```
!pip install --upgrade pip
!pip install pandas
!pip install scikit-learn
```

```
Requirement already satisfied: pip in
/opt/anaconda3/lib/python3.11/site-packages (24.0)
Requirement already satisfied: pandas in
/opt/anaconda3/lib/python3.11/site-packages (2.1.4)
Requirement already satisfied: numpy<2,>=1.23.2 in
/opt/anaconda3/lib/python3.11/site-packages (from pandas) (1.26.4)
Requirement already satisfied: python-dateutil>=2.8.2 in
/opt/anaconda3/lib/python3.11/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in
/opt/anaconda3/lib/python3.11/site-packages (from pandas)
(2023.3.post1)
Requirement already satisfied: tzdata>=2022.1 in
/opt/anaconda3/lib/python3.11/site-packages (from pandas) (2023.3)
Requirement already satisfied: six>=1.5 in
/opt/anaconda3/lib/python3.11/site-packages (from python-
dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: scikit-learn in
/opt/anaconda3/lib/python3.11/site-packages (1.2.2)
Requirement already satisfied: numpy>=1.17.3 in
/opt/anaconda3/lib/python3.11/site-packages (from scikit-learn)
(1.26.4)
Requirement already satisfied: scipy>=1.3.2 in
/opt/anaconda3/lib/python3.11/site-packages (from scikit-learn)
(1.11.4)
Requirement already satisfied: joblib>=1.1.1 in
/opt/anaconda3/lib/python3.11/site-packages (from scikit-learn)
(1.2.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/opt/anaconda3/lib/python3.11/site-packages (from scikit-learn)
(2.2.0)
```

```
#Import necessary libraries
import pandas as pd  # For data manipulation and analysis
import numpy as np  # For numerical computations

# Importing functions for model training and evaluation
from sklearn.model_selection import train_test_split  # To split the
data into training and testing sets
from sklearn.preprocessing import StandardScaler  # To standardize
features
from sklearn.neural_network import MLPRegressor  # To build a Multi-
Layer Perceptron regressor model
from sklearn.metrics import mean_squared_error, r2_score  # To
evaluate the model's performance

# Importing functions for data preprocessing
```

```
from sklearn.preprocessing import OneHotEncoder  # To encode
categorical features as one-hot numeric arrays
from sklearn.compose import ColumnTransformer  # To apply different
preprocessing steps to different columns
from sklearn.pipeline import Pipeline  # To chain multiple
preprocessing steps and the model into a single pipeline

# Importing functions for hyperparameter tuning
from sklearn.model_selection import GridSearchCV  # To perform
hyperparameter tuning using cross-validation
import matplotlib.pyplot as plt

from sklearn.preprocessing import PowerTransformer



food_trade_indicators_file_path = 'ML Coursework Dataset/Food trade
indicators - FAOSTAT_data_en_2-22-2024.csv'


food_trade_indicators_df =
pd.read_csv(food_trade_indicators_file_path)


food_trade_indicators_df


print(food_trade_indicators_df.columns)

Index(['Domain Code', 'Domain', 'Area Code (M49)', 'Area', 'Element
Code',
       'Element', 'Item Code (CPC)', 'Item', 'Year Code', 'Year',
'Unit',
       'Value', 'Flag', 'Flag Description', 'Note'],
      dtype='object')


if food_trade_indicators_df['Value'].isnull().any():
    median_value = food_trade_indicators_df['Value'].median()
    food_trade_indicators_df['Value'].fillna(median_value,
inplace=True)  # Fill missing values with the median


print(food_trade_indicators_df.info())


if (food_trade_indicators_df['Value'] < 0).any():
    print("Negative values found in 'Value'. Setting them to their
absolute values.")
    food_trade_indicators_df['Value'] =
food_trade_indicators_df['Value'].abs()
```

```
print(food_trade_indicators_df.describe())


print(food_trade_indicators_df.head())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 141738 entries, 0 to 141737
Data columns (total 15 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   Domain Code         141738 non-null  object
 1   Domain              141738 non-null  object
 2   Area Code (M49)     141738 non-null  int64
 3   Area                141738 non-null  object
 4   Element Code        141738 non-null  int64
 5   Element             141738 non-null  object
 6   Item Code (CPC)     141738 non-null  object
 7   Item                141738 non-null  object
 8   Year Code           141738 non-null  int64
 9   Year                141738 non-null  int64
 10  Unit                141738 non-null  object
 11  Value               141738 non-null  float64
 12  Flag                141738 non-null  object
 13  Flag Description    141738 non-null  object
 14  Note                0 non-null       float64
dtypes: float64(2), int64(4), object(9)
memory usage: 16.2+ MB
None
       Area Code (M49)   Element Code      Year Code           Year  \
count    141738.000000  141738.000000  141738.000000  141738.000000
mean        424.988359    5765.555010    2006.724273    2006.724273
std         253.512489     149.862005       9.168199       9.168199
min           4.000000    5622.000000    1991.000000    1991.000000
25%         204.000000    5622.000000    1999.000000    1999.000000
50%         414.000000    5622.000000    2007.000000    2007.000000
75%         643.000000    5922.000000    2015.000000    2015.000000
max         894.000000    5922.000000    2022.000000    2022.000000

              Value  Note
count  1.417380e+05   0.0
mean   4.572981e+05   NaN
std    1.876930e+06   NaN
min    0.000000e+00   NaN
25%    2.150000e+03   NaN
50%    2.406200e+04   NaN
75%    1.764239e+05   NaN
max    8.355806e+07   NaN
  Domain Code                        Domain  Area Code (M49)
```

```
Area  \
0         TCL  Crops and livestock products                4
Afghanistan
1         TCL  Crops and livestock products                4
Afghanistan
2         TCL  Crops and livestock products                4
Afghanistan
3         TCL  Crops and livestock products                4
Afghanistan
4         TCL  Crops and livestock products                4
Afghanistan

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

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

## Feature Engineering and Data Filtering

These steps prepare the dataset (Food trade indicators - FAOSTAT_data_en_2-22-2024.csv) by focusing on relevant export data and encoding categorical variables for further analysis and modeling.

```
# Create a copy of the 'Element' column to use for creating dummy
variables
food_trade_indicators_df['Element Copy'] =
food_trade_indicators_df['Element']

food_trade_indicators_export_import_df =
pd.get_dummies(food_trade_indicators_df, columns=['Element Copy'],
prefix='', prefix_sep='')

food_trade_indicators_export_import_df
```

```
count_export_true = (food_trade_indicators_export_import_df['Export 
Value'] == True).sum()

count_import_true = (food_trade_indicators_export_import_df['Import 
Value'] == True).sum()


count_import_true + count_export_true

food_trade_indicators_export_df = 
food_trade_indicators_export_import_df[food_trade_indicators_export_im
port_df['Export Value'] == True]

food_trade_indicators_export_df
```

```
       Domain Code                         Domain  Area Code (M49)  \
19             TCL  Crops and livestock products                 4   
21             TCL  Crops and livestock products                 4   
23             TCL  Crops and livestock products                 4   
25             TCL  Crops and livestock products                 4   
27             TCL  Crops and livestock products                 4   
...            ...                           ...               ...   
141729         TCL  Crops and livestock products               716   
141731         TCL  Crops and livestock products               716   
141733         TCL  Crops and livestock products               716   
141735         TCL  Crops and livestock products               716   
141737         TCL  Crops and livestock products               716   

