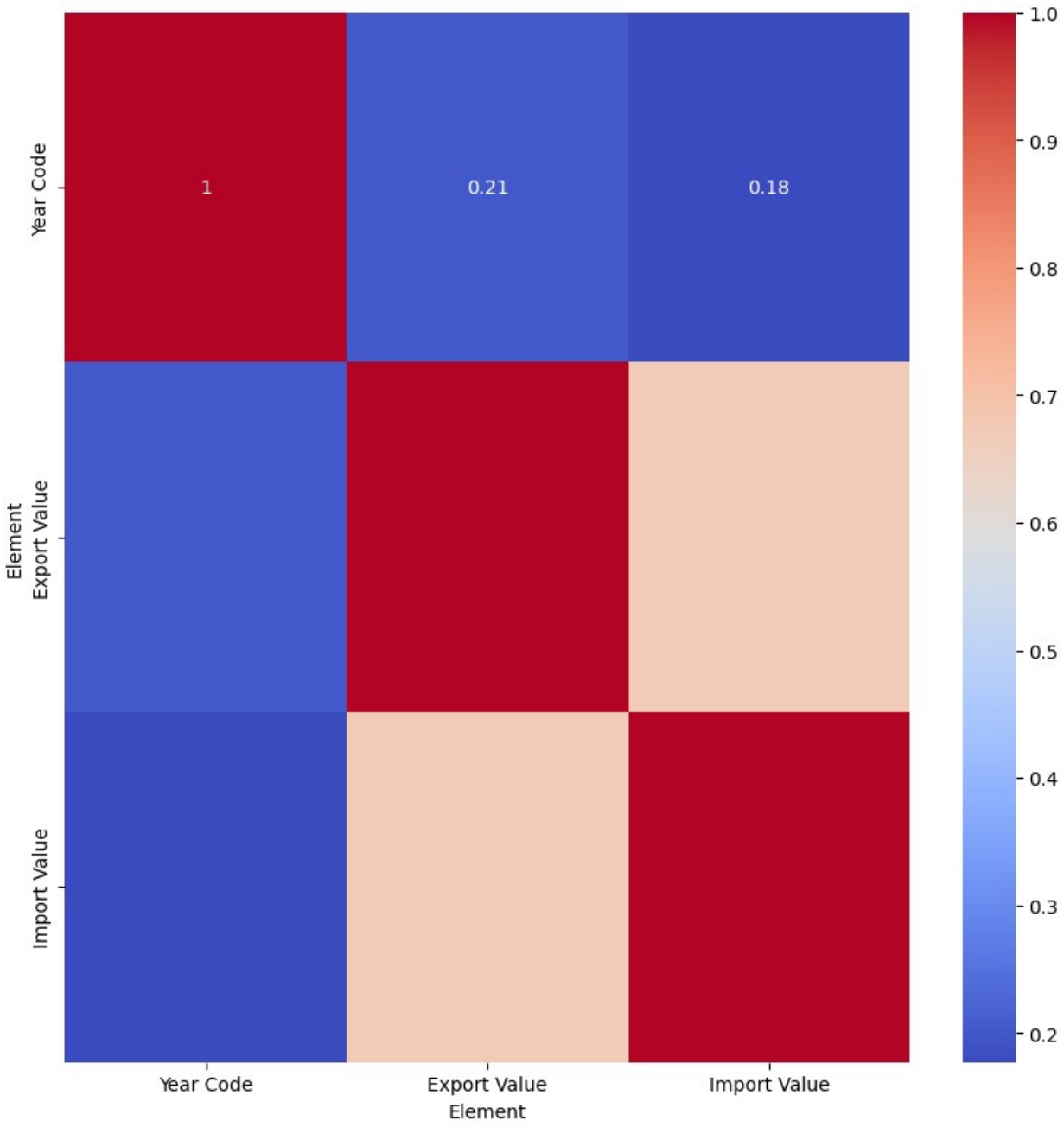

Year Code
Value
Element
1
-0.021
Year Code
Value
Element
1.0
0.8
0.6
0.4
0.2
0.0

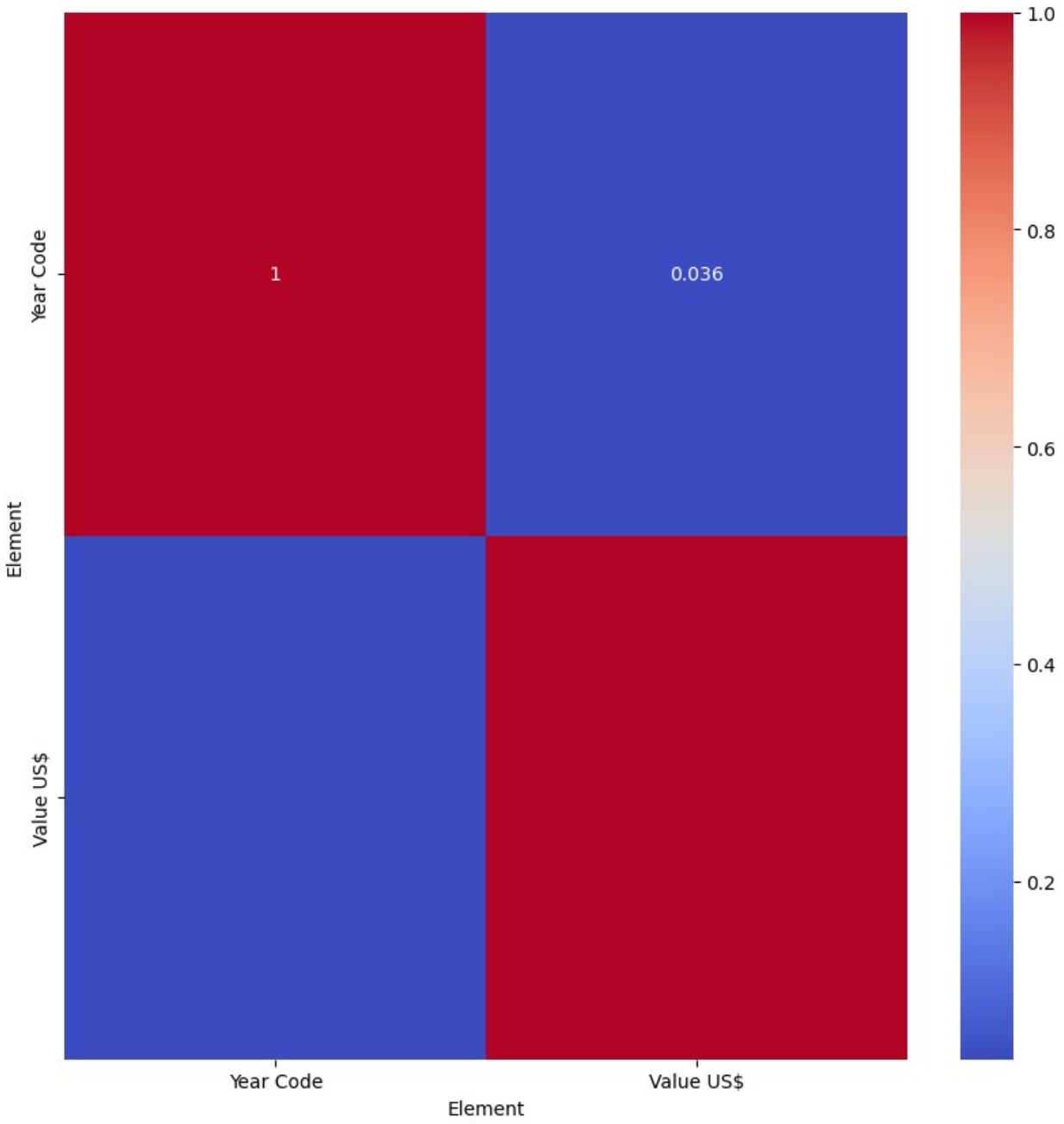

1
0.0047
Year Code
Local currency units per USD
Element
Year Code
Local currency units per USD
Element
1.0
0.8
0.6
0.4
0.2

