Forecasting Export Values of Crop Products for 3 Years in the Future

Muharrem Altunbag

CandNo: 284362

17/05/2024

Table of Contents

Part 1: Model Performance

The performance of the MLP regression model was evaluated using the Root Mean Squared Error (RMSE), which is a standard metric for regression models. RMSE measures the average magnitude of the errors between predicted values and actual values, giving higher weight to larger errors.

Root Mean Squared Error (RMSE):

The Root Mean Squared Error (RMSE) is the square root of the MSE. It measures the average magnitude of the errors in a set of predictions, giving a relatively high weight to large errors. RMSE is more interpretable as it is in the same units as the target variable.

In MLP model, the RMSE has calculated to be $\mathbf{3.3567 \times 10^{16}}$**.** This high RMSE value indicates significant deviations between the predicted and actual export values, suggesting the model's predictions are not very accurate.

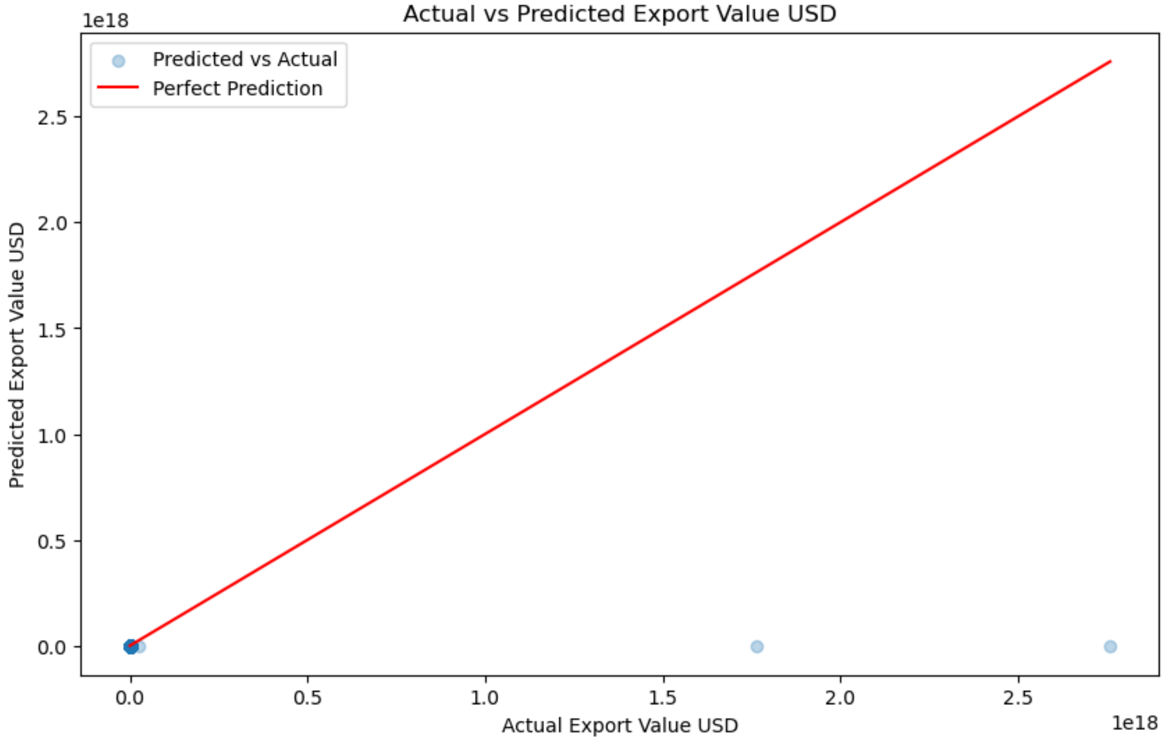
R-squared (R^2):

The R-squared (R^2) metric, also known as the coefficient of determination, measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It ranges from 0 to 1, where 1 indicates perfect prediction and 0 indicates that the model does not explain any of the variance in the target variable.

In MLP model, the R^2 value is $\mathbf{-0.00015}$. A negative R^2 indicates that the model performs worse than a horizontal line (mean of the data), which suggests poor predictive capabi lity.