               Area  Element Code       Element Item Code (CPC)  \
19      Afghanistan          5922  Export Value           F1888   
21      Afghanistan          5922  Export Value           F1888   
23      Afghanistan          5922  Export Value           F1888   
25      Afghanistan          5922  Export Value           F1888   
27      Afghanistan          5922  Export Value           F1888   
...             ...           ...           ...             ...   
141729     Zimbabwe          5922  Export Value           F1896   
141731     Zimbabwe          5922  Export Value           F1896   
141733     Zimbabwe          5922  Export Value           F1896   
141735     Zimbabwe          5922  Export Value           F1896   
141737     Zimbabwe          5922  Export Value           F1896   

                            Item  Year Code  Year      Unit      Value 
Flag  \
19      Cereals and Preparations       2009  2009  1000 USD      15.00   
A   
21      Cereals and Preparations       2010  2010  1000 USD      54.00   
A   
23      Cereals and Preparations       2011  2011  1000 USD       0.00   
E   
25      Cereals and Preparations       2012  2012  1000 USD       0.00   
```

```
E
27      Cereals and Preparations       2013  2013  1000 USD       0.00
E
...                          ...        ...   ...       ...        ...
...
141729                   Tobacco       2018  2018  1000 USD  893113.05
A
141731                   Tobacco       2019  2019  1000 USD  828488.44
A
141733                   Tobacco       2020  2020  1000 USD  794956.99
A
141735                   Tobacco       2021  2021  1000 USD  836533.69
A
141737                   Tobacco       2022  2022  1000 USD  998057.60
A

       Flag Description  Note  Export Value  Import Value
19      Official figure   NaN          True         False
21      Official figure   NaN          True         False
23      Estimated value   NaN          True         False
25      Estimated value   NaN          True         False
27      Estimated value   NaN          True         False
...                 ...   ...           ...           ...
141729  Official figure   NaN          True         False
141731  Official figure   NaN          True         False
141733  Official figure   NaN          True         False
141735  Official figure   NaN          True         False
141737  Official figure   NaN          True         False

[67824 rows x 17 columns]
```

## Data Cleaning

This code block **below** performs the following tasks to clean and prepare the dataset:

1. Display Value Counts
2. Count Missing Values
3. Drop Unnecessary Columns
4. Convert `Unit` Column
5. Calculate `Export Value
6. Drop Redundant Columns

```
food_trade_indicators_export_df['Item'].value_counts()

food_trade_indicators_export_df.isnull().sum()

food_trade_indicators_export_df =
food_trade_indicators_export_df.drop(columns=['Import Value', 'Export
Value', 'Note'])
```

```
food_trade_indicators_export_df['Unit'] =
food_trade_indicators_export_df['Unit'].str.replace(' USD', '',
regex=False).astype(float)

food_trade_indicators_export_df['Export Value'] =
food_trade_indicators_export_df['Unit'] *
food_trade_indicators_export_df['Value']

columns_to_drop = ['Unit', 'Value', 'Domain Code', 'Area Code (M49)',
'Element Code', 'Element', 'Item Code (CPC)', 'Year Code', 'Flag']

food_trade_indicators_export_df =
food_trade_indicators_export_df.drop(columns=columns_to_drop)

food_trade_indicators_export_df.head(5)

                          Domain         Area
Item  Year  \
19  Crops and livestock products  Afghanistan  Cereals and
Preparations  2009
21  Crops and livestock products  Afghanistan  Cereals and
Preparations  2010
23  Crops and livestock products  Afghanistan  Cereals and
Preparations  2011
25  Crops and livestock products  Afghanistan  Cereals and
Preparations  2012
27  Crops and livestock products  Afghanistan  Cereals and
Preparations  2013

   Flag Description  Export Value
19  Official figure       15000.0
21  Official figure       54000.0
23  Estimated value           0.0
25  Estimated value           0.0
27  Estimated value           0.0

food_trade_indicators_export_df.tail(5)

                              Domain      Area     Item  Year  \
141729  Crops and livestock products  Zimbabwe  Tobacco  2018
141731  Crops and livestock products  Zimbabwe  Tobacco  2019
141733  Crops and livestock products  Zimbabwe  Tobacco  2020
141735  Crops and livestock products  Zimbabwe  Tobacco  2021
141737  Crops and livestock products  Zimbabwe  Tobacco  2022

       Flag Description  Export Value
141729  Official figure   893113050.0
141731  Official figure   828488440.0
141733  Official figure   794956990.0
```

```
141735  Official figure   836533690.0
141737  Official figure   998057600.0

output_file_path = 'ML Coursework
Dataset/processed_food_trade_indicators.csv'

food_trade_indicators_export_df.to_csv(output_file_path, index=False)
```

## Processing Food Security Indicators Dataset

```
food_security_indicator_df = pd.read_csv('ML Coursework Dataset/Food
security indicators  - FAOSTAT_data_en_2-22-2024.csv')

# One-hot encode the 'Item' column
food_security_indicator_df_itemized =
pd.get_dummies(food_security_indicator_df, columns=['Item'])

food_security_indicator_df_itemized =
food_security_indicator_df_itemized[food_security_indicator_df_itemize
d['Item_Value of food imports in total merchandise exports (percent)
(3-year average)'] == True]

columns_to_keep = ['Area', 'Year Code', 'Year', 'Unit', 'Value']

# Keep only the specified columns in the DataFrame
food_security_indicator_df_itemized =
food_security_indicator_df_itemized.loc[:,
food_security_indicator_df_itemized.columns.isin(columns_to_keep)]

# Function to transform 'Unit' based on specified rules
def transform_unit(unit):
    if unit.endswith('%'):
        return 1/100
    elif unit in ['g/pc/d', 'index']:
        return 1
    else:
        return None

food_security_indicator_df_itemized['Transformed Unit'] =
food_security_indicator_df_itemized['Unit'].apply(transform_unit)

food_security_indicator_df_itemized['food imports in total merchandise
exports-percent-3-year average'] =
food_security_indicator_df_itemized['Value'] *
food_security_indicator_df_itemized['Transformed Unit'].fillna(0)

food_security_indicator_df_itemized

              Area  Year Code      Year Unit  Value  Transformed Unit
\
79     Afghanistan   20002002  2000-2002    %  240.0                0.01
80     Afghanistan   20012003  2001-2003    %  281.0                0.01
81     Afghanistan   20022004  2002-2004    %  199.0                0.01
82     Afghanistan   20032005  2003-2005    %  187.0                0.01
83     Afghanistan   20042006  2004-2006    %  175.0                0.01
...            ...        ...        ...  ...    ...                 ...
36403     Zimbabwe   20152017  2015-2017    %   25.0                0.01
36404     Zimbabwe   20162018  2016-2018    %   20.0                0.01
36405     Zimbabwe   20172019  2017-2019    %   13.0                0.01
36406     Zimbabwe   20182020  2018-2020    %   14.0                0.01
36407     Zimbabwe   20192021  2019-2021    %   15.0                0.01

       food imports in total merchandise exports-percent-3-year
average
79                                                         2.40
80                                                         2.81
81                                                         1.99
82                                                         1.87
83                                                         1.75
...                                                         ...
36403                                                      0.25
36404                                                      0.20
36405                                                      0.13
36406                                                      0.14
36407                                                      0.15

[3858 rows x 7 columns]