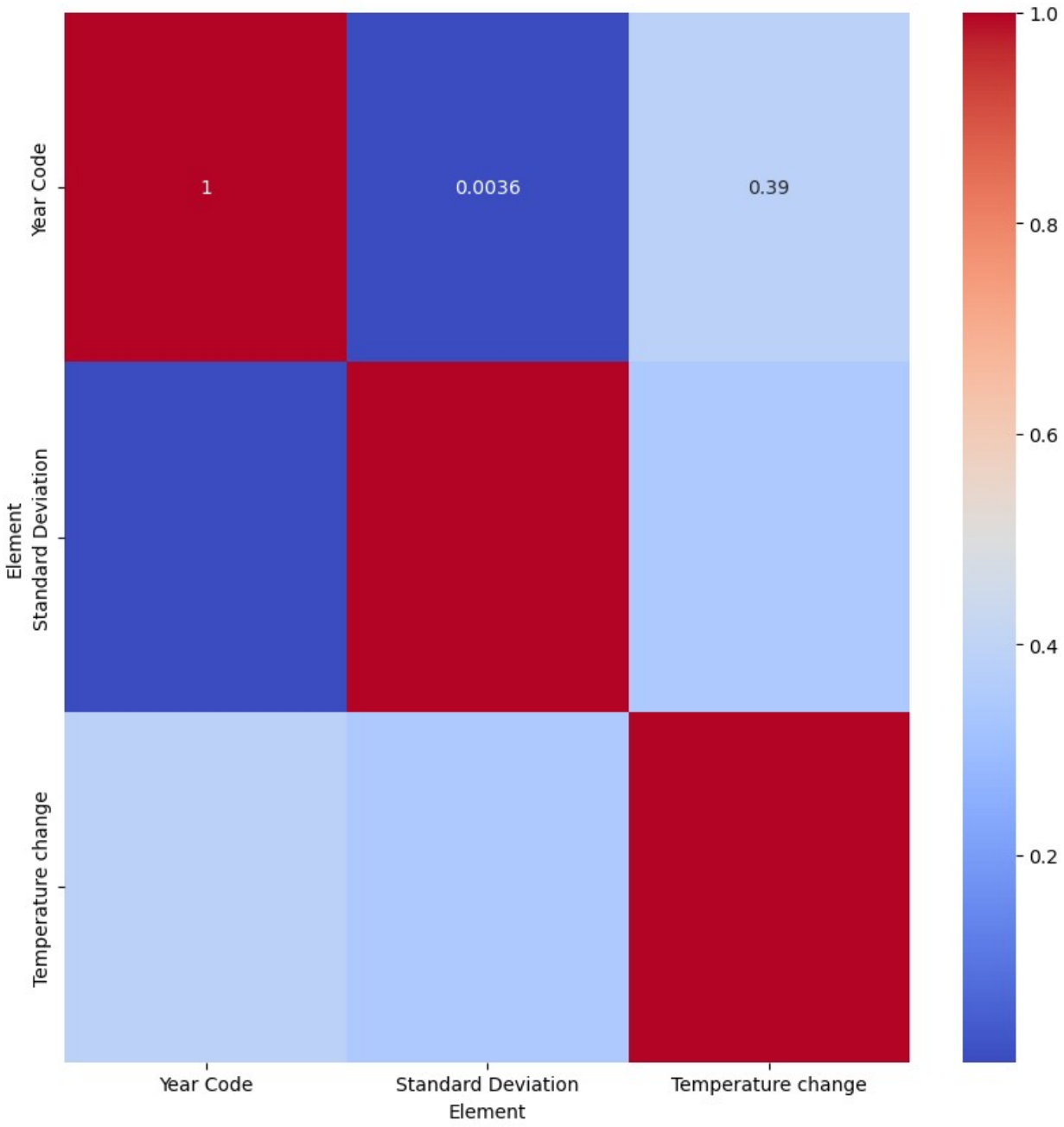

1
0.12
Year Code
Agricultural Use
Element
Year Code
Agricultural Use
Element
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2

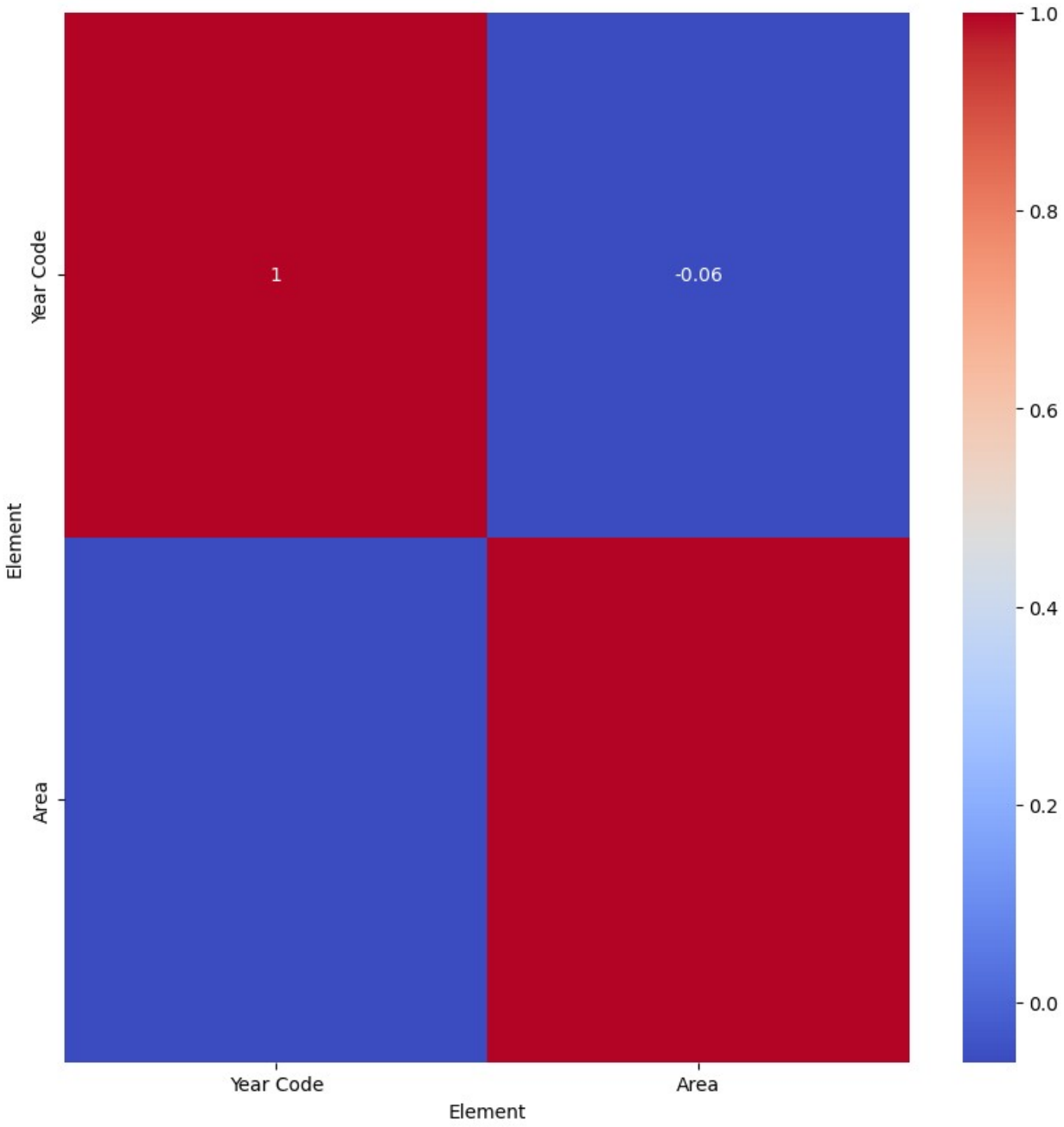

Year Code
Export Quantity
Food
Element
1
0.042
0.012
Year Code
Export Quantity
Food
Element
1.0
0.8
0.6
0.4
0.2