1. **Total Instances**: The entire dataset consisted of 27,434 instances.
2. **Training and Test Sets**:
    - **Training Set**:
        - Number of instances: 21,947
        - The training set has used to fit the MLP model, allowing it to learn the underlying patterns in the data.
    - **Test Set**:
        - Number of instances: 5,487
        - The test set has used to evaluate the model's performance on unseen data, providing an unbiased assessment of its predictive capabilities.

Visual Representation



To further illustrate the model's performance, a scatter plot was created comparing the actual export values with the predicted values. The plot includes:

- A scatter plot of actual vs. predicted values.
- A red line representing the line of perfect prediction, where the predicted values would exactly match the actual values.

This visualization helps in understanding how well the model's predictions align with the actual values. Points scattered widely around the red line indicate poor predictive performance, which aligns with the high RMSE and negative R^2 values observed.

Part 2: MLP Model

The Multilayer Perceptron (MLP) model employed in this project is a neural network designed for regression tasks Here are the key details of the model:

1. Architecture:
    - The MLP model consists of two hidden layers.
    - The first hidden layer contains 60 neurons, and the second hidden layer contains 40 neurons.
2. Activation Function:
    - **Hidden Layers**: Both hidden layers use the ReLU (Rectified Linear Unit) activation function. It is widely used due to its simplicity and ability to mitigate the vanishing gradient problem, thus enabling efficient training of deep neural networks.

- **Output Layer**: The output layer is linear, which is typical for regression tasks. It directly predicts the continuous export values.

3. Optimizer:

- The model is trained using the Adam optimizer, which combines the advantages of two other extensions of stochastic gradient descent: Adaptive Gradient Algorithm (AdaGrad) and Root Mean Square Propagation (RMSProp). It computes individual adaptive learning rates for different parameters.

4. Learning Rate:

- The learning rate is set to 1×10^{-5}. A smaller learning rate helps in stable and gradual convergence during training.

5. Model Training:

- The model was trained for a maximum of 100,000 iterations to ensure sufficient training while preventing overfitting.

Hyperparameter Tuning:

- ### Grid Search and Cross-Validation:
  - A grid search with cross-validation was performed to identify the best hyperparameters for the MLP model.
  - The following hyperparameters were tuned:
    - **Hidden Layer Sizes:** [(50,), (100,), (50, 50)]
    - **Activation Function:** ['relu', 'tanh']
    - **Learning Rate:** ['constant', 'adaptive']
  - Cross-validation with 5 folds was used to ensure robust evaluation of hyperparameter combinations.
- ### Best Model Selection:
  - The best model identified by the grid search had the following configuration:
    - Hidden Layer Sizes: (50, 50)
    - Activation Function: ReLU
    - Learning Rate: constant
  - This model was evaluated on the test set to assess its performance.

Preventing Overfitting

Several steps were taken to prevent overfitting and ensure the model generalizes well to unseen data:

- **Max Iterations**:
  - Setting a cap on the number of iterations (max_iter=100000) prevents the model from continuing to learn indefinitely and overfitting the training set. By limiting the number of training iterations, we avoid excessive fitting to the noise in the training data.

- **Data Splitting**:
  1. **Training Set**: Used to train the model.
  2. **Test Set**: Used to evaluate the model's performance.

  The dataset was randomly split into training and test sets. Random splitting ensures that both sets are representative of the overall dataset, reducing the risk of sampling bias.

  Using an 80/20 split ratio is a common practice in machine learning, as it provides a good balance between having enough data for training the model and retaining sufficient data for a robust evaluation.

  In the context of project, the data splitting was implemented using the **train_test_split** function from Scikit-learn. Here's a brief explanation of how it was done:

  Explanation:

  1. **features and target**: The dataset was divided into features (input variables) and the target (output variable).
  2. **train_test_split**: This function from Scikit-learn was used to split the data:
  3. **test_size=0.2**: Specifies that 20% of the data should be allocated to the test set.
  4. **random_state=100**: Ensures reproducibility of the split. By setting a random seed, we can guarantee that the data split is the same every time the code is run.