# Drop unnecessary columns from the food security indicators DataFrame
columns_to_drop = ['Unit', 'Transformed Unit', 'Value']
```

```
food_security_indicator_df_itemized =
food_security_indicator_df_itemized.drop(columns=columns_to_drop)

# Extract the last year from the 'Year' column and convert it to an
integer
food_security_indicator_df_itemized['Year'] =
food_security_indicator_df_itemized['Year'].astype(str).str.split('-')
.str[-1].astype(int)

food_trade_indicators_export_df.columns

food_security_indicator_df_itemized.columns

food_trade_indicators_export_security_indicator_df =
pd.merge(food_trade_indicators_export_df,
food_security_indicator_df_itemized,
                     on=['Area', 'Year'], how='left')

food_trade_indicators_export_security_indicator_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 67824 entries, 0 to 67823
Data columns (total 8 columns):
 #   Column
Non-Null Count  Dtype
---  ------
--------------  -----
 0   Domain
67824 non-null  object
 1   Area
67824 non-null  object
 2   Item
67824 non-null  object
 3   Year
67824 non-null  int64
 4   Flag Description
67824 non-null  object
 5   Export Value
67824 non-null  float64
 6   Year Code
42919 non-null  float64
 7   food imports in total merchandise exports-percent-3-year average
42919 non-null  float64
dtypes: float64(3), int64(1), object(4)
memory usage: 4.1+ MB
```

```
output_file_path = 'ML Coursework
Dataset/processed_food_trade_indicators_export_security_indicator.csv'
```

```
food_trade_indicators_export_security_indicator_df.to_csv(output_file_
path, index=False)
```

## Processing Food Balances Indicators Dataset

```
food_balance_df = pd.read_csv('ML Coursework Dataset/Food balances
indicators - FAOSTAT_data_en_2-22-2024.csv')

# Perform one-hot encoding on the 'Element' column
food_balance_df = pd.get_dummies(food_balance_df, columns=['Element'])

# Filter the DataFrame to keep rows where 'Element_Export Quantity' is
True
food_balance_export_quantity_df =
food_balance_df[food_balance_df['Element_Export Quantity'] == True]

columns_to_keep = ['Area', 'Item', 'Year Code', 'Year', 'Unit',
'Value', 'Flag', 'Flag Description', 'Element_Export Quantity']

food_balance_export_quantity_prepared_df =
food_balance_export_quantity_df.loc[:,
food_balance_export_quantity_df.columns.isin(columns_to_keep)]

mapping_dict = {
    'Cereals - Excluding Beer': 'Cereals and Preparations',
    'Starchy Roots': 'Other food',
    'Sugar Crops': 'Sugar and Honey',
    'Sugar & Sweeteners': 'Sugar and Honey',
    'Pulses': 'Other food',
    'Treenuts': 'Other food',
    'Oilcrops': 'Fats and Oils (excluding Butter)',
    'Vegetable Oils': 'Fats and Oils (excluding Butter)',
    'Vegetables': 'Fruit and Vegetables',
    'Fruits - Excluding Wine': 'Fruit and Vegetables',
    'Stimulants': 'Non-alcoholic Beverages',
    'Spices': 'Other food',
    'Alcoholic Beverages': 'Alcoholic Beverages',
    'Meat': 'Meat and Meat Preparations',
    'Eggs': 'Dairy Products and Eggs',
    'Milk - Excluding Butter': 'Dairy Products and Eggs',
    'Fish, Seafood': 'Meat and Meat Preparations'
}

food_balance_export_quantity_prepared_df['export item'] =
food_balance_export_quantity_prepared_df['Item'].map(mapping_dict)

food_balance_export_quantity_prepared_df
```

```
/var/folders/cn/hpnpd66n0yd2mzw_dnkrygz80000gn/T/
ipykernel_50478/4065834294.py:33: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  food_balance_export_quantity_prepared_df['export item'] =
food_balance_export_quantity_prepared_df['Item'].map(mapping_dict)

               Area                      Item  Year Code  Year    Unit
Value  \
12      Afghanistan  Cereals - Excluding Beer       2010  2010  1000 t
0.00
13      Afghanistan  Cereals - Excluding Beer       2011  2011  1000 t
0.00
14      Afghanistan  Cereals - Excluding Beer       2012  2012  1000 t
0.00
15      Afghanistan  Cereals - Excluding Beer       2013  2013  1000 t
0.00
16      Afghanistan  Cereals - Excluding Beer       2014  2014  1000 t
2.00
...             ...                       ...        ...   ...     ...
...
148012     Zimbabwe             Fish, Seafood       2017  2017  1000 t
3.85
148013     Zimbabwe             Fish, Seafood       2018  2018  1000 t
4.94
148014     Zimbabwe             Fish, Seafood       2019  2019  1000 t
5.53
148015     Zimbabwe             Fish, Seafood       2020  2020  1000 t
5.53
148016     Zimbabwe             Fish, Seafood       2021  2021  1000 t
5.53

       Flag Flag Description  Element_Export Quantity  \
12        E  Estimated value                     True
13        E  Estimated value                     True
14        E  Estimated value                     True
15        E  Estimated value                     True
16        E  Estimated value                     True
...     ...              ...                      ...
148012    E  Estimated value                     True
148013    E  Estimated value                     True
148014    E  Estimated value                     True
148015    E  Estimated value                     True
148016    E  Estimated value                     True

                       export item
12        Cereals and Preparations
13        Cereals and Preparations
```

```
14        Cereals and Preparations
15        Cereals and Preparations
16        Cereals and Preparations
...                            ...
148012  Meat and Meat Preparations
148013  Meat and Meat Preparations
148014  Meat and Meat Preparations
148015  Meat and Meat Preparations
148016  Meat and Meat Preparations

[33676 rows x 10 columns]
```

```
# Define a function to convert units
def convert_unit(unit):
    if 't' in unit:
        # Remove 't' and convert to float
        return float(unit.replace('t', ''))
    return 1000.0  # default factor if no specific unit is recognized

food_balance_export_quantity_prepared_df['Unit Numeric'] =
food_balance_export_quantity_prepared_df['Unit'].apply(convert_unit)

food_balance_export_quantity_prepared_df['export_quantity_tons'] =
food_balance_export_quantity_prepared_df['Value'] *
food_balance_export_quantity_prepared_df['Unit Numeric']

columns_to_keep = ['Area', 'export item', 'Year Code', 'Year',
'export_quantity_tons']
food_balance_export_quantity_prepared_df =
food_balance_export_quantity_prepared_df[columns_to_keep]

food_trade_indicators_export_security_balance_df = pd.merge(
    food_trade_indicators_export_security_indicator_df,
    food_balance_export_quantity_prepared_df,
    how='left',
    left_on=['Area', 'Year', 'Item'],
    right_on=['Area', 'Year', 'export item']
)

print(food_trade_indicators_export_security_balance_df.columns)
```

```
Index(['Domain', 'Area', 'Item', 'Year', 'Flag Description', 'Export
Value',
       'Year Code_x',
       'food imports in total merchandise exports-percent-3-year
average',
       'export item', 'Year Code_y', 'export_quantity_tons'],
      dtype='object')
```