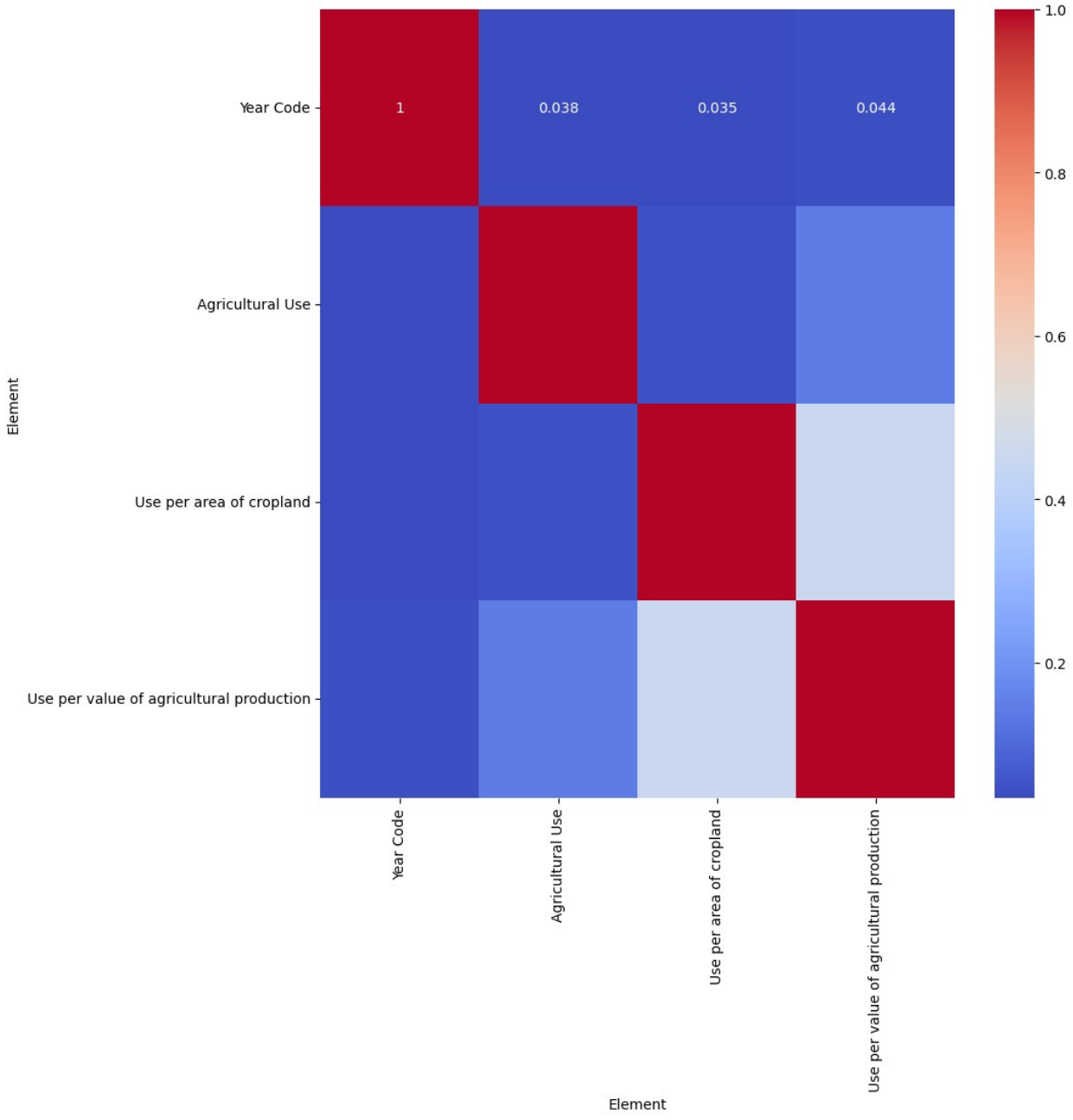

1
0.24
Year Code
Value
Element
Year Code
Value
Element
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3


1
0.21
0.18
Year Code
Element
Export Value
Import Value
Year Code
Export Value
Import Value
Element
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2


1
0.036
Year Code
Value US$
Element
Year Code
Value US$
Element
1.0
0.8
0.6
0.4
0.2


1
0.0036
0.39
Year Code
Standard Deviation
Temperature change
Element
Year Code
Standard Deviation
Temperature change
Element
1.0
0.8
0.6
0.4
0.2


Year Code
Area
Element
1
-0.06
Year Code
Area
Element
1.0
0.8
0.6
0.4
0.2
0.0




```
Unique selected features:
['Standard Deviation', 'Area', 'Use per value of agricultural
production', 'Value', 'Agricultural Use', 'Local currency units per
USD', 'Year Code', 'Use per area of cropland', 'Value US$', 'Export
Quantity', 'Yield', 'Emissions (CO2)', 'Crops total (Emissions CH4)',
'Food', 'Import Value']

from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error,
r2_score
```

