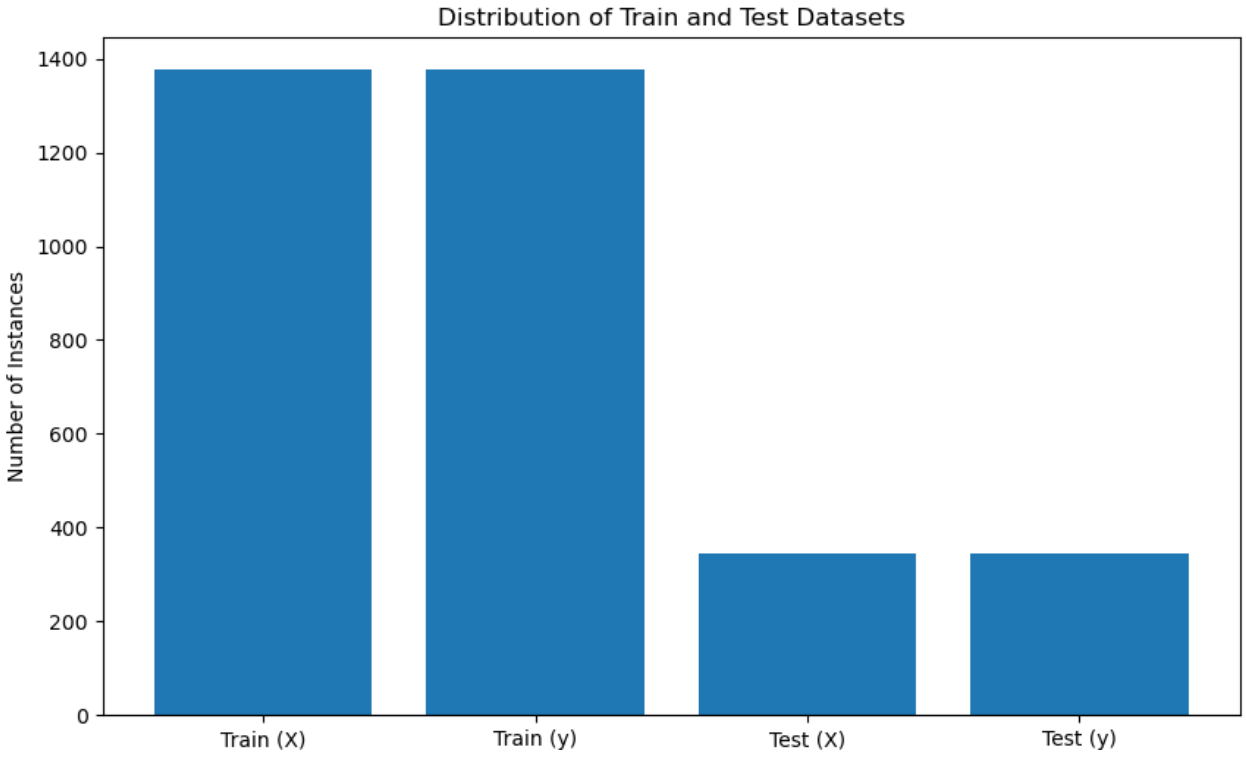
Machine Learning

Name and Surname: Mert OLCAMAN

May, 2024

Contents

1. PERFORMANCE



Metrics to measure the performance of the model:

- **Mean Sqared Error:** Average squared difference between actual and predicted labels

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

- **Root Mean Square Error:**

$$RMSE = \sqrt{MSE}$$

- **Mean Absolute Error (MAE):**

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

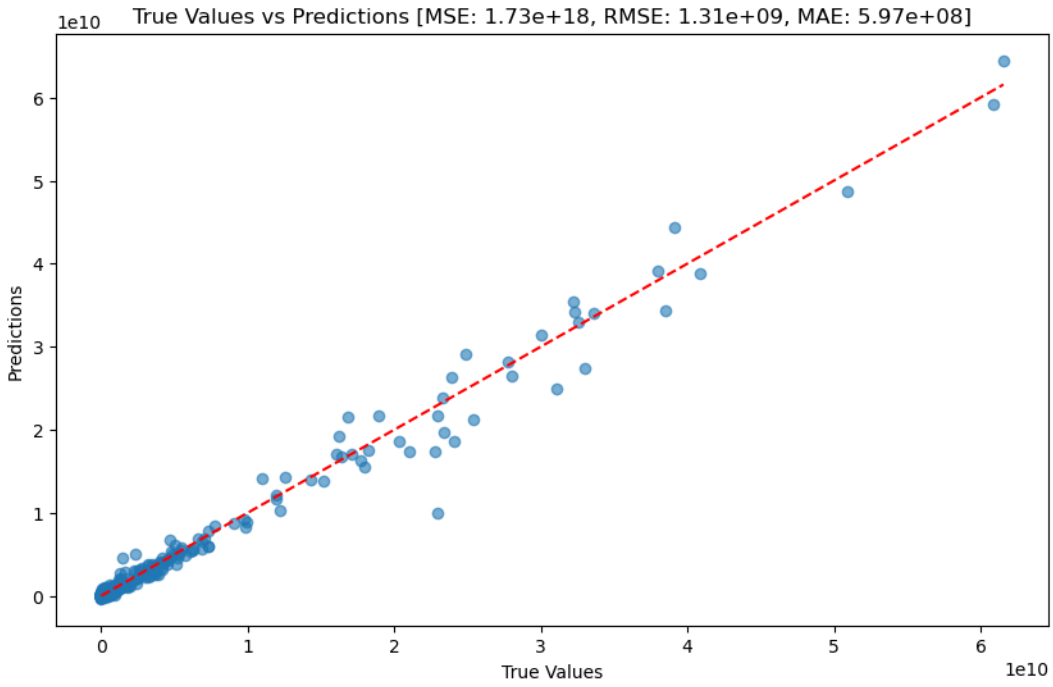
Before Fine-Tuning:

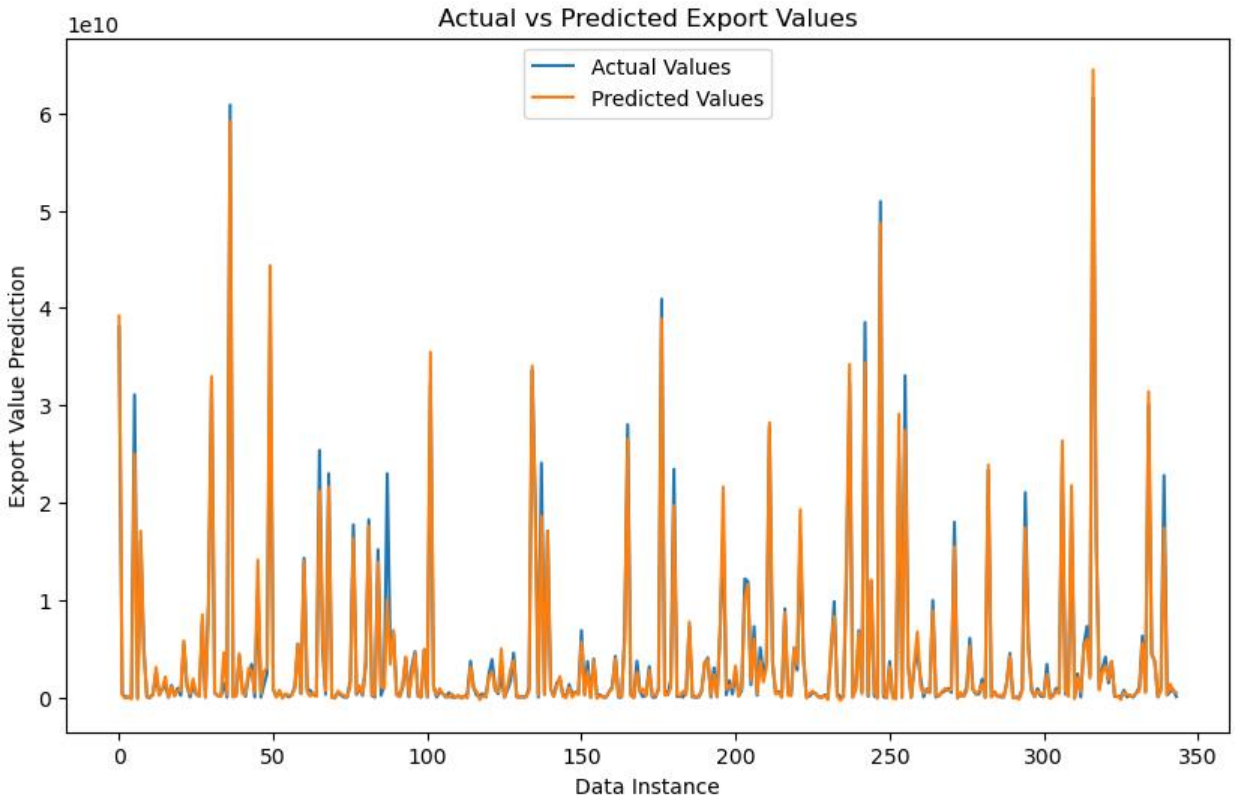
learning rate	0.001
activation function	relu
loss function	mean squared error
epochs	100
batch size	32
optimizer	Adam
activation	ReLU
hidden layer number	2

Mean Squared Error: 1.7263661445643807e+18

Root Mean Squared Error: 1313912533.0722668

Mean Absolute Error: 597358726.2049419





I changed the model parameters to find the best ones.

Learning rate: [1e-6, 1e-5, 1e-4, 1e-3, 1e-2, 1e-1]

The learning rate determines how quickly or slowly a model updates its weights during training.

- small learning rate: longer time to the point supposed to reach, many epoch is required.
- bigger learing rate: preventing to reach the optimal point, the model is never stabilized.

Epoch: [10, 25, 50, 100, 200, 250, 500, 1000]

Epoch indicates how many times the training dataset is processed.

- small epochs: it causes underfitting, the model doesn't learn
- bigger epochs: it causes overfitting, the model learns much on training data, while it performs poor on new dataset

Batch size: [4,8,16,32,64,128,256]

It determines the number of samples

- smaller batch size: More updates per epoch, better generalization but it takes much time.
- bigger batch size: Less updates per epoch, takes less time, but expensive in terms of memory. Also, it might overfit.

Dropout rates: [0.1, 0.2, 0.3]