```
/var/folders/cn/hpnpd66n0yd2mzw_dnkrygz80000gn/T/
ipykernel_50478/2043598582.py:8: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  food_balance_export_quantity_prepared_df['Unit Numeric'] =
food_balance_export_quantity_prepared_df['Unit'].apply(convert_unit)
/var/folders/cn/hpnpd66n0yd2mzw_dnkrygz80000gn/T/ipykernel_50478/20435
98582.py:10: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  food_balance_export_quantity_prepared_df['export_quantity_tons'] =
food_balance_export_quantity_prepared_df['Value'] *
food_balance_export_quantity_prepared_df['Unit Numeric']



output_file_path = 'ML Coursework
Dataset/processed_food_trade_indicators_export_security_balance.csv'
food_trade_indicators_export_security_balance_df.to_csv(output_file_pa
th, index=False)
```

## Processing Crops Production Indicators

```
# Integrate crops production
crops_production_df = pd.read_csv('ML Coursework Dataset/Crops
production indicators - FAOSTAT_data_en_2-22-2024.csv')

crops_production_df['Element'].unique()

crops_production_df['Unit'].unique()

# Convert 'Unit' to numeric
crops_production_df['Unit'] = crops_production_df['Unit'].astype(str)
crops_production_df['Unit'] =
crops_production_df['Unit'].str.extract('(\d+)').astype(float)

crops_production_df['yield_value'] = crops_production_df['Unit'] *
crops_production_df['Value']

columns_to_keep = ['Area', 'Item', 'Year Code', 'Year', 'yield_value']
crops_production_df = crops_production_df[columns_to_keep]

crops_production_df['Item'].unique()
```

```
food_trade_indicators_export_security_balance_df['Item'].unique()

mapping_dict = {
    'Cereals, primary': 'Cereals and Preparations',
    'Citrus Fruit, Total': 'Fruit and Vegetables',
    'Fibre Crops, Fibre Equivalent': 'Other food',
    'Fruit Primary': 'Fruit and Vegetables',
    'Oilcrops, Cake Equivalent': 'Fats and Oils (excluding Butter)',
    'Oilcrops, Oil Equivalent': 'Fats and Oils (excluding Butter)',
    'Pulses, Total': 'Other food',
    'Roots and Tubers, Total': 'Other food',
    'Sugar Crops Primary': 'Sugar and Honey',
    'Treenuts, Total': 'Other food',
    'Vegetables Primary': 'Fruit and Vegetables'
}

# Map the 'Item' column to new categories
crops_production_df['crops_target_item'] =
crops_production_df['Item'].map(mapping_dict)
crops_production_df

              Area                Item  Year Code  Year
yield_value  \
0      Afghanistan    Cereals, primary       2000  2000      806300.0

1      Afghanistan    Cereals, primary       2001  2001     1006700.0

2      Afghanistan    Cereals, primary       2002  2002     1669800.0

3      Afghanistan    Cereals, primary       2003  2003     1458000.0

4      Afghanistan    Cereals, primary       2004  2004     1334800.0

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

41644     Zimbabwe  Vegetables Primary       2018  2018     6651800.0

41645     Zimbabwe  Vegetables Primary       2019  2019     6483000.0

41646     Zimbabwe  Vegetables Primary       2020  2020     6562800.0

41647     Zimbabwe  Vegetables Primary       2021  2021     6612600.0

41648     Zimbabwe  Vegetables Primary       2022  2022     6585600.0


              crops_target_item
0      Cereals and Preparations
1      Cereals and Preparations
2      Cereals and Preparations
3      Cereals and Preparations
```

```
4      Cereals and Preparations
...                         ...
41644      Fruit and Vegetables
41645      Fruit and Vegetables
41646      Fruit and Vegetables
41647      Fruit and Vegetables
41648      Fruit and Vegetables

[41649 rows x 6 columns]
```

```
#This code block merges the processed crops production DataFrame with
the main DataFrame using a left join.
#The merge is based on the `Area`, `Year`, and `Item` columns from the
main DataFrame and the `Area`, `Year`, and `crops_target_item` columns
from the crops production DataFrame.
#This integration step is crucial for building a comprehensive dataset
that includes various features relevant to crop exports.

food_trade_indicators_export_security_balance_crops_prod_df =
pd.merge(
    left=food_trade_indicators_export_security_balance_df,
    right=crops_production_df,
    how='left',
    left_on=['Area', 'Year', 'Item'],
    right_on=['Area', 'Year', 'crops_target_item']
)

food_trade_indicators_export_security_balance_crops_prod_df.describe()
```

```
                Year  Export Value   Year Code_x  \
count  121123.000000  1.211230e+05  9.345100e+04
mean     2009.844811  8.459619e+08  2.011097e+07
std         8.441557  2.977981e+09  5.398682e+04
min      1991.000000  0.000000e+00  2.000200e+07
25%      2004.000000  1.554910e+06  2.007201e+07
50%      2012.000000  3.687600e+07  2.011201e+07
75%      2017.000000  3.535740e+08  2.015202e+07
max      2022.000000  5.784916e+10  2.019202e+07

       food imports in total merchandise exports-percent-3-year
average  \
count                                           93451.000000

mean                                                0.374212

std                                                 0.945548

min                                                 0.010000

25%                                                 0.060000
```

```
50%                                             0.130000
75%                                             0.300000
max                                            57.350000

        Year Code_y  export_quantity_tons     Year Code   yield_value
count  61585.000000          6.158500e+04  69053.000000  6.905300e+04
mean    2015.458212          4.907689e+05   2012.833678  1.014085e+07
std        3.464759          2.899271e+06      5.932474  1.599025e+07
min     2010.000000         -6.200000e+04   2000.000000  0.000000e+00
25%     2012.000000          0.000000e+00   2009.000000  8.604000e+05
50%     2015.000000          1.000000e+04   2013.000000  3.333300e+06
75%     2018.000000          1.110000e+05   2018.000000  1.313840e+07
max     2021.000000          1.102460e+08   2022.000000  1.359231e+08
```

```
columns_to_drop = ['Year Code_x','Year Code_y','Year Code']
food_trade_indicators_export_security_balance_crops_prod_df =
food_trade_indicators_export_security_balance_crops_prod_df.drop(colum
ns = columns_to_drop)

columns_to_drop = ['Flag Description','export
item','Item_y','crops_target_item']
food_trade_indicators_export_security_balance_crops_prod_df =
food_trade_indicators_export_security_balance_crops_prod_df.drop(colum
ns = columns_to_drop)

output_file_path = 'ML Coursework
Dataset/processed_food_trade_indicators_export_security_balance_crops_
prod.csv'
food_trade_indicators_export_security_balance_crops_prod_df.to_csv(out
put_file_path, index=False)
```

## Exploring Unique Elements in Land Use Data

```
land_use_df = pd.read_csv('ML Coursework Dataset/Land use -
FAOSTAT_data_en_2-22-2024.csv', low_memory=False)

land_use_df['Element'].unique()

array(['Area'], dtype=object)
```

```
land_use_df['Unit'].unique()

array(['1000 ha'], dtype=object)