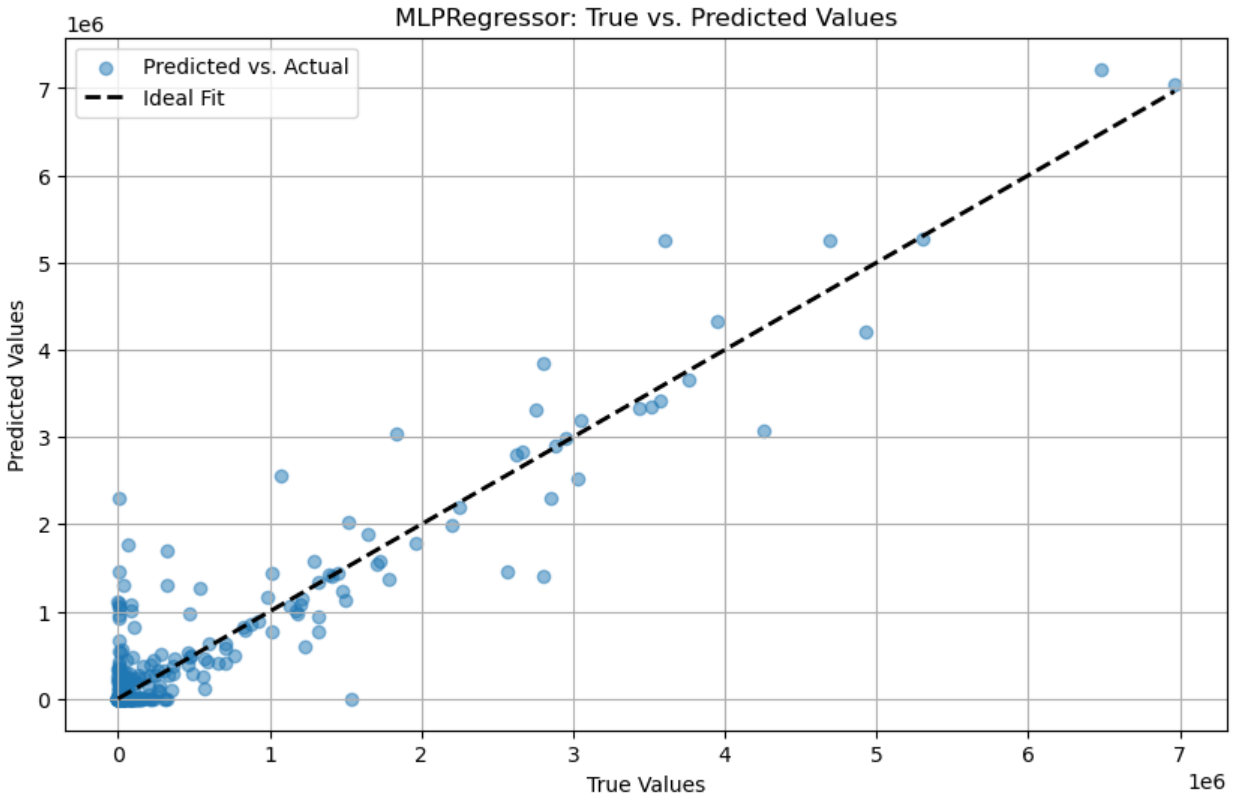
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt

# Assuming that 'Area Code (M49)' and 'Year Code' are the common
columns in all dataframes
common_columns = ['Area Code (M49)', 'Year Code']

# Merge all dataframes on 'Area Code (M49)' and 'Year Code'
merged_df = pd.concat(dataframes, axis=1, join='inner')

# Select the unique_selected_features from merged_df for X
X = merged_df[unique_selected_features]

# Set y as 'Export Value' from merged_df
y = merged_df['Export Value']

# Create a DataFrame that includes both features and target
data = X.copy()
data['Target'] = y

# Split the data into training and testing sets
train_data, test_data = train_test_split(data, test_size=0.2,
random_state=42)

# Separate the features and target in training set
X_train = train_data.drop('Target', axis=1)
y_train = train_data['Target']

# Separate the features and target in testing set
X_test = test_data.drop('Target', axis=1)
y_test = test_data['Target']

# Now, X_test and y_test should have the same length
assert len(X_test) == len(y_test)

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Create a MLPRegressor model
model = MLPRegressor(hidden_layer_sizes=(64, 32, 16), max_iter=10000,
activation='relu', solver='adam', random_state=42)