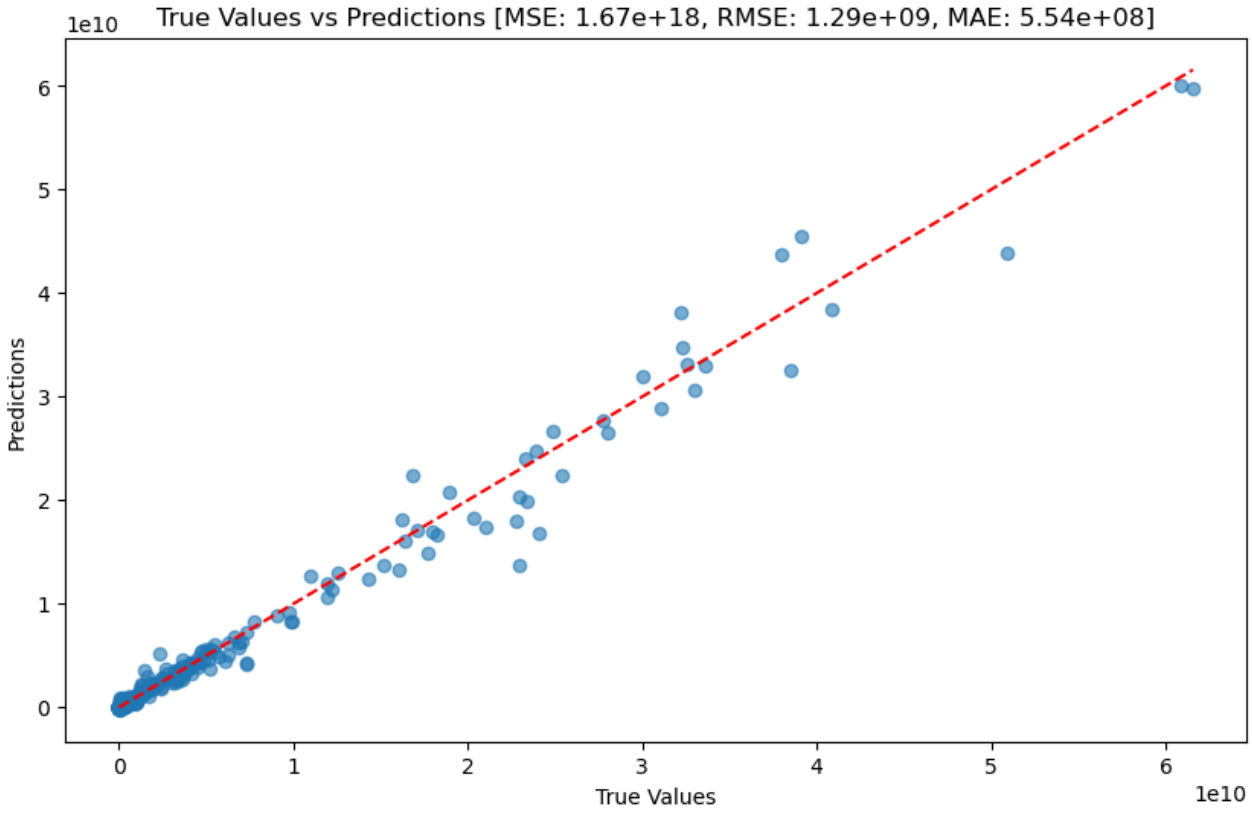
Final Model:

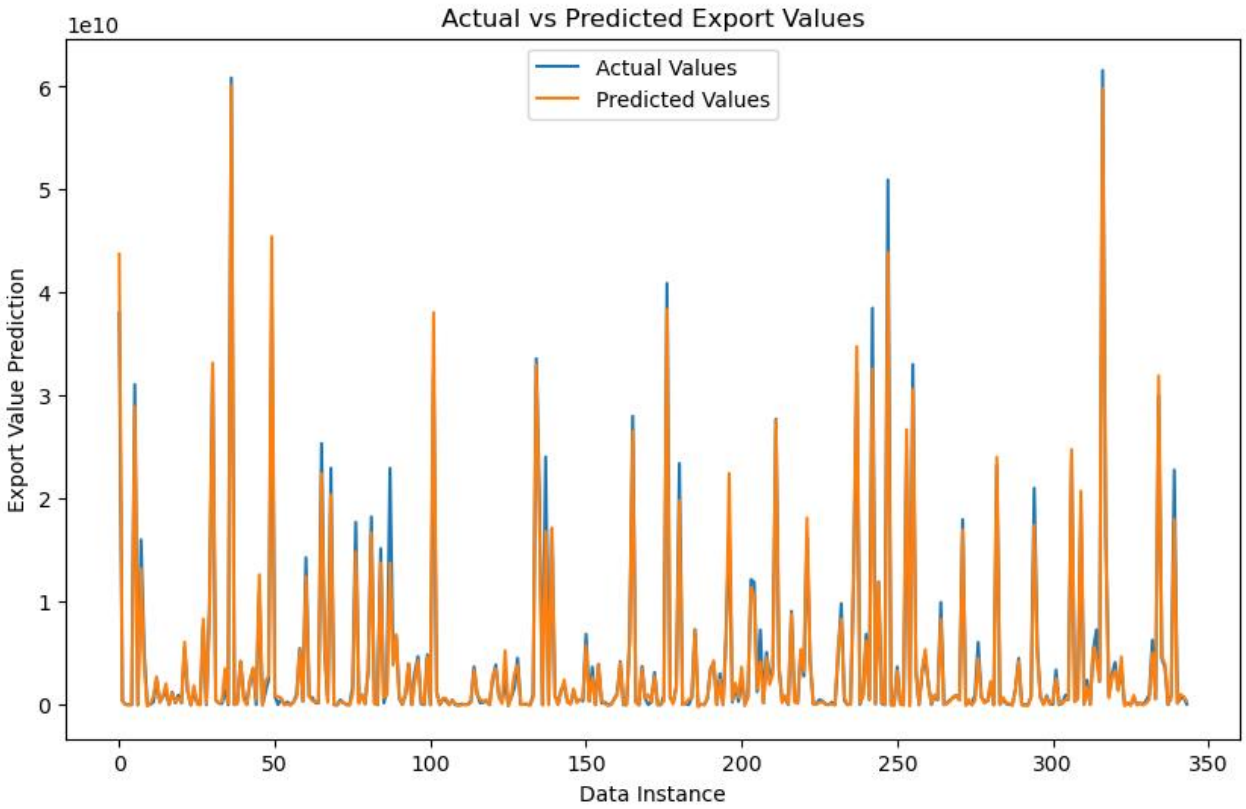
activation function	relu
learning rate	0.005
loss function	mean squared error
epochs	500
batch size	32
optimizer	Adam
activation	ReLU
hidden layer number	3

Mean Squared Error: 1.6696621763376858e+18

Root Mean Squared Error: 1292154083.8219278

Mean Absolute Error: 554073122.6017442





2. MLP MODEL

Input Layer: The input dimension is determined by the number of features in the training dataset

Hidden Layers:

- The first hidden layer has 64 units and uses the ReLU activation function.