# Convert 'Unit' column to numeric values
land_use_df['Unit'] = land_use_df['Unit'].astype(str)  # Convert
'Unit' to string type
land_use_df['Unit'] = land_use_df['Unit'].str.extract('(\
d+)').astype(float)  # Extract numeric values from 'Unit' and convert
to float

land_use_df['area_value'] = land_use_df['Unit'] * land_use_df['Value']

land_use_df['Item'].unique()

array(['Country area', 'Land area', 'Agriculture', 'Agricultural
land',
       'Cropland', 'Arable land', 'Temporary crops',
       'Temporary meadows and pastures', 'Temporary fallow',
       'Permanent crops', 'Permanent meadows and pastures',
       'Perm. meadows & pastures - Nat. growing',
       'Land area equipped for irrigation',
       'Land area actually irrigated',
       'Agriculture area actually irrigated',
       'Farm buildings and Farmyards', 'Cropland area actually
irrigated',
       'Perm. meadows & pastures - Cultivated',
       'Perm. meadows & pastures area actually irrig.',
       'Forestry area actually irrigated'], dtype=object)

# One-hot encode the 'Item' column
land_use_df = pd.get_dummies(land_use_df, columns=['Item'], prefix='',
prefix_sep='')

land_use_cropland_df = land_use_df[land_use_df['Cropland'] == True]
land_use_cropland_df

     Domain Code    Domain  Area Code (M49)         Area  Element
Code  \
168           RL  Land Use                4  Afghanistan
5110
169           RL  Land Use                4  Afghanistan
5110
170           RL  Land Use                4  Afghanistan
5110
171           RL  Land Use                4  Afghanistan
5110
172           RL  Land Use                4  Afghanistan
5110
...          ...       ...              ...          ...          ..
.
```

```
97759          RL  Land Use              716     Zimbabwe
5110
97760          RL  Land Use              716     Zimbabwe
5110
97761          RL  Land Use              716     Zimbabwe
5110
97762          RL  Land Use              716     Zimbabwe
5110
97763          RL  Land Use              716     Zimbabwe
5110

      Element  Item Code  Year Code  Year    Unit  ...  \
168      Area       6620       1980  1980  1000.0  ...
169      Area       6620       1981  1981  1000.0  ...
170      Area       6620       1982  1982  1000.0  ...
171      Area       6620       1983  1983  1000.0  ...
172      Area       6620       1984  1984  1000.0  ...
...       ...        ...        ...   ...     ...  ...
97759    Area       6620       2017  2017  1000.0  ...
97760    Area       6620       2018  2018  1000.0  ...
97761    Area       6620       2019  2019  1000.0  ...
97762    Area       6620       2020  2020  1000.0  ...
97763    Area       6620       2021  2021  1000.0  ...

       Land area actually irrigated Land area equipped for irrigation
\
168                           False                              False

169                           False                              False

170                           False                              False

171                           False                              False

172                           False                              False

...                             ...                                ...

97759                         False                              False

97760                         False                              False

97761                         False                              False

97762                         False                              False

97763                         False                              False


      Perm. meadows & pastures - Cultivated  \
168                                   False
```

```
169                                   False
170                                   False
171                                   False
172                                   False
...                                     ...
97759                                 False
97760                                 False
97761                                 False
97762                                 False
97763                                 False

      Perm. meadows & pastures - Nat. growing  \
168                                     False
169                                     False
170                                     False
171                                     False
172                                     False
...                                       ...
97759                                   False
97760                                   False
97761                                   False
97762                                   False
97763                                   False

       Perm. meadows & pastures area actually irrig.  Permanent crops
\
168                                            False            False

169                                            False            False

170                                            False            False

171                                            False            False

172                                            False            False

...                                              ...              ...

97759                                          False            False

97760                                          False            False

97761                                          False            False

97762                                          False            False

97763                                          False            False


       Permanent meadows and pastures  Temporary crops  Temporary
fallow  \
168                             False            False
False
169                             False            False
False
170                             False            False
False
171                             False            False
False
172                             False            False
False
...                               ...              ...                .
..
97759                           False            False
False
97760                           False            False
False
97761                           False            False
False
97762                           False            False
False
97763                           False            False
False

       Temporary meadows and pastures
168                             False
169                             False
170                             False
171                             False
172                             False
...                               ...
97759                           False
97760                           False
97761                           False
97762                           False
97763                           False

[9086 rows x 35 columns]
```

```
land_use_cropland_df.columns
```

```
Index(['Domain Code', 'Domain', 'Area Code (M49)', 'Area', 'Element
Code',
       'Element', 'Item Code', 'Year Code', 'Year', 'Unit', 'Value',
'Flag',
       'Flag Description', 'Note', 'area_value', 'Agricultural land',
       'Agriculture', 'Agriculture area actually irrigated', 'Arable
land',
       'Country area', 'Cropland', 'Cropland area actually irrigated',
       'Farm buildings and Farmyards', 'Forestry area actually
irrigated',
       'Land area', 'Land area actually irrigated',
       'Land area equipped for irrigation',
       'Perm. meadows & pastures - Cultivated',
       'Perm. meadows & pastures - Nat. growing',
       'Perm. meadows & pastures area actually irrig.', 'Permanent
crops',
       'Permanent meadows and pastures', 'Temporary crops', 'Temporary
fallow',
       'Temporary meadows and pastures'],
      dtype='object')
```

## Integrating Cropland Data with Main Dataset

```
columns_to_keep = ['Area','Year Code','Year','Unit', 'Value']
land_use_cropland_df = land_use_cropland_df[columns_to_keep]

land_use_cropland_df['Cropland'] = land_use_cropland_df['Unit'] *
land_use_cropland_df['Value']

columns_to_keep = ['Area','Year Code','Year','Cropland']
land_use_cropland_df = land_use_cropland_df[columns_to_keep]

foodtrade_export_security_balance_crops_prod_cropland_df = pd.merge(
    left=food_trade_indicators_export_security_balance_crops_prod_df,
    right=land_use_cropland_df,
    how='left',
    left_on=['Area', 'Year'],
    right_on=['Area', 'Year']
)

output_file_path = 'ML Coursework
Dataset/processed_food_trade_indicators_export_security_balance_crops_
prod_land.csv'
foodtrade_export_security_balance_crops_prod_cropland_df.to_csv(output
_file_path, index=False)
```

```
/var/folders/cn/hpnpd66n0yd2mzw_dnkrygz80000gn/T/
ipykernel_50478/2434386554.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  land_use_cropland_df['Cropland'] = land_use_cropland_df['Unit'] *
land_use_cropland_df['Value']
```

## Integrating Consumer Price Data

This code block processes the "Consumer prices indicators" dataset by reading the CSV file, selecting relevant columns, and identifying unique items in the 'Item' column. The unique items help in understanding the variety of data present in the dataset. And also, it processes the Consumer Prices dataset to extract and filter relevant data, performs one-hot encoding, aggregates the data, and merges it with the existing combined dataset. The goal is to integrate consumer price indices into the main dataset for further analysis and modeling.