# Fit the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)
```

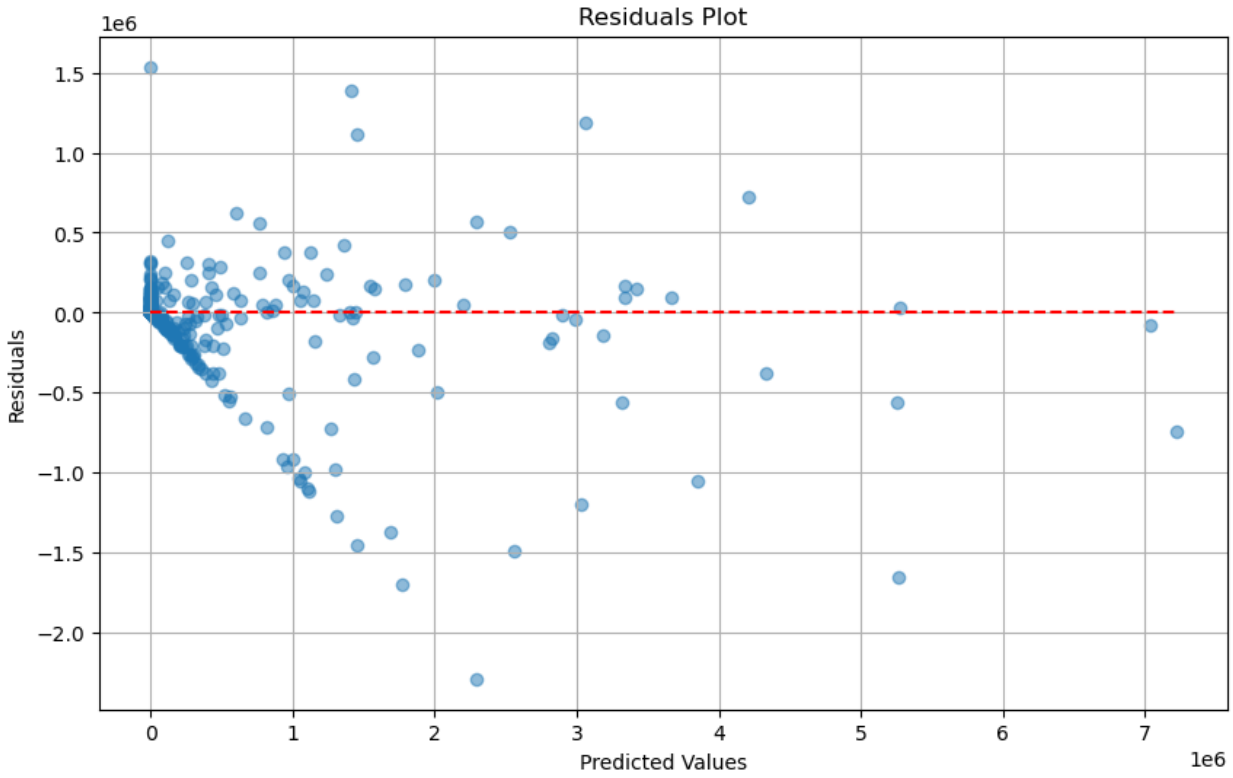
```
# Calculate performance metrics
mse = mean_squared_error(y_test, y_pred)
normalized_mean_squared_err = mse / np.var(y_test)
mean_abs_err = mean_absolute_error(y_test, y_pred)
normalized_mean_abs_err = mean_abs_err / (np.max(y_test) -
np.min(y_test))
r2_score_value = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'Normalized Mean Squared Error: {normalized_mean_squared_err}')
print(f'Mean Absolute Error: {mean_abs_err}')
print(f'Normalized Mean Absolute Error: {normalized_mean_abs_err}')
print(f'R^2 Score: {r2_score_value}')
```

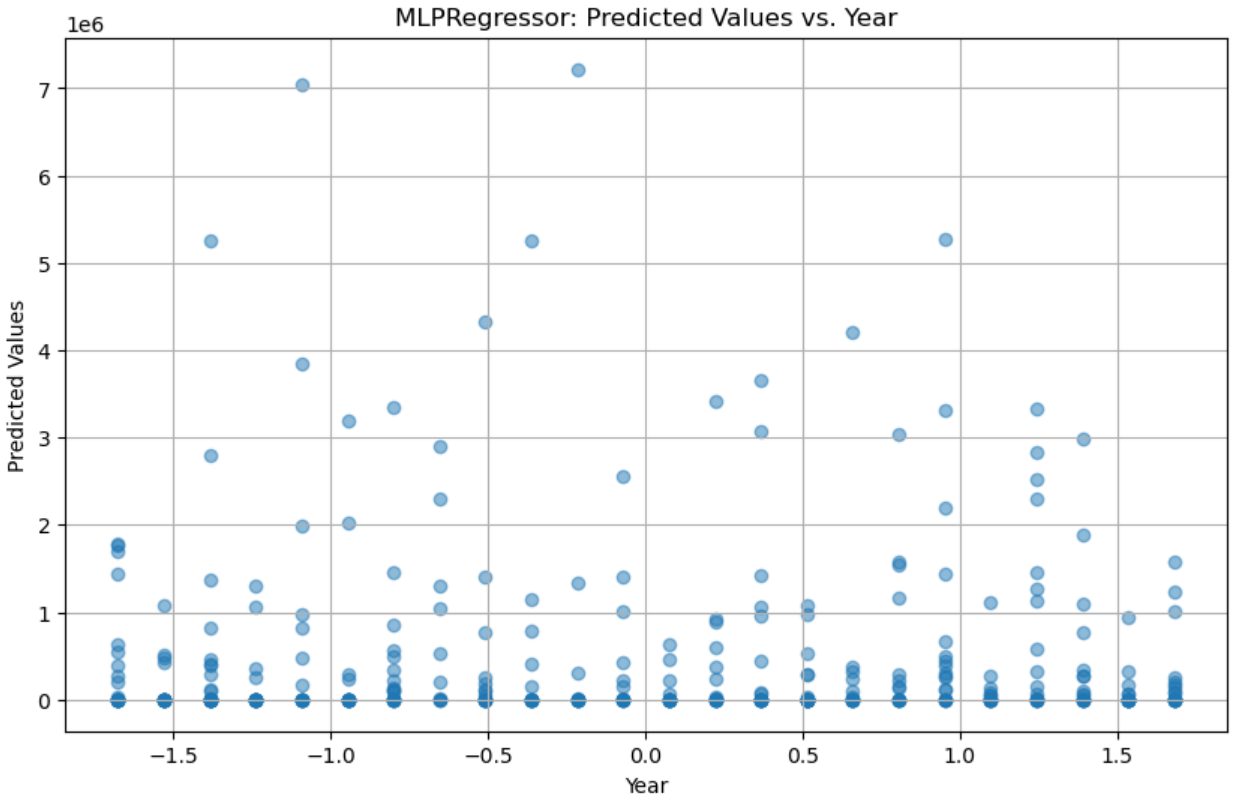
```
Mean Squared Error: 125718908324.67043
Normalized Mean Squared Error: 0.13428035987886638
Mean Absolute Error: 166230.1403851357
Normalized Mean Absolute Error: 0.023873779714042406
R^2 Score: 0.8657196401211337
```

```
# Plotting
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, alpha=0.5, label='Predicted vs. Actual')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()],
'k--', lw=2, label='Ideal Fit')
plt.xlabel('True Values')
plt.ylabel('Predicted Values')
plt.title('MLPRegressor: True vs. Predicted Values')
plt.legend()
plt.grid(True)
plt.show()

# Additional plot to visualize residuals
plt.figure(figsize=(10, 6))
residuals = y_test - y_pred
plt.scatter(y_pred, residuals, alpha=0.5)
plt.hlines(y=0, xmin=y_pred.min(), xmax=y_pred.max(), colors='r',
linestyles='--')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('Residuals Plot')
plt.grid(True)
plt.show()

# Plot y_pred vs. X_test['Year']
plt.figure(figsize=(10, 6))
plt.scatter(X_test[:, list(X.columns).index('Year Code')], y_pred,
alpha=0.5)
```