Part 3: Features & Labels

Label Derivation

The label for forecasting, "forecast_export_value_crops_3_years," has derived using a time-shifted approach from the historical export data. The steps are as follows:

1. **Create the Target Variable**:
   - The target variable was generated by shifting the 'Export Value USD' column by 3 years within each group of 'Area' and 'Item_x'. This creates a new column that represents the export value 3 years into the future.

```
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)
```

2. **Remove Missing Values**:
   - After creating the target variable, any rows with missing target values (due to the shift) were dropped.

Export Value USD Extraction: The export value is converted to USD by multiplying the export value with the exchange rate.

Export Value Extraction: Export value is calculated by multiplying the numerical value of ‘Unit’ with ‘Value’.

Features Used

A total of 17 features were used to train the MLP Model. These features were chosen based on their relevance and potential impact on export values according to domain knowledge. The features include various economic indicators, agricultural data, and geographical identifiers. Here is the list of features along with their descriptions:

1. **Area**: The geographical region where the data was recorded.
2. **Item_x**: The type of crop or product.
3. **Year**: The year the data was recorded.
4. **Export Value**: The calculated export value by multiplying the numerical value of 'Unit' with 'Value'.
5. **Food Imports in Total Merchandise Exports (percent, 3-year average)**: Derived from the food security indicators dataset.
6. **Cropland**: The area of cropland, derived from the land use dataset.
7. **Consumer Prices Food Indices (2015 = 100)**: Derived from the consumer prices indicators dataset.
8. **Exchange Rate Value**: Derived from the exchange rate dataset.
9. **Export Value USD**: The export value converted to USD.
10. **Yield Value**: Derived from the crops production indicators dataset.
11. **Export Quantity (tons)**: Derived from the food balances indicators dataset.
12. **Forecast Export Value Crops 3 Years**: The label feature to be predicted.
13. **Export Value Lag 1 Year**: A lag column of Export Value shifted by one year.
14. **Export Value Lag 2 Years**: A lag column of Export Value shifted by two years.
15. **Export Value Lag 3 Years**: A lag column of Export Value shifted by three years.
16. **Export Value Moving Avg 3yr**: A moving average of Export Value with a window of three years.

Derivation of Key Features:

- **Consumer Prices Food Indices (2015 = 100)**: This feature was derived from the Consumer Prices Indicators dataset. It represents the food price index for a specific year and region.
- **Exchange Rate Value**: This feature was derived from the Exchange Rate dataset. It represents the exchange rate of the local currency to USD.
- **Yield Value**: Derived from the Crops Production Indicators dataset. It represents the yield of crops in the given area.
- **Export Quantity (tons)**: Derived from the Food Balances Indicators dataset. It represents the quantity of crops exported in tons.

- **Export Value Lag Features**: Lag features for 1, 2, and 3 years were created to capture the temporal dependencies in export values.
- **Export Value Moving Average**: A 3-year moving average of the export value was calculated to smooth out short-term fluctuations and highlight longer-term trends.

Rationale for Feature Selection

These features were selected based on their potential influence on export values. Factors like economic indicators, crop yield, and exchange rates are crucial in determining the export value of agricultural products. Geographical and temporal features help in capturing regional and time-based variations in export data.