```
consumer_price_df = pd.read_csv('ML Coursework Dataset/Consumer prices
indicators - FAOSTAT_data_en_2-22-2024.csv')

# Select relevant columns from the dataset
columns_to_keep = ['Area', 'Year', 'Item', 'Months', 'Element',
'Unit', 'Value']
consumer_price_df = consumer_price_df[columns_to_keep]

consumer_price_df['Item'].unique()
```

```
array(['Consumer Prices, Food Indices (2015 = 100)',
       'Food price inflation'], dtype=object)
```

```
consumer_price_df = pd.get_dummies(consumer_price_df,
columns=['Item'], prefix='', prefix_sep='')

consumer_price_df = consumer_price_df[consumer_price_df['Consumer
Prices, Food Indices (2015 = 100)'] == True]

consumer_price_df['Consumer_Prices_Food_Indices_2015_100'] =
consumer_price_df['Value']

columns_to_keep = ['Area', 'Year',
'Consumer_Prices_Food_Indices_2015_100']
consumer_price_df = consumer_price_df[columns_to_keep]

consumer_price_aggregated_df = consumer_price_df.groupby(['Year',
'Area']).sum().reset_index()

# Merge the aggregated consumer price data with the existing dataset
foodtrade_export_security_balance_crops_prod_cropland_consumer_indices
_df = pd.merge(
    left=foodtrade_export_security_balance_crops_prod_cropland_df,
    right=consumer_price_aggregated_df,
    how='left',
    left_on=['Area', 'Year'],
    right_on=['Area', 'Year']
)
```

## Integrating Exchange Rate Data

```
# Integrate the exchange rate to ensure all monetary values are in USD
regardless of the country
# The dataset is monthly, so we use the average value per year per
country

exchange_rate_df = pd.read_csv('ML Coursework Dataset/Exchange rate -
FAOSTAT_data_en_2-22-2024.csv')

exchange_rate_df['exchange_rate_value'] = exchange_rate_df['Value']

columns_to_keep = ['Area', 'Year', 'exchange_rate_value']
exchange_rate_df = exchange_rate_df[columns_to_keep]

exchange_rate_aggregated_df = exchange_rate_df.groupby(['Area',
'Year']).mean().reset_index()

foodtrade_export_security_balance_crops_prod_cropland_consumer_indices
_exchange_df = pd.merge(

left=foodtrade_export_security_balance_crops_prod_cropland_consumer_in
dices_df,
    right=exchange_rate_aggregated_df,
    how='left',
    left_on=['Area', 'Year'],
    right_on=['Area', 'Year']
)

# Calculate the export value in USD
foodtrade_export_security_balance_crops_prod_cropland_consumer_indices
_exchange_df['Export Value USD'] =
foodtrade_export_security_balance_crops_prod_cropland_consumer_indices
_exchange_df['Export Value'] *
foodtrade_export_security_balance_crops_prod_cropland_consumer_indices
_exchange_df['exchange_rate_value']

foodtrade_export_security_balance_crops_prod_cropland_consumer_indices
_exchange_df =
foodtrade_export_security_balance_crops_prod_cropland_consumer_indices
_exchange_df.drop_duplicates(keep='last')

# column yield to be aggregated with SUM for the items that maps to
the same target item in food trade dataset
foodtrade_export_security_balance_crops_prod_cropland_consumer_indices
_exchange_df.columns
```

```
Index(['Domain', 'Area', 'Item_x', 'Year', 'Export Value',
       'food imports in total merchandise exports-percent-3-year
average',
       'export_quantity_tons', 'yield_value', 'Year Code', 'Cropland',
       'Consumer_Prices_Food_Indices_2015_100', 'exchange_rate_value',
       'Export Value USD'],
      dtype='object')
```

```
foodtrade_export_security_balance_crops_prod_cropland_consumer_indices
_exchange_df =
foodtrade_export_security_balance_crops_prod_cropland_consumer_indices
_exchange_df.groupby(
    ['Domain', 'Area', 'Item_x', 'Year', 'Export Value',
     'food imports in total merchandise exports-percent-3-year
average',
     'Year Code', 'Cropland', 'Consumer_Prices_Food_Indices_2015_100',
'exchange_rate_value',
     'Export Value USD']).agg(
    {'yield_value':'sum', 'export_quantity_tons':'sum'}).reset_index()

output_file_path = 'ML Coursework
Dataset/processed_dataset_crop_products_export_price.csv'
foodtrade_export_security_balance_crops_prod_cropland_consumer_indices
_exchange_df.to_csv(output_file_path, index=False)

missing_values =
foodtrade_export_security_balance_crops_prod_cropland_consumer_indices
_exchange_df.isnull().sum()

missing_values
```

```
Domain                                                                0
Area                                                                  0
Item_x                                                                0
Year                                                                  0
Export Value                                                          0
food imports in total merchandise exports-percent-3-year average     0
Year Code                                                             0
Cropland                                                              0
Consumer_Prices_Food_Indices_2015_100                                 0
exchange_rate_value                                                   0
Export Value USD                                                      0
yield_value                                                           0
export_quantity_tons                                                  0
dtype: int64
```

```
foodtrade_export_security_balance_crops_prod_cropland_consumer_indices
_exchange_df
```

```
                             Domain         Area              Item_x
Year  \
0      Crops and livestock products  Afghanistan  Alcoholic Beverages
2014
1      Crops and livestock products  Afghanistan  Alcoholic Beverages
2015
```

```
2      Crops and livestock products  Afghanistan  Alcoholic Beverages
2016
3      Crops and livestock products  Afghanistan  Alcoholic Beverages
2018
4      Crops and livestock products  Afghanistan  Alcoholic Beverages
2020
...                             ...          ...                  ...
...
40248  Crops and livestock products     Zimbabwe              Tobacco
2017
40249  Crops and livestock products     Zimbabwe              Tobacco
2018
40250  Crops and livestock products     Zimbabwe              Tobacco
2019
40251  Crops and livestock products     Zimbabwe              Tobacco
2020
40252  Crops and livestock products     Zimbabwe              Tobacco
2021

       Export Value  \
0           39040.0
1           66620.0
2            8250.0
3           30940.0
4            8780.0
...             ...
40248   837638520.0
40249   893113050.0
40250   828488440.0
40251   794956990.0
40252   836533690.0

       food imports in total merchandise exports-percent-3-year
average  \
0                                                   3.83

1                                                   3.84

2                                                   4.11

3                                                   3.78

4                                                   2.84

...                                                  ...

40248                                               0.25

40249                                               0.20
```

```
40250                                               0.13

40251                                               0.14

40252                                               0.15


       Year Code   Cropland  Consumer_Prices_Food_Indices_2015_100  \
0         2014.0  7910000.0                            1210.343257   
1         2015.0  7910000.0                            1200.131287   
2         2016.0  7910000.0                            1268.173032   
3         2018.0  8010000.0                            1341.236770   
4         2020.0  8051000.0                            1531.306924   
...          ...        ...                                    ...   
40248     2017.0  4100000.0                            1190.492118   
40249     2018.0  4100000.0                            1367.369253   
40250     2019.0  4100000.0                            6238.523310   
40251     2020.0  4100000.0                           43733.310812   
40252     2021.0  4100000.0                           90002.505212   

       exchange_rate_value  Export Value USD  yield_value  \
0                57.247500      2.234942e+06          0.0   
1                61.143462      4.073377e+06          0.0   
2                67.866086      5.598952e+05          0.0   
3                72.083247      2.230256e+06          0.0   
4                76.813536      6.744228e+05          0.0   
...                    ...               ...          ...   
40248           361.893274      3.031357e+11          0.0   
40249           322.206265      2.877666e+11          0.0   
40250            16.923764      1.402114e+10          0.0   
40251            51.329013      4.080436e+10          0.0   
40252            88.552447      7.407711e+10          0.0   

       export_quantity_tons  
0                       0.0  
1                       0.0  
2                       0.0  
3                       0.0  
4                       0.0  
...                     ...  
40248                   0.0  
40249                   0.0  
40250                   0.0  
40251                   0.0  
40252                   0.0  