```
plt.xlabel('Year')
plt.ylabel('Predicted Values')
plt.title('MLPRegressor: Predicted Values vs. Year')
plt.grid(True)
plt.show()
```




Residuals Plot
1e6
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
Residuals
0
1
2
3
4
5
6
7
1e6
Predicted Values


MLPRegressor: Predicted Values vs. Year
1e6
7
6
5
4
3
2
1
0
Predicted Values
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
Year

```
# Drop duplicate rows
merged_df =
merged_df.loc[~merged_df.duplicated(subset=common_columns), :]
merged_df
```

```
Element  Area Code (M49)  Year Code       Value  Area Code (M49)  Year
Code  \
0                      4       2000   26.629848                4
2000
1                      4       2001  653.981386                4
2001
2                      4       2002  930.398250                4
2002
3                      4       2003  725.213777                4
2003
4                      4       2004  726.528864                4
2004
...                  ...        ...         ...              ...
...
1928                 344       2008  445.215511              364
2005
1929                 344       2009  416.232295              364
2006
1930                 344       2010  391.710717              364
2007
1931                 344       2011  370.764381              364
2008
1932                 344       2012  352.727198              364
2009

Element          Yield  Area Code (M49)  Year Code  \
0         60177.909091                4       2000
1         60701.272727                4       2001
2         61135.363636                4       2002
3         61209.181818                4       2003
4         61449.454545                4       2004
...                ...              ...        ...
1928     124882.727273              328       2014
1929     121720.545455              328       2015
1930     128180.909091              328       2016
1931     113527.545455              328       2017
1932     116735.363636              328       2018

Element  Crops total (Emissions CH4)  Crops total (Emissions N2O)  ...
\
0                            20.8471                       0.7056  ...

1                            19.2605                       0.7054  ...

2                            21.2553                       1.0656  ...
```

```
3                              23.7017                         1.3117  ...
4                              30.3089                         1.0856  ...
...                                ...                            ...  ...
1928                           49.6861                         0.1733  ...
1929                           51.1828                         0.1833  ...
1930                           40.2734                         0.1437  ...
1931                           46.1502                         0.1666  ...
1932                           44.7573                         0.1633  ...

Element  Temperature change  Area Code (M49)  Year Code          Area
\
0                   0.993000                4       1980  28356.666667
1                   1.311000                4       1981  28360.111111
2                   1.365000                4       1982  28362.222222
3                   0.587000                4       1983  28363.888889
4                   1.373200                4       1984  28367.333333
...                      ...              ...        ...           ...
1928                0.949000              180       1982  60686.555556
1929                1.425250              180       1983  60699.888889
1930                1.051048              180       1984  60722.111111
1931                1.051048              180       1985  60744.444444
1932                0.679000              180       1986  60766.666667

Element  Area Code (M49)  Year Code  Agricultural Use  \
0                      8       2000         86.842857
1                      8       2001         89.870000
2                      8       2002         92.895714
3                      8       2003         95.920000
4                      8       2004         98.945714
...                  ...        ...               ...
1928                 352       2014          1.105714
```

```
1929                  352       2015          1.158571
1930                  352       2016          0.857143
1931                  352       2017          0.571429
1932                  352       2018          0.571429

Element  Use per area of cropland  Use per value of agricultural
production  \
0                                0.44
0.23
1                                0.46
0.23
2                                0.47
0.24
3                                0.49
0.24
4                                0.51
0.23
...                               ...
...
1928                             0.03
0.02
1929                             0.03
0.02
1930                             0.02
0.02
1931                             0.02
0.01
1932                             0.02
0.01

Element  instance_id
0             8_2000
1             8_2001
2             8_2002
3             8_2003
4             8_2004
...              ...
1928        352_2014
1929        352_2015
1930        352_2016
1931        352_2017
1932        352_2018

[1933 rows x 52 columns]

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.neural_network import MLPRegressor
```

```
import os

# Assuming that 'Area' and 'Year Code' are the common columns in all
dataframes
common_columns = ['Area', 'Year Code']

# Merge all dataframes on 'Area' and 'Year Code'
merged_df = pd.concat(dataframes, axis=1, join='inner')

# Remove duplicate 'Year Code' entries
merged_df = merged_df.drop_duplicates(subset='Year Code',
keep='first')

# Select the unique_selected_features from merged_df for X
X = merged_df[unique_selected_features]

# Set y as 'Export Value' from merged_df
y = merged_df['Export Value']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Store column names before scaling
column_names = X.columns

# Standardize the data
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Convert numpy arrays back to DataFrames to keep column names
X_train = pd.DataFrame(X_train, columns=column_names)
X_test = pd.DataFrame(X_test, columns=column_names)

# Extract 'Year Code' from X_test
test_years = X_test['Year Code'].copy()

# Create a MLPRegressor model
model = MLPRegressor(hidden_layer_sizes=(64, 32, 16), max_iter=10000,
activation='relu', solver='adam', random_state=42)

# Fit the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Create a DataFrame with true and predicted values
if 'Area' in X_test.columns:
    test_areas = X_test['Area'].values.ravel()
```

```
    test_years = X_test['Year Code'].values.ravel()
    test_results_df = pd.DataFrame({
        'Area': test_areas,
        'Year Code': test_years,
        'Real value (dollars)': y_test.ravel(),
        'Predicted value (dollars)': y_pred.ravel()
    })
else:
    test_years = X_test['Year Code'].values.ravel()
    test_results_df = pd.DataFrame({
        'Year Code': test_years,
        'Real value (dollars)': y_test.to_numpy(),
        'Predicted value (dollars)': y_pred
    })