$$hidden\ layer_1 = ReLU(W_1 x\ input + b_1)$$

- The second hidden layer has 32 units and also uses the ReLU activation function.

$$hidden\ layer_2 = ReLU(W_2 x\ hidden\ layer_1 + b_2)$$

- A Dropout layer with a rate of 0.1 is applied after the second hidden layer to prevent overfitting.

$$dropout\ layer = Dropout\ (hidden\ layer_2, 0.1)$$

Output Layer: The output layer has 1 unit and uses a linear activation function.

$$output = W_3 x\ dropout\ layer + b_3$$

Loss Function: Mean squared error (MSE) was used in training.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2$$

Preventing Overfitting:

- Dropout Layer: this randomly drops 10% of the neurons in the layer during the training, which helps to prevent the model overfits.
- EarlyStopping: it monitors the validation loss during the training. The patince was taken as 10, which means that if the validation loss isn't imroved, then the training process is stopped.
- Validation Split: 20% of the data was randomly taken as validation. Therefore, the model performance was checked by using unseen data.

3. FEATURES & LABELS

There were common unnecessary columns in all such as "Domain Code", "Area Code", "Year Code", "Item Code", "Months Code", "Element Code", "Flag", "Flag Description", "Note". These kinds of columns were excluded from all.

The values in exchange rate was excluded.

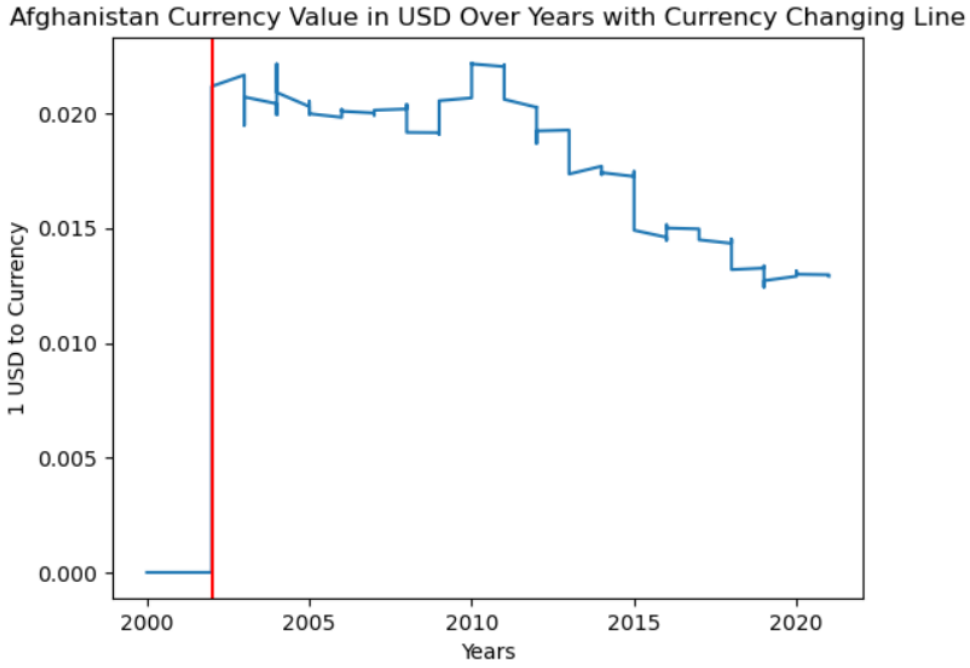


Figure 1: Afghanistan Currency in USD over Years with Currency Changing Line

As it can be seen in the ***Figure***, after around 2002, there was a huge increase in the value of the money. It's probably because some digits were excluded from the currency. On the other hand, other prices were indicated as USD. Therefore, excluding exchange rates wouldn't affect the analysis.

Other values and how they were derived can be seen in ***Table.***

Feature Name	Table Name	Explanation
Area	output_y	Area was the common column in all. I took the areas which exist in the output data frame.

Year	output_y	The common minimum and maximum years were found as 2010 and 2022 respectively. Therefore, 2022 was the maximum year which could be predicted. Because of that, the data was filtered until 2019.
value_CPindex	Consumer Prices Indicators	The data was arranged by taking 2015 as a base year by using the formulation below **REF**. $$\text{Monthly CPI in month t with base year 2015} = \frac{\text{Monthly CPI in month t}}{\text{geometric mean of the Monthly CPI in all months of 2015}} \text{ x } 100$$ $$\text{CPI}_{t,y,base=2015} = \frac{\text{CPI}_{t,y}}{\sqrt[n]{\prod_{t=1}^{n} \text{CPI}_{t,y=2015}}}$$ where $CPI_{t,y}$ = CPI for month t and year y and t = 1,2, … ,12 Since the geometric mean of all the months of 2015 is pretty close to 100 for each country (there were 0.00000001 changes for some countries), the value which was obtained from the formula was the same. On the other hand, each record was indicated by both percentage unit and unitless. After checking the values for the same area in 2 sequential years, I noticed that the percentage value could be found from the unitless values. If the area of Türkiye and the years of 2000 and 2001 on January is taken as an example: $$\text{Percentage } [\%] = \frac{\text{Value}_{\text{Türkiye,2001,January}}}{\text{Value}_{\text{Türkiye,2000,January}}} \text{ x } 100$$ Because of that, the percentage values were excluded. Some countries didn't have values for previous years. I found them by interpolating backwards.
crop_products_kg/m2	Crops Production Indicators	Units were converted from 100g/ha to kg/m^2. Some countries were combined such as Sudan. Domain changed as crop products. Because all of them was crop products. Also, I changed the value title to "crop_products_kg/m2" to understand it easier what that value means while checking later.