[40253 rows x 13 columns]
```

# Machine Learning

## Data Preparation and Feature Engineering

```
export_df =
foodtrade_export_security_balance_crops_prod_cropland_consumer_indices
_exchange_df

# Create the target variable by shifting 'Export Value USD' by 3 years
export_df["forecast_export_value_crops_3_years"] =
export_df.groupby(['Area', 'Item_x'])['Export Value USD'].shift(3)

export_df.dropna(subset = ['forecast_export_value_crops_3_years'],
inplace = True)

export_df.isnull().sum()

export_df = export_df.drop(columns = ['Domain', 'Year Code'])

export_sorted_df = export_df.sort_values(by=['Area', 'Year'])

# Create lag features for 'Export Value USD'
export_sorted_df['export_value_lag_1_year'] =
export_sorted_df.groupby('Area')['Export Value USD'].shift(1)
export_sorted_df['export_value_lag_2_years'] =
export_sorted_df.groupby('Area')['Export Value USD'].shift(2)
export_sorted_df['export_value_lag_3_years'] =
export_sorted_df.groupby('Area')['Export Value USD'].shift(3)

# Create a 3-year moving average feature for 'Export Value USD'
export_sorted_df['export_value_moving_avg_3yr'] =
export_sorted_df.groupby('Area')['Export Value USD'].transform(lambda
x: x.rolling(window=3, min_periods=1).mean())

mean1 = export_sorted_df['export_value_lag_1_year']
mean2 = export_sorted_df['export_value_lag_2_years']
mean3 = export_sorted_df['export_value_lag_3_years']

export_sorted_df['export_value_lag_1_year'] =
export_sorted_df.groupby(['Area', 'Item_x'])
['export_value_lag_1_year'].transform(lambda x: x.fillna(x.mean()))
export_sorted_df['export_value_lag_2_years'] =
export_sorted_df.groupby(['Area', 'Item_x'])
['export_value_lag_2_years'].transform(lambda x: x.fillna(x.mean()))
export_sorted_df['export_value_lag_3_years'] =
export_sorted_df.groupby(['Area', 'Item_x'])
['export_value_lag_3_years'].transform(lambda x: x.fillna(x.mean()))

export_sorted_df.dropna(inplace=True)
```

```
# Create a copy of the cleaned dataset
original_df = export_sorted_df.copy()


features =
export_sorted_df.drop(['forecast_export_value_crops_3_years'], axis=1)
target = export_sorted_df['forecast_export_value_crops_3_years']

export_sorted_df.columns

categorical_columns = ['Area', 'Item_x']
numerical_columns = ['Year', 'Export Value', 'food imports in total
merchandise exports-percent-3-year average',
                     'Cropland',
'Consumer_Prices_Food_Indices_2015_100', 'exchange_rate_value',
'Export Value USD', 'yield_value', 'export_quantity_tons',
                     'export_value_lag_1_year',
'export_value_lag_2_years', 'export_value_lag_3_years',
'export_value_moving_avg_3yr']

numerical_transformer = StandardScaler()
categorical_transformer = OneHotEncoder(handle_unknown='ignore')

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numerical_transformer, numerical_columns),
        ('cat', categorical_transformer, categorical_columns)
    ])

mlp_pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', MLPRegressor(hidden_layer_sizes=(60,40),
activation='relu', alpha = 0.001, learning_rate_init = 0.00001,
random_state=10, max_iter=100000))
])
X_train, X_test, y_train, y_test = train_test_split(features, target,
test_size=0.2, random_state=100)

X_train.shape
```

```
(27034, 15)
```

```
X_test.shape
```

```
(6759, 15)
```

```
y_train.shape
```

```
(27034,)
```

```
y_test.shape
```

```
(6759,)

mlp_pipeline.fit(X_train, y_train)

Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('num',
StandardScaler(),
                                                  ['Year', 'Export
Value',
                                                   'food imports in
total '
                                                   'merchandise '
                                                   'exports-percent-3-
year '
                                                   'average',
                                                   'Cropland',

'Consumer_Prices_Food_Indices_2015_100',

'exchange_rate_value',
                                                   'Export Value USD',
                                                   'yield_value',

'export_quantity_tons',

'export_value_lag_1_year',

'export_value_lag_2_years',

'export_value_lag_3_years',

'export_value_moving_avg_3yr']),
                                                 ('cat',

OneHotEncoder(handle_unknown='ignore'),
                                                  ['Area',
'Item_x'])])),
                ('regressor',
                 MLPRegressor(alpha=0.001, hidden_layer_sizes=(60,
40),
                              learning_rate_init=1e-05,
max_iter=100000,
                              random_state=10))])

y_pred = mlp_pipeline.predict(X_test)

# Separate features and target
X = export_sorted_df.drop(['forecast_export_value_crops_3_years'],
axis=1)
y = export_sorted_df['forecast_export_value_crops_3_years']
```

```
# Preprocess categorical columns
categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])


numerical_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())
])


preprocessor = ColumnTransformer(
    transformers=[
        ('cat', categorical_transformer, categorical_columns),
        ('num', numerical_transformer, numerical_columns)
    ])


pipeline = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', MLPRegressor(max_iter=500))  # Increase max_iter to
allow more iterations
])

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

# Define the parameter grid for hyperparameter tuning
param_grid = {
    'regressor__hidden_layer_sizes': [(50,), (100,), (50, 50)],
    'regressor__activation': ['relu', 'tanh'],
    'regressor__learning_rate': ['constant', 'adaptive']
}

# Perform grid search with cross-validation
grid_search = GridSearchCV(pipeline, param_grid, cv=5,
scoring='neg_mean_squared_error')
grid_search.fit(X_train, y_train)


best_model = grid_search.best_estimator_

# Evaluate the model on the test set
y_pred = best_model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
r2 = r2_score(y_test, y_pred)

print("Best Model:")
```

```python
print(best_model)
print("RMSE:", rmse)
print("R-squared:", r2)
```

```
/opt/anaconda3/lib/python3.11/site-packages/sklearn/neural_network/
_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (500) reached and the optimization
hasn't converged yet.
  warnings.warn(
/opt/anaconda3/lib/python3.11/site-packages/sklearn/neural_network/
_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (500) reached and the optimization
hasn't converged yet.
  warnings.warn(
/opt/anaconda3/lib/python3.11/site-packages/sklearn/neural_network/
_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (500) reached and the optimization
hasn't converged yet.
  warnings.warn(
/opt/anaconda3/lib/python3.11/site-packages/sklearn/neural_network/
_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (500) reached and the optimization
hasn't converged yet.
  warnings.warn(
/opt/anaconda3/lib/python3.11/site-packages/sklearn/neural_network/
_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (500) reached and the optimization
hasn't converged yet.
  warnings.warn(
/opt/anaconda3/lib/python3.11/site-packages/sklearn/neural_network/
_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (500) reached and the optimization
hasn't converged yet.
  warnings.warn(
/opt/anaconda3/lib/python3.11/site-packages/sklearn/neural_network/
_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (500) reached and the optimization
hasn't converged yet.
  warnings.warn(
/opt/anaconda3/lib/python3.11/site-packages/sklearn/neural_network/
_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (500) reached and the optimization
hasn't converged yet.
  warnings.warn(
/opt/anaconda3/lib/python3.11/site-packages/sklearn/neural_network/
_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (500) reached and the optimization
hasn't converged yet.
  warnings.warn(
/opt/anaconda3/lib/python3.11/site-packages/sklearn/neural_network/
```