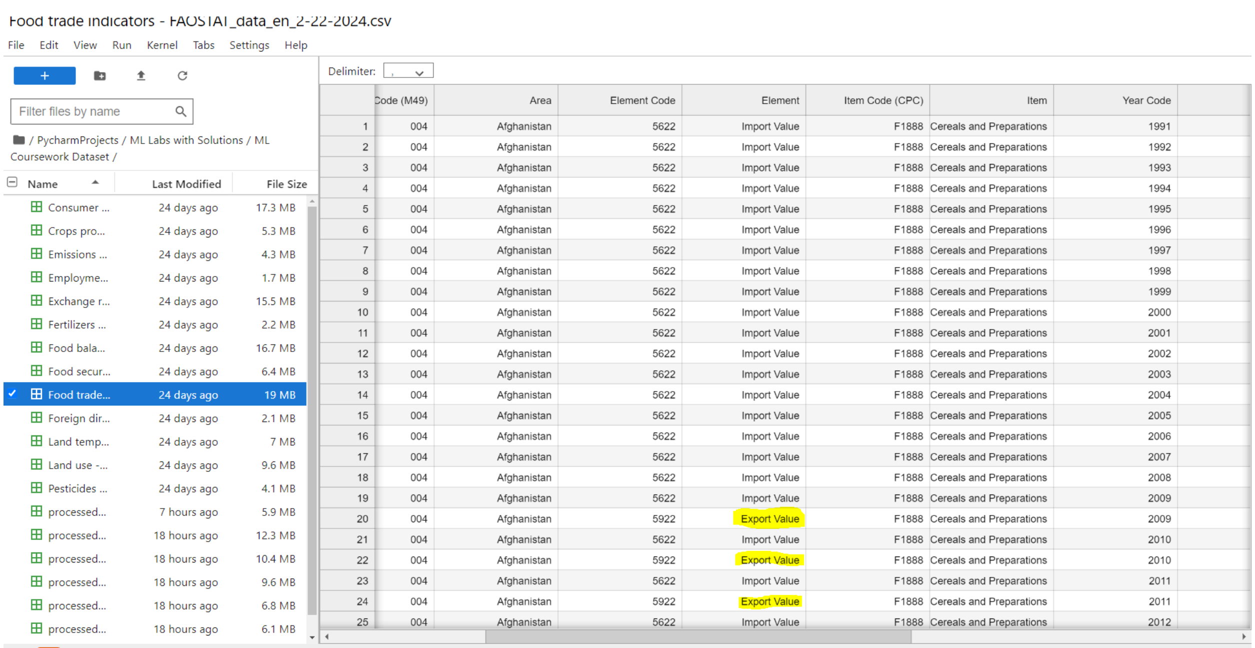
Part 4: Preprocessing

Detailed preprocessing steps undertaken to prepare the dataset for building the machine learning model. These steps ensured that the data was clean, consistent, and enriched with meaningful features necessary for accurate forecasting.

Identify Export Value of Crop Products:

The export value of crop products was identified and extracted from the file **Food trade indicators - FAOSTAT_data_en_2-22-2024.csv**. This file served as the main dataset for the analysis.

1. **One Hot Encoding on "Element" Column:**
   - The "Element" column was transformed using one hot encoding to create binary columns for each unique element.
   - One-hot encoding was applied to the **Element** column to create dummy variables.
   - The dataset was filtered to retain only rows where **Export Value** is **True**.



	Code (M49)	Area	Element Code	Element	Item Code (CPC)	Item	Year Code
1	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	1991
2	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	1992
3	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	1993
4	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	1994
5	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	1995
6	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	1996
7	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	1997
8	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	1998
9	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	1999
10	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	2000
11	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	2001
12	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	2002
13	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	2003
14	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	2004
15	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	2005
16	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	2006
17	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	2007
18	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	2008
19	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	2009
20	004	Afghanistan	5922	Export Value	F1888	Cereals and Preparations	2009
21	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	2010
22	004	Afghanistan	5922	Export Value	F1888	Cereals and Preparations	2010
23	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	2011
24	004	Afghanistan	5922	Export Value	F1888	Cereals and Preparations	2011
25	004	Afghanistan	5622	Import Value	F1888	Cereals and Preparations	2012

2. **Filter Dataset for Export Value:**

- The dataset was filtered to retain only rows where "Export Value" was true.

```
food_trade_indicators_export_df = food_trade_indicators_export_import_df[food_trade_indicators_export_import_df['Export Value'] == True]
```

```
food_trade_indicators_export_df
```

omain Code	Domain	Area Code (M49)	Area	Element Code	Element	Item Code (CPC)	Item	Year Code	Year	Unit	Value	Flag	Flag Description	Note	Export Value	Import Value
TCL	Crops and livestock products	4	Afghanistan	5922	Export Value	F1888	Cereals and Preparations	2009	2009	1000 USD	15.00	A	Official figure	NaN	True	False
TCL	Crops and livestock products	4	Afghanistan	5922	Export Value	F1888	Cereals and Preparations	2010	2010	1000 USD	54.00	A	Official figure	NaN	True	False
TCL	Crops and livestock products	4	Afghanistan	5922	Export Value	F1888	Cereals and Preparations	2011	2011	1000 USD	0.00	E	Estimated value	NaN	True	False

