CandNo: 277191

Machine Learning

Contents

Preprocessing

1.1 Data Preparation

13 datasets have been given to build a multilayer perceptron model used for forecasting the export value of crop products for a geographical region three years into the future. 12 datasets were used in this project, included in Grid 1. The only dataset that's been left out is Food security indicators. In this dataset, the column Year contains a range of year time (e.g., 2001-2003) instead of the data for each year, making it unclear for the merge process performed later if group by on each year is conducted across the other dataset.

Food trade indicators	Crops production indicators	Emissions	Employment
Exchange rate	Fertilizers use	Food balances indicators	Foreign direct investment
Land temperature change	Land use	Pesticides Use	Consumer prices indicators

Grid 1 The dataset used in this project

1.2 Feature Extraction

At this stage, data cleaning on each dataset is performed.

The process steps mainly follow:

1. **Unused column drop**: Unused columns will be dropped out for the later processes.
2. **Columns concatenation**: If the dataset contains more than one unique category to represent the value column, concatenation on those columns will be performed to make sure the value corresponds to each unique category.
3. **Groupby function**: This is done for every dataset to summarize the 'Year', 'Area', and the unique categories with an appropriate summary method (e.g., sum or mean). By doing so, the value will be able to represent the value for each country in a year based on the categories.
4. **Pivot table transformation**: The pivot table is created based on the category column with the 'Area' and 'Year' columns as the index with a summary method to get the value for each category based on each country and year.
5. **Additional new columns**: A new column is added to represent the value summary.
6. **Merge**: Each processed dataset will be merged with the export value dataset (Food trade indicators) so that all the features from the other dataset will be available for the MLP model as the input.

The below process shows how each dataset has been handled in detail:

- Food trade indicators:

'Note', 'Flag Description', and 'Flag' columns are dropped since no use for them later. Concatenation will be performed on 'Item' and 'Element' to get the unique category for each export and import value. Next, I grouped by 'Area', 'Year', and 'Item_Element' with 'Value' columns. After grouping the data, the 'Value' column of each group is calculated by the sum of this column for each group. Then, the pivot table is created based on the 'Item_Element' column with the 'Area' and 'Year' columns as the index to sum all the values within that country and Year. By doing this, we can get the export value for each item based on each country and year. Lastly, a new column 'total_y' is created by summing up all the export values in the column. Here, a base

Food trade indicators dataset is created, which will be used for merging with other datasets to expand the features.

- Crops production indicators:

Columns 'Note','Flag Description','Flag','Year Code','Item Code (CPC)','Area Code (M49)','Domain Code','Domain','Element','Element Code','Unit' are dropped. Then, I grouped by 'Area', 'Year', and 'Item' by the sum of the 'Value' column for each group. The reason to choose sum over other options is that the crop production for each country in each year is calculated by the summation. Then, the pivot table is created based on the same method as used in Food trade indicators. An additional column has been created to sum up all the values cross Item columns, named total_crop_production. Lastly, the dataset is merged with the previous df_food_trade dataset on the same 'Year' and 'Area' columns. There are 39 columns in total after the merge.

- Emissions:

Since there are two categories (Item and Element) to represent each emission data in this dataset. I first concatenate both columns. And then group by the 'Area','Year','Item_Element' by the sum of the 'Value' column for each group. Summing emissions value for each year provides a clear picture of the total environmental impact of an entity (such as a country, industry, or corporation) over that year. The same method for pivot tables separates each element in 'Item_Element' by the sum of the values. Column 'total_emission' is added at the end. the dataset is merged with the previous merged_crop_production dataset on the same 'Year' and 'Area' columns. There are 46 columns in total after the merge.

- Employment:

The unit of the value in the employment dataset is different so I divided all the values labeled 1000 in the 'Unit' column by 1000 to normalize the value. And then the concatenation of the column 'Indicator' and 'Source' is performed to get the unique category. Since the value is the only thing that's been considered, columns such as 'Value','Source Code','Source','Indicator','Indicator Code','Sex Code','Sex','Element','Note','Flag Description','Flag','Year Code','Area Code (M49)','Domain Code','Domain','Element Code','Unit','Source','Source Code' are dropped. For the group by and the pivot table method, the only difference is this time the mean of each group is taken, for it's the ratio number of the employment rate in a year. And the total sum column is not added at the end. The merge function is also conducted with the previous dataset emission dataset on 'Year' and 'Area' columns. There are 53 columns after the merge.

- Exchange rate:

The process for exchange rate is almost the same as the previous steps except for the total sum column is not added since it's not right to sum the ratio exchange rate. There are 54 columns after the merge with the previous dataset Employment.

- Fertilizers Use:

Columns 'Domain Code','Domain','Area Code (M49)','Element Code','Element','Item Code','Flag Description','Year Code','Unit','Area Code (M49)','Flag','Flag Description' are dropped. 'Area','