```
_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic
Optimizer: Maximum iterations (500) reached and the optimization
hasn't converged yet.
  warnings.warn(

Best Model:
Pipeline(steps=[('preprocessor',
                 ColumnTransformer(transformers=[('cat',

Pipeline(steps=[('onehot',

OneHotEncoder(handle_unknown='ignore'))]),
                                                   ['Area', 'Item_x']),
                                                  ('num',

Pipeline(steps=[('scaler',

StandardScaler())]),
                                                   ['Year', 'Export
Value',
                                                    'food imports in
total '
                                                    'merchandise '
                                                    'exports-percent-3-
year '
                                                    'average',
                                                    'Cropland',

'Consumer_Prices_Food_Indices_2015_100',

'exchange_rate_value',
                                                    'Export Value USD',
                                                    'yield_value',

'export_quantity_tons',

'export_value_lag_1_year',

'export_value_lag_2_years',

'export_value_lag_3_years',

'export_value_moving_avg_3yr'])])),
                ('regressor',
                 MLPRegressor(hidden_layer_sizes=(50, 50),
max_iter=500))])
RMSE: 3.3567609437788144e+16
R-squared: -0.00015301332388539812

/opt/anaconda3/lib/python3.11/site-packages/sklearn/neural_network/
_multilayer_perceptron.py:686: ConvergenceWarning: Stochastic
```

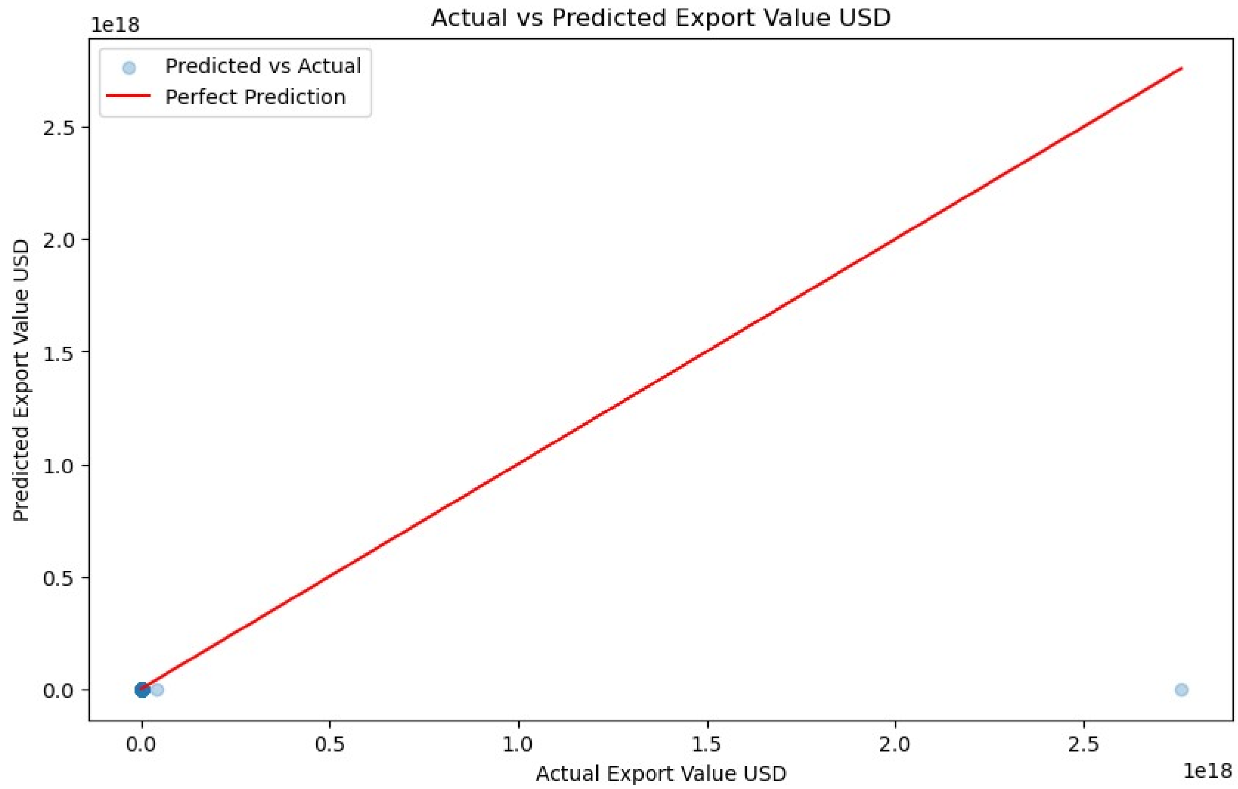
```
Optimizer: Maximum iterations (500) reached and the optimization
hasn't converged yet.
  warnings.warn(
```

```
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Root Mean Squared Error: {rmse}")
```

```
Root Mean Squared Error: 3.3567609437788144e+16
```

```
result_df = X_test.copy()
result_df['Actual Export Value'] = y_test
result_df['Predicted Export Value'] = y_pred
output_columns = ['Area', 'Item_x', 'Year', 'Actual Export Value',
'Predicted Export Value']
final_output_df = result_df[output_columns]
```

```
plt.figure(figsize=(10, 6))  # Set the figure size
plt.scatter(y_test, y_pred, alpha=0.3, label='Predicted vs Actual')  #
Plot actual vs. predicted values
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)],
color='red', label='Perfect Prediction')  # Plot a red line for
perfect predictions
plt.xlabel('Actual Export Value USD')  # Label for x-axis
plt.ylabel('Predicted Export Value USD')  # Label for y-axis
plt.title('Actual vs Predicted Export Value USD')  # Title of the plot
plt.legend()  # Add a legend
plt.show()  # Show the plot
```



```
final_output_df

                   Area                            Item_x  Year  \
29045          Portugal           Dairy Products and Eggs  2018   
9244             Cyprus                          Non-food  2017   
21213             Libya                        Other food  2009   
6373           Cambodia          Cereals and Preparations  2020   
23331         Mauritius                          Non-food  2015   
...                 ...                               ...   ...   
22836             Malta  Fats and Oils (excluding Butter)  2014   
35930       Timor-Leste  Fats and Oils (excluding Butter)  2005   
2564            Bahrain        Meat and Meat Preparations  2006   
15954          Honduras           Dairy Products and Eggs  2015   
13326  French Polynesia        Meat and Meat Preparations  2009   

       Actual Export Value  Predicted Export Value  
29045         3.389595e+08            8.451714e+06  
9244          2.035142e+07            8.305063e+06  
21213         1.313572e+04            5.055258e+06  
6373          1.393500e+12            5.113989e+06  
23331         1.213598e+09            8.178553e+06  
...                    ...                     ...  
22836         0.000000e+00            5.951291e+06  
35930         0.000000e+00            6.110128e+06  
2564          1.759680e+05            1.138114e+07  
```

```
15954         2.588339e+08            7.656804e+06
13326         8.560694e+05            7.908450e+06

[6759 rows x 5 columns]
```

```
final_output_df.to_csv('MLP_Predictions_export_value_13_5_3.csv',
index=False)
print('Predictions with additional details have been saved to
MLP_Predictions_export_value.csv')
```

```
Predictions with additional details have been saved to
MLP_Predictions_export_value.csv
```