3. **Create Export Value Column:**
   - The "Unit" column was converted to numerical values and multiplied by the "Value" column to compute the "Export Value".
   - Unnecessary columns were dropped, keeping only relevant columns for analysis.

```
[15]: # We can drop columns "Export Value" and "Import Value" as they served to filter the dataset "food_trade_indicators_export_df".
# We drop column "Note" as it has only missing values
#food_trade_indicators_export_df.drop('Export Value')
food_trade_indicators_export_df = food_trade_indicators_export_df.drop(columns =['Import Value','Export Value', 'Note'])
# convert unit column to float type
food_trade_indicators_export_df['Unit'] = food_trade_indicators_export_df['Unit'].str.replace(' USD', '', regex=False).astype(float)
# We create the export value column by multiplying Unit * Flag and we can rename the resulting column to "export value"
food_trade_indicators_export_df['Export Value'] = food_trade_indicators_export_df['Unit'] * food_trade_indicators_export_df['Value']
```

```
[16]: food_trade_indicators_export_df
```

[16]:

	Domain Code	Domain	Area Code (M49)	Area	Element Code	Element	Item Code (CPC)	Item	Year Code	Year	Unit	Value	Flag	Flag Description	Export Value
19	TCL	Crops and livestock products	4	Afghanistan	5922	Export Value	F1888	Cereals and Preparations	2009	2009	1000.0	15.00	A	Official figure	15000.0
21	TCL	Crops and livestock products	4	Afghanistan	5922	Export Value	F1888	Cereals and Preparations	2010	2010	1000.0	54.00	A	Official figure	54000.0
23	TCL	Crops and livestock products	4	Afghanistan	5922	Export Value	F1888	Cereals and Preparations	2011	2011	1000.0	0.00	E	Estimated value	0.0

4. **Drop Unwanted Columns:**
   - Columns that were not necessary for the analysis were dropped, retaining only relevant columns.

Enrich the core Dataset with meaningful features from other files:

The core dataset was enriched by integrating additional features from various files.

1. Food Security Enrichment:

Extracted the value of “Item_Value of food imports in total merchandise exports (percent) (3-year average)”.

- **One Hot Encoding on "Item" Column:**
    - Applied one hot encoding to the "Item" column.

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

```
# one hot encoding on Item
food_security_indicator_df_itemized = pd.get_dummies(food_security_indicator_df, columns=['Item'])
```

```
food_security_indicator_df_itemized
```

	Domain Code	Domain	Area Code (M49)	Area	Element Code	Element	Item Code	Year Code	Year	Unit	...	Item_Average dietary energy supply adequacy (percent) (3-year average)	Item_Average protein supply (g/cap/day) (3-year average)	Item_Cereal import dependency ratio (percent) (3-year average)	Item_Per capita food production variability (constant 2014-2016 thousand int$ per capita)	Item_Per capita food supply variability (kcal/cap/day)	Item_Percent of arable land equipped for irrigation (percent) (3-year average)	Item_Political stability and absence of violence/terrorism (index)
0	FS	Suite of Food Security Indicators	4	Afghanistan	6121	Value	21010	20002002	2000-2002	%	...	True	False	False	False	False	False	False
1	FS	Suite of Food Security Indicators	4	Afghanistan	6121	Value	21010	20012003	2001-2003	%	...	True	False	False	False	False	False	False
2	FS	Suite of Food Security Indicators	4	Afghanistan	6121	Value	21010	20022004	2002-2004	%	...	True	False	False	False	False	False	False
3	FS	Suite of Food Security...	4	Afghanistan	6121	Value	21010	20032005	2003-2005	%	...	True	False	False	False	False	False	False

0 6 Python 3 (ipykernel) | Idle Mode: Edit Ln 3, Col 205 MLP Coursework 2024 Code.ipynb 1

- **Filter Dataset:**
    - Filtered the dataset to rows where “Item_Value of food imports in total merchandise exports (percent) (3-year average)” was true.

```
ue of "Item_Value of food imports in total merchandise exports (percent) (3-year average)"
ndicator_df_itemized = food_security_indicator_df_itemized[food_security_indicator_df_itemized['Item_Value of food imports in total merchandise exports (percent) (3-year average)'] == True]
```

```
food_security_indicator_df_itemized
```