# Save the test results to a CSV file
output_test_results_filename = 'test_results_with_area.csv'
test_results_df.to_csv(output_test_results_filename, index=False)
print(f'Test results with true and predicted values are saved to
{output_test_results_filename}')
print(test_results_df.head())

# Predict future values for the next three years
last_year = X_test['Year Code'].max()
num_future_years = 3
future_predictions = []
for area in X['Area'].unique() if 'Area' in X.columns else
['General']:
    for i in range(num_future_years):
        year = last_year + 1 + i
        # Use the mean values of the features for the future
predictions
        mean_features = X[unique_selected_features].mean().to_dict()
        mean_features['Year Code'] = year
        if 'Area' in X.columns:
            mean_features['Area'] = area
        future_predictions.append(mean_features)
future_df = pd.DataFrame(future_predictions)

# Preserve the 'Area' and 'Year' columns for future predictions
if 'Area' in future_df.columns:
    future_areas = future_df['Area']
    future_years = future_df['Year Code']
else:
    future_years = future_df['Year Code']

# Select the same features from future_df that were used in X_train
future_features = future_df[unique_selected_features]

# Standardize the future features
```

```
future_features_scaled = scaler.transform(future_features)

# Make predictions for future years
future_predictions = model.predict(future_features_scaled)

# Create a DataFrame with future predictions and include 'Area' and
'Year'
if 'Area' in future_df.columns:
    future_results_df = pd.DataFrame({
        'Area': future_areas,
        'Year Code': future_years,
        'Predicted value (dollars)': future_predictions
    })
else:
    future_results_df = pd.DataFrame({
        'Year Code': future_years,
        'Predicted value (dollars)': future_predictions
    })

# Save the future results to a CSV file
output_future_results_filename = 'future_predictions_with_area.csv'
future_results_df.to_csv(output_future_results_filename, index=False)
print(f'Future predictions are saved to
{output_future_results_filename}')
print(future_results_df.head())
```

```
/var/folders/65/vppmmcj54y79z48zyz80vrk80000gn/T/
ipykernel_10918/2851498160.py:57: FutureWarning: Series.ravel is
deprecated. The underlying array is already 1D, so ravel is not
necessary.  Use `to_numpy()` for conversion to a numpy array instead.
  'Real value (dollars)': y_test.ravel(),

---------------------------------------------------------------------
-----
ValueError                                Traceback (most recent call
last)
Cell In[560], line 54
     52     test_areas = X_test['Area'].values.ravel()
     53     test_years = X_test['Year Code'].values.ravel()
---> 54     test_results_df = pd.DataFrame({
     55         'Area': test_areas,
     56         'Year Code': test_years,
     57         'Real value (dollars)': y_test.ravel(),
     58         'Predicted value (dollars)': y_pred.ravel()
     59     })
     60 else:
     61     test_years = X_test['Year Code'].values.ravel()

File
/opt/anaconda3/lib/python3.11/site-packages/pandas/core/frame.py:778,
```

```
in DataFrame.__init__(self, data, index, columns, dtype, copy)
    772     mgr = self._init_mgr(
    773         data, axes={"index": index, "columns": columns},
dtype=dtype, copy=copy
    774     )
    776 elif isinstance(data, dict):
    777     # GH#38939 de facto copy defaults to False only in non-
dict cases
--> 778     mgr = dict_to_mgr(data, index, columns, dtype=dtype,
copy=copy, typ=manager)
    779 elif isinstance(data, ma.MaskedArray):
    780     from numpy.ma import mrecords

File
/opt/anaconda3/lib/python3.11/site-packages/pandas/core/internals/
construction.py:503, in dict_to_mgr(data, index, columns, dtype, typ,
copy)
    499     else:
    500         # dtype check to exclude e.g. range objects, scalars
    501         arrays = [x.copy() if hasattr(x, "dtype") else x for x
in arrays]
--> 503 return arrays_to_mgr(arrays, columns, index, dtype=dtype,
typ=typ, consolidate=copy)

File
/opt/anaconda3/lib/python3.11/site-packages/pandas/core/internals/
construction.py:114, in arrays_to_mgr(arrays, columns, index, dtype,
verify_integrity, typ, consolidate)
    111 if verify_integrity:
    112     # figure out the index, if necessary
    113     if index is None:
--> 114         index = _extract_index(arrays)
    115     else:
    116         index = ensure_index(index)

File
/opt/anaconda3/lib/python3.11/site-packages/pandas/core/internals/
construction.py:677, in _extract_index(data)
    675 lengths = list(set(raw_lengths))
    676 if len(lengths) > 1:
--> 677     raise ValueError("All arrays must be of the same length")
    679 if have_dicts:
    680     raise ValueError(
    681         "Mixing dicts with non-Series may lead to ambiguous
ordering."
    682     )

ValueError: All arrays must be of the same length
```

```
print("Length of test_years: ", len(test_years))
print("Length of y_test: ", len(y_test))
print("Length of y_pred: ", len(y_pred))
```

```
Length of test_years:  5031
Length of y_test:  387
Length of y_pred:  387
```

```
print('Area' in future_df.columns)
```

```
True
```

```
print('Area' in X_train.columns)
```

```
True
```

```
print("Length of test_area_codes: ", len(test_area_codes))
print("Length of test_year_codes: ", len(test_year_codes))
print("Length of y_test: ", len(y_test))
print("Length of y_pred: ", len(y_pred))
```

```
Length of test_area_codes:  387
Length of test_year_codes:  387
Length of y_test:  387
Length of y_pred:  387
```