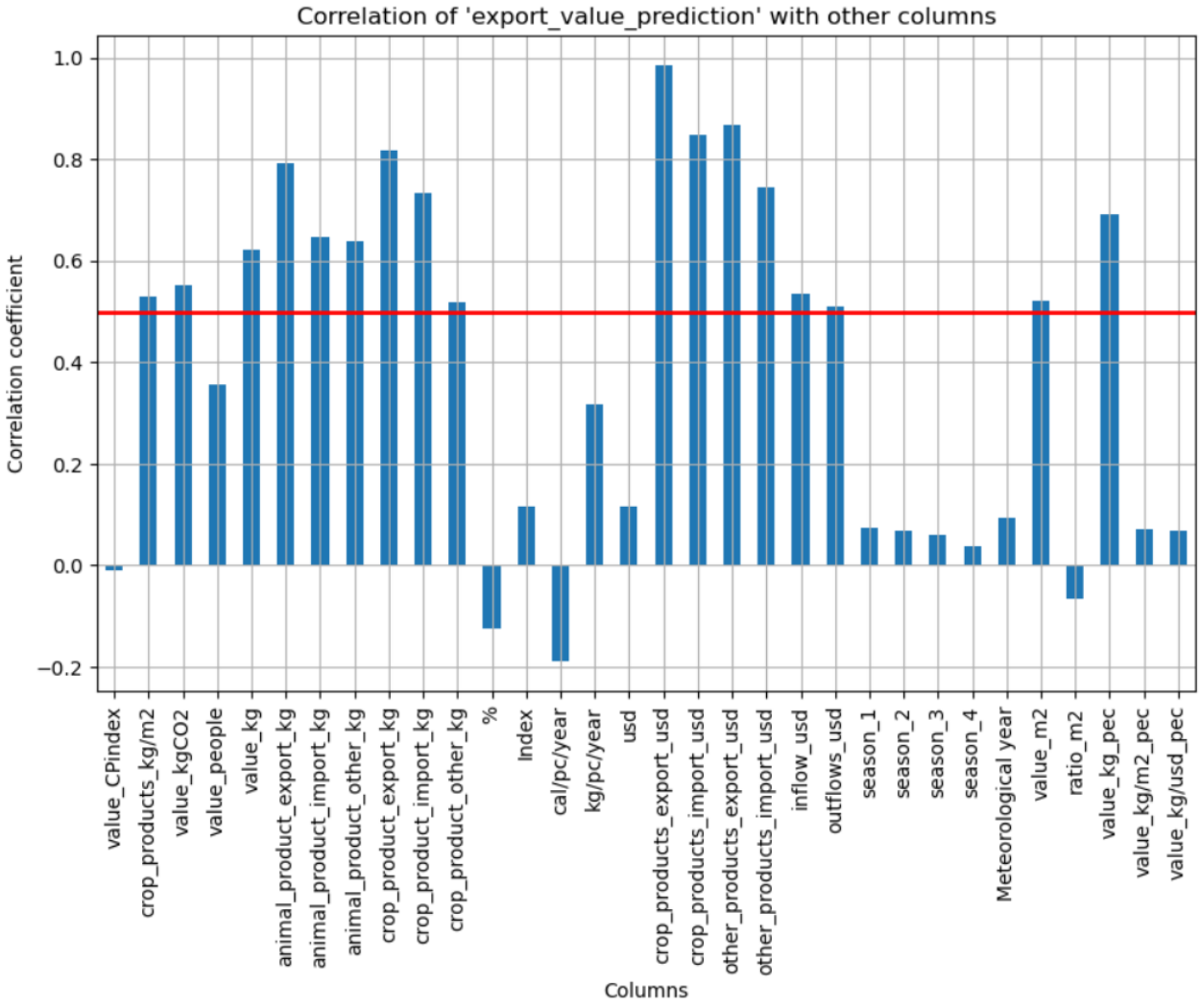
value_kgCO2	Emissions	It was found through emissions. There were different types of emissions indicated with different elements (N_2O, CH_4, CO_2). I wanted to show them their equivalents in CO_2. 1 kg CH_4 is equal to 25 kg of CO_2, and 1 kg of N_2O is equal to 298 kg CO_2. Also, the text in the domain was long a bit. A changed them whether it indicates crops emission or soil emission. After checking their number of instances of both, I saw that they had roughly 250% differences between each other. Due to that reason, I took crop and soil emissions and emission. Then, I changed the name of the value column as "value_kgCO2" to understand easier.
value_people	Employment	I checked the number of instances for the indicator column. Mean weekly hours were 2.5 times fewer occurred in the dataset. I excluded them to prevent NaN value occurrence. The unit of the number of employees was 1000 people. I converted them to person by multiplying 1000. At the end, I changed the name of the column called value to value_people.
value_kg	Fertilizers Use	The values were indicated with tonnes. All of the values were converted into kg. Also, the the column named "Value", was changed to "value_kg" by summing the values of all item types.
animal_product_export_kg animal_product_import_kg animal_product_other_kg crop_product_export_kg crop_product_import_kg crop_product_other_kg	Food Balances Indicators	It was derived from food balances indicators. At first, the unit was converted from 1000t to kg to obtain common units in all values. After checking the number of records for each element grouping by country, there was only 1 missing one in one of them. Because of that, I excluded this missing country to make them equal. I divided elements into 3 groups: 1- Import 2- Export 3- Others Additionally, I checked the item numbers in the same way as I did in element. I divided items into 2 groups: 1- Crop product 2- Animal product To get rid of the element, and unit columns, I added import, export, or other at the end of items as well as kg.