Element Code	Element	Item Code	Year Code	Year	Unit	...	Item_Average dietary energy supply adequacy (percent) (3-year average)	Item_Average protein supply (g/cap/day) (3-year average)	Item_Cereal import dependency ratio (percent) (3-year average)	Item_Per capita food production variability (constant 2014-2016 thousand int$ per capita)	Item_Per capita food supply variability (kcal/cap/day)	Item_Percent of arable land equipped for irrigation (percent) (3-year average)	Item_Political stability and absence of violence/terrorism (index)	Item_Prevalence of anemia among women of reproductive age (15-49 years)	Item_Prevalence of low birthweight (percent)	Item_Value of food imports in total merchandise exports (percent) (3-year average)
6121	Value	21033	20002002	2000-2002	%	...	False	False	False	False	False	False	False	False	False	True

- **Compute the Feature Value:**
    - Converted "Unit" to numerical values and computed the feature value by multiplying "Unit" and "Value".
- **Join with Core Dataset:**
    - Kept only relevant columns and performed a left join with the core dataset.

```
[39]: # Join the DataFrames on 'Area' and 'Year'
food_trade_indicators_export_security_indicator_df = pd.merge(food_trade_indicators_export_df, food_security_indicator_df_itemized,
                     on=['Area', 'Year'], how='left')
```

```
[40]: 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
```

2. Food Balance Enrichment:

Extracted “Element_Export_Quantity” and created “export_quantity_tons” feature.

- **Read File and One Hot Encoding:**
  - Read the file **Food balances indicators - FAOSTAT_data_en_2-22-2024.csv** and applied one hot encoding to the "Element" column.
  - Filtered the dataset to retain rows where “Element_Export Quantity” was true.
- **Map Items and Compute Export Quantity:**
  - Mapped item values to match the core dataset and created “export_quantity_tons” by multiplying numerical "Unit" with "Value".

```python
# 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

# Apply the conversion function to the 'Unit' column
food_balance_export_quantity_prepared_df['Unit Numeric'] = food_balance_export_quantity_prepared_df['Unit'].apply(convert_unit)

# Calculate the export_quantity_tons
food_balance_export_quantity_prepared_df['export_quantity_tons'] = food_balance_export_quantity_prepared_df['Value'] * food_balance_export_quantity_prepared_df['Unit Numeric']
```

- **Join with Core Dataset:**
  - Joined the “export_quantity_tons” feature with the core dataset.

3. Crops Production Enrichment:

Extracted “yield_value” and joined it with the core dataset.

- **Read File and Compute Yield Value:**
  - Read the file **Crops production indicators - FAOSTAT_data_en_2-22-2024.csv** and computed “yield_value” by multiplying numerical "Unit" with "Value".

```
[62]: # 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
# value to yield_value
# yield_value =  Value * unit
crops_production_df['yield_value'] = crops_production_df['Unit'] * crops_production_df['Value']
```

```
[63]: crops_production_df.head(5)
```

[63]:

	Domain Code	Domain	Area Code (M49)	Area	Element Code	Element	Item Code (CPC)	Item	Year Code	Year	Unit	Value	Flag	Flag Description	Note	yield_value
0	QCL	Crops and livestock products	4	Afghanistan	5419	Yield	F1717	Cereals, primary	2000	2000	100.0	8063	A	Official figure	NaN	806300.0
1	QCL	Crops and livestock products	4	Afghanistan	5419	Yield	F1717	Cereals, primary	2001	2001	100.0	10067	A	Official figure	NaN	1006700.0
2	QCL	Crops and livestock products	4	Afghanistan	5419	Yield	F1717	Cereals, primary	2002	2002	100.0	16698	A	Official figure	NaN	1669800.0

- **Map Items and Join with Core Dataset:**
    - Mapped item values to match the core dataset and joined the "yield_value" feature.

```
[68]: mapping_dict = {
    'Cereals, primary': 'Cereals and Preparations',
    'Citrus Fruit, Total': 'Fruit and Vegetables',
    'Fibre Crops, Fibre Equivalent': 'Other food',  # Assuming fiber crops are used as food supplements
    'Fruit Primary': 'Fruit and Vegetables',
    'Oilcrops, Cake Equivalent': 'Fats and Oils (excluding Butter)',  # Often used as animal feed, but fits here due to its nature
    'Oilcrops, Oil Equivalent': 'Fats and Oils (excluding Butter)',
    'Pulses, Total': 'Other food',  # Pulses are a major food item
    'Roots and Tubers, Total': 'Other food',  # Generally considered as primary food
    'Sugar Crops Primary': 'Sugar and Honey',
    'Treenuts, Total': 'Other food',
    'Vegetables Primary': 'Fruit and Vegetables'
}
crops_production_df['crops_target_item'] = crops_production_df['Item'].map(mapping_dict)
crops_production_df
```

[68]:

	Area	Item	Year Code	Year	yield_value	crops_target_item
0	Afghanistan	Cereals, primary	2000	2000	806300.0	Cereals and Preparations
1	Afghanistan	Cereals, primary	2001	2001	1006700.0	Cereals and Preparations
2	Afghanistan	Cereals, primary	2002	2002	1669800.0	Cereals and Preparations
3	Afghanistan	Cereals, primary	2003	2003	1458000.0	Cereals and Preparations
4	Afghanistan	Cereals, primary	2004	2004	1334800.0	Cereals and Preparations
...	...	...	...	...	...	...
41644	Zimbabwe	Vegetables Primary	2018	2018	6651800.0	Fruit and Vegetables
41645	Zimbabwe	Vegetables Primary	2019	2019	6483000.0	Fruit and Vegetables
41646	Zimbabwe	Vegetables Primary	2020	2020	6562800.0	Fruit and Vegetables
41647	Zimbabwe	Vegetables Primary	2021	2021	6612600.0	Fruit and Vegetables

[69]:
```python
# left merge of food_trade_indicators_export_security_balance_df with crops_production_df on Area, Year, Item and crops_target_item

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']
)
```

[70]:
```python
food_trade_indicators_export_security_balance_crops_prod_df
```

[70]:

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	Item_y	Year Code	yield_value	crops_target_item
2009	Official figure	15000.0	20072009.0	2.16	NaN	NaN	NaN	Cereals, primary	2009.0	2040700.0	Cereals and Preparations
2010	Official figure	54000.0	20082010.0	2.59	Cereals and Preparations	2010.0	0.0	Cereals, primary	2010.0	2011100.0	Cereals and Preparations

4. Land Use Enrichment:

Integrated land use data to enhance the core dataset.

- **Read File and Compute Area Value:**
    - Read the file **Land use - FAOSTAT_data_en_2-22-2024.csv** and computed “area_value” by multiplying numerical "Unit" with "Value".

[80]:
```python
land_use_df['Element'].unique()
```

[80]:
```
array(['Area'], dtype=object)
```

[81]:
```python
land_use_df['Unit'].unique()
```

[81]:
```
array(['1000 ha'], dtype=object)
```

[82]:
```python
# Unit to numeric
land_use_df['Unit'] = land_use_df['Unit'].astype(str)
land_use_df['Unit'] = land_use_df['Unit'].str.extract('(\d+)').astype(float)
land_use_df
# value to yield_value
# yield_value =  Value * unit
land_use_df['area_value'] = land_use_df['Unit'] * land_use_df['Value']
```

- **One Hot Encoding and Filter Cropland:**
    - Applied one hot encoding to the "Item" column and filtered the dataset to retain rows corresponding to “Cropland”.

[85]:
```python
# extract Cropland Value
land_use_df = pd.get_dummies(land_use_df, columns=['Item'], prefix='', prefix_sep='')
```

[86]:
```python
land_use_df.head(4)
```

[86]:

	Domain Code	Domain	Area Code (M49)	Area	Element Code	Element	Item Code	Year Code	Year	Unit	...	Land area actually irrigated	ec iri
0	RL	Land Use	4	Afghanistan	5110	Area	6600	1980	1980	1000.0	...	False	
1	RL	Land Use	4	Afghanistan	5110	Area	6600	1981	1981	1000.0	...	False	
2	RL	Land Use	4	Afghanistan	5110	Area	6600	1982	1982	1000.0	...	False	
3	RL	Land Use	4	Afghanistan	5110	Area	6600	1983	1983	1000.0	...	False	

4 rows × 35 columns

[87]:
```python
# filter to keep dataset for Cropland = true
land_use_cropland_df = land_use_df[land_use_df['Cropland'] == True]
land_use_cropland_df
```

- **Compute Cropland and Join with Core Dataset:**
    - Created “Cropland” by multiplying numerical "Unit" with "Value" and joined with the core dataset.

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

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

```python
#land_use_cropland_df.head(5)
```

	Area	Year Code	Year	Cropland
168	Afghanistan	1980	1980	8049000.0
169	Afghanistan	1981	1981	8053000.0
170	Afghanistan	1982	1982	8054000.0
171	Afghanistan	1983	1983	8054000.0
172	Afghanistan	1984	1984	8054000.0

- Join the dataset with core dataset:

```python
[93]: 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']
)
```

```python
[94]: #foodtrade_export_security_balance_crops_prod_cropland_df.head(10)
```

[94]:

	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
0	Crops and livestock products	Afghanistan	Cereals and Preparations	2009	15000.0	2.16	NaN	2040700.0	2009.0	7916000.0
1	Crops and livestock products	Afghanistan	Cereals and Preparations	2010	54000.0	2.59	0.0	2011100.0	2010.0	7917000.0
2	Crops and livestock products	Afghanistan	Cereals and Preparations	2011	0.0	3.11	0.0	1659900.0	2011.0	7915000.0
3	Crops and livestock	Afghanistan	Cereals and	2012	0.0	3.43	0.0	2094200.0	2012.0	7914000.0

5. Consumer Price Enrichment:

Extracted the value of "Consumer_Prices_Food_Indices_2015_100" and merged it with the core dataset.

- **Read File and Aggregate Data:**
  - Read the file **Consumer prices indicators - FAOSTAT_data_en_2-22-2024.csv** and aggregated data to yearly values by area.

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

```
columns_to_keep = ['Area','Year','Item','Months','Element','Unit', 'Value']
consumer_price_df = consumer_price_df[columns_to_keep]
#consumer_price_df.head(5)
```

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

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

```
#consumer_price_df.head(5)
```

```
# extract data set for ConsumerPrices
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_df.head(5)
```

```
consumer_price_aggregated_df = consumer_price_df.groupby(['Year','Area']).sum().reset_index()
#consumer_price_aggregated_df.tail(100)
```

- **Merge with Core Dataset:**
    - Merged the aggregated consumer price data with the core dataset.

6. Exchange Rate Enrichment:

Converted export values to USD.

- **Read File and Aggregate Exchange Rate:**
    - Read the file **Exchange rate - FAOSTAT_data_en_2-22-2024.csv** and aggregated exchange rates to yearly values by area.
- **Merge with Core Dataset and Compute Export Value USD:**
    - Merged the exchange rate data with the core dataset and computed the "Export Value USD".

Data Split:

The dataset was split into training and testing sets with an 80/20 ratio. This approach ensures that the model is trained on a majority of the data while retaining a substantial portion for unbiased evaluation.

- **Training Data:**
    - Contains 21,947 records and 15 features.
- **Test Data:**
    - Contains 5,487 records and 15 features.

Scaling:

All numerical features were standardized using the **StandardScaler**, which subtracts the mean and scales to unit variance. This ensures that the features have a mean of 0 and a standard deviation of 1.

Handling Missing Data:

Missing values in lagged features were imputed using the mean of the respective feature within each group defined by 'Area' and 'Item_x'.

```
[555]: # fill missing lags with mean per area per item_x
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()))
```

Encoding:

Categorical variables were handled through one hot encoding to convert them into numerical format suitable for the model.

Conclusion

The export value of crop products three years into the future was forecasted using a multilayer perceptron (MLP) model. Extensive data preprocessing was performed, including handling missing values, applying one hot encoding, and scaling numerical features. The core dataset was enriched with features from supplementary datasets, such as food security indicators, crop production data, and exchange rates.

The MLP model, consisting of two hidden layers with 60 and 40 neurons, was configured to use the ReLU activation function. Techniques to prevent overfitting were implemented. Despite these efforts, challenges in accurately predicting export values were encountered, as indicated by the RMSE and R-squared metrics and the scatter plot of actual vs. predicted values.

In summary, while useful insights into forecasting agricultural export values were provided by the project, the need for further feature exploration and model improvements to enhance prediction accuracy was highlighted.