% index cal/pc/year kg/pc/year usd	Food Security Indicators	Some values were indicated as 3-year period. Those values splitted into the years. If there were multiple values for a year, I took the average. The beginning year and the end year had only 1 value. A year had maximum 3 different value. However, I created lists of dictionaries of dictionary for each type of item (took them as key). After that, I added in-dictionary by taking each year as key and list as value. In the list I added the value and country. The even values were values and the odds were areas. Then, I created a dataframe and groupped it according to area, year, and item by taking the mean value. The value and unit conversions were performed here as well: - g/pc/d => kg/p/year (1/365000) - kcal/pc/d => cal/pc/year (1000/365) - 1000 I$ => USD (1000)
crop_products_export_usd crop_products_import_usd other_products_export_usd other_products_import_usd	Food Trade Indicators	The unit was converted from 1000 USD to USD multiplying by 1000. The number of occurrences in areas for both element and item columns was checked. All were equal. The items were separated into 2 groups: 1- **Crop products:** 'Cereals and Preparations', 'Sugar and Honey', 'Tobacco', 'Fruit and Vegetables', 'Non-alcoholic Beverages', 'Fats and Oils (excluding Butter)', 'Alcoholic Beverages', 'Non-edible Fats and Oils' 2- **Other products:** 'Meat and Meat Preparations', 'Dairy Products and Eggs', 'Other food', 'Non-food' To exclude the element column, whether an item indicates export of import was added to the end of each product like crop_products_export. At the end, it was converted into pivot table.
inflow_usd outflows_usd	Foreign Direct Investment	The number of occurrences of each item grouping by area was checked. They were divided into 2 groups as inflows and outflows, since some of them had very few numbers. The unit of million USD converted into USD multiplying by a million. Then, a pivot table was created by choosing inflows and outflows as column features. Both forward and backward interpolations were performed. If there still were less than 10 years of records in terms of areas, they were excluded.

season_1 season_2 season_3 season_4 meteorological year	Land Temperatur e Change	Months were divided into seasons from season_1 to season_4. The temperature was indicated as seasonal and yearly. Therefore, I noticed that if there was only one missing season value, I could fill it by using yearly changes. $$season_1 + season_2 + season_3 + season_4 = Meteorological\ year$$ In that way, 51 NaN values were filled. When I checked the standard deviation values for all countries. I noticed that they were the same for each country over years. Thanks to this information, I filled some of the other NaN values as well. Nevertheless, there were many null values for standard deviations especially. 43 of the countries had more than 50% null values in standard deviation for seasons, while it was only 9 for temperature change. Because of that, I dropped standard deviation values from the data frame. The remaining null values were filled by backward and forward interpolation methods except for 5 areas. At the end, a pivot table where seasons and meteorological years are columns was created.
value_m2 ratio_m2	Land Use	Country area and land area were the items that occurred more than others. Due to that reason, I dropped the other ones. Also, the unit was converted to m^2 to make the units in all tables the same. In addition to that, I found the ratio of land area over country area followed by dropping the land area. This feature only indicates the country area.
value_kg_pec value_kg/m2_pec value_kg/usd_pec	Pesticides Use	The units were converted: - t => kg (1000) - kg/ha => kg/m^2 (10000) - g/Int$ => kg/USD (0.001) Then, the units were divided into columns seperately by creating pivot table.
export_value_prediction	y_values	After a pivot table was created in Food Trade Indicators, the columns of area, year, and crop_products_export_usd were taken as another data frame. A new column was created and added the value 3

		years after according to the year and country. Since 3 years into the future was desired to be predicted, the maximum year was checked in the table (it was 2022). Until 3 years before was taken the data frame, in other words until 2019. The countries which had NaN values were excluded from the table at the end.

After obtaining these features, the correlation matrix and different representations were created as it can be seen in ***Figure***. From here, I took the features which have more than 50% correlation with the target.



value_kg was excluded from the features. Because it had many null values. Other 17 features in the graph with 2 others called area, and year were chosen.

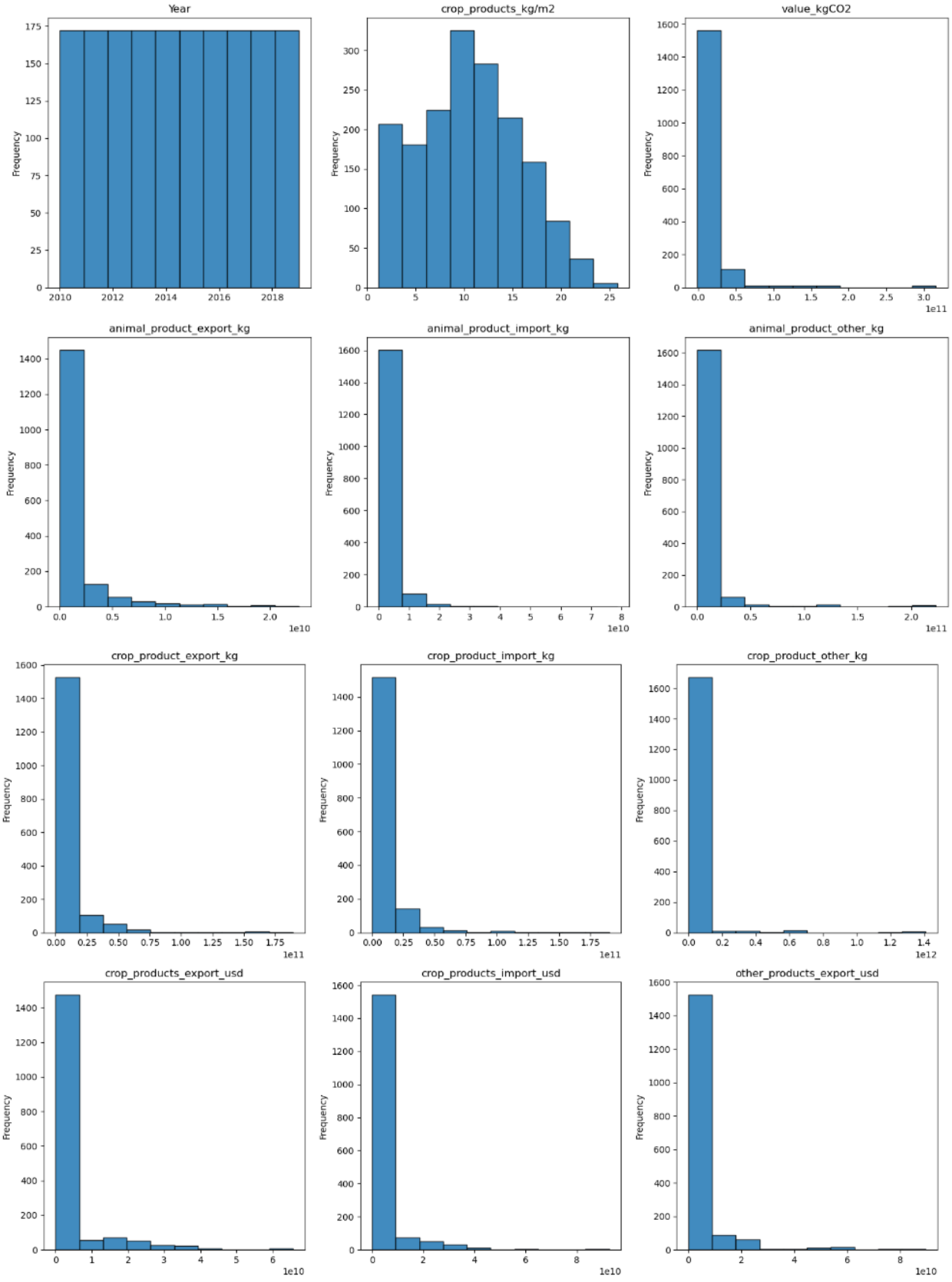
Area	0
Year	0
crop_products_kg/m2	10
value_kgCO2	0

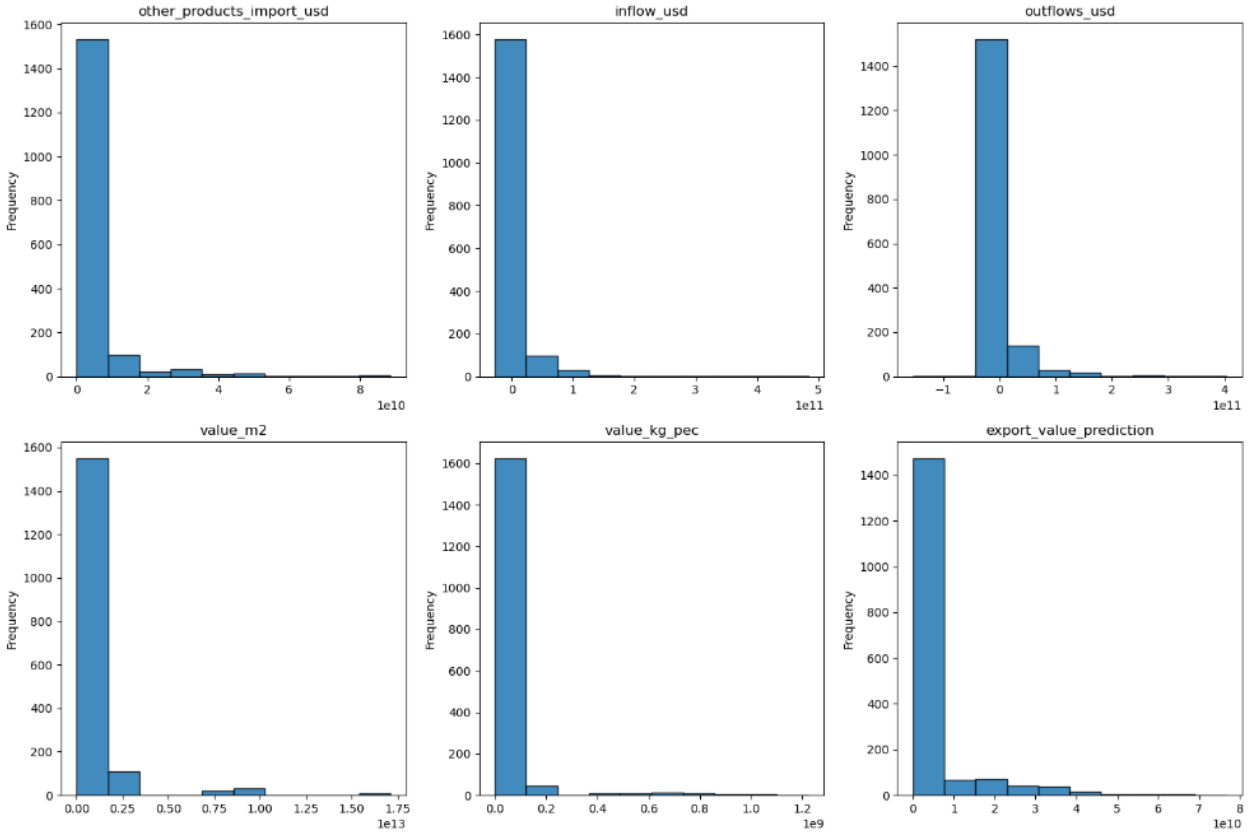
value_kg	388
animal_product_export_kg	146
animal_product_import_kg	146
animal_product_other_kg	146
crop_product_export_kg	146
crop_product_import_kg	146
crop_product_other_kg	146
crop_products_export_usd	0
crop_products_import_usd	0
other_products_export_usd	0
other_products_import_usd	0
inflow_usd	60
outflows_usd	60
value_m2	0
value_kg_pec	60
export_value_prediction	0

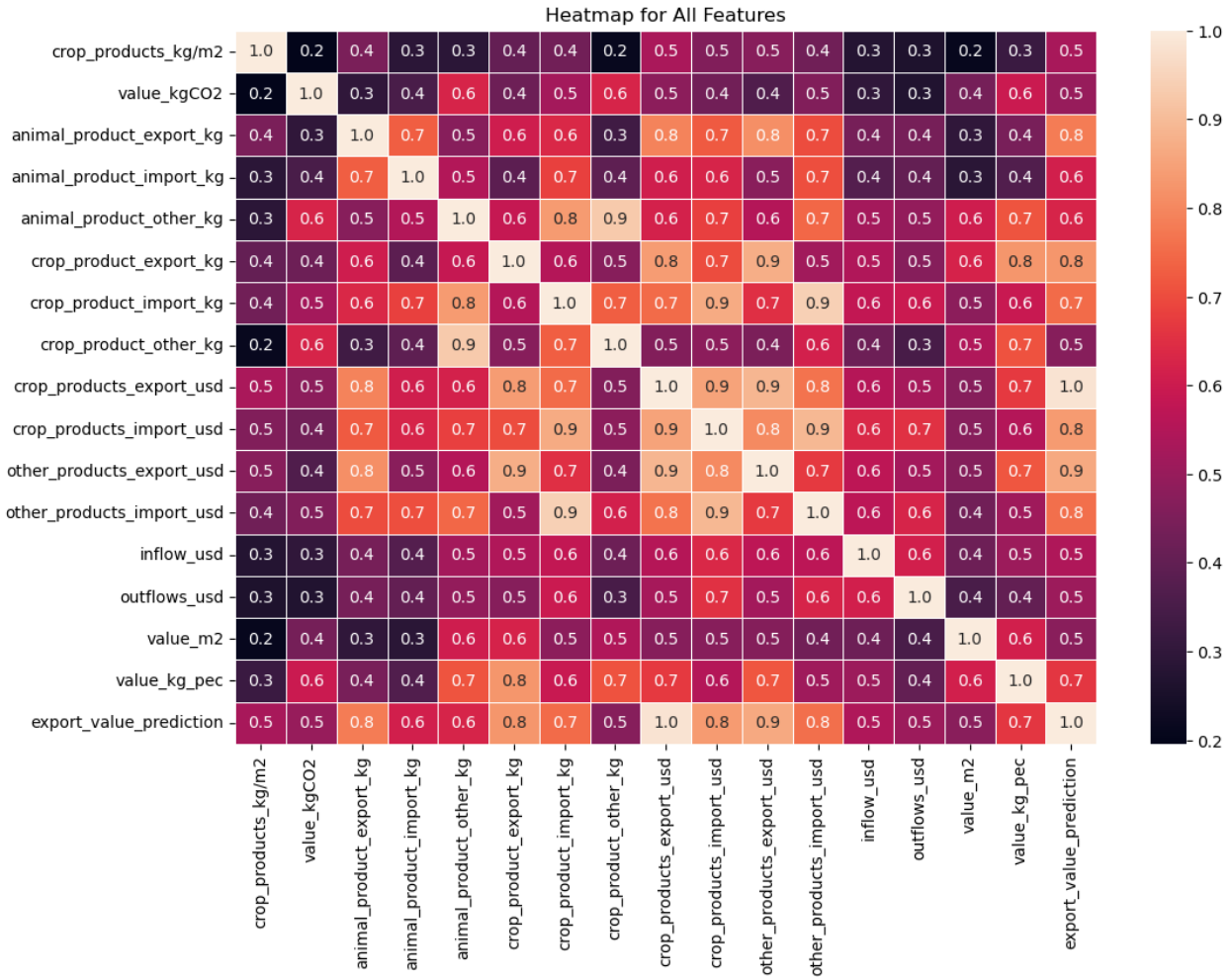
The chosen features:

- Area
- Year
- crop_products_kg/m2
- value_kgCO2
- animal_product_export_kg
- animal_product_import_kg
- animal_product_other_kg
- crop_product_export_kg
- crop_product_import_kg
- crop_product_other_kg
- crop_products_export_usd
- crop_products_import_usd
- other_products_export_usd
- other_products_import_usd
- inflow_usd
- outflows_usd
- value_m2
- value_kg_pec
- export_value_prediction

Distribution of Each Feature:


Year
Frequency
crop_products_kg/m2
Frequency
value_kgCO2
Frequency
1e11
animal_product_export_kg
Frequency
1e10
animal_product_import_kg
Frequency
1e10
animal_product_other_kg
Frequency
1e11
crop_product_export_kg
Frequency
1e11
crop_product_import_kg
Frequency
1e11
crop_product_other_kg
Frequency
1e12
crop_products_export_usd
Frequency
1e10
crop_products_import_usd
Frequency
1e10
other_products_export_usd
Frequency
1e10


other_products_import_usd
Frequency
1e10
inflow_usd
Frequency
1e11
outflows_usd
Frequency
1e11
value_m2
Frequency
1e13
value_kg_pec
Frequency
1e9
export_value_prediction
Frequency
1e10



4. PREPROCESSING

4.1. Missing Values

Interpolation method: Before merging and after merging the main dataframe, the interpolation method was used.

$$y = y_1 + (x - x_1)\frac{(y_2 - y_1)}{(x_2 - x_1)}$$

y: linear interpolation value

x: independent variable

x_1,y_1: first known values

x_2,y_2: second known values

Dropping: After applying interpolation to all columns in the merged data frame, there were still null values. I didn't want to change the meaning of the data. Because of that, the areas which have null values were dropped from the data frame.

4.2. Encoding Categorical Variables

The only column which had categorical data was Area, which contains the country names. I converted them into number by label encoding.

Label encoder: I used the label encoder for countries to label each country with a different number. It gave a unique number for each country. It's less expensive than one-hot encoding in terms of memory.

4.3. Scaling Numerical Variables

Other columns contained fully numerical value. When there is a huge difference between 2 features, the model doesn't work properly. According to the distribution of each feature, each has completely different variance.

Standard Scaler: Scale data by taking the variance into account. It helps to model by improving the performance and converging gradient descent.

$$Standardized\ Value\ (z) = \frac{x - \mu}{\sigma}$$

4.4. Splitting Data

The dataset was split into 2 parts by shuffling:

1-Training set (80%)

2-Test set (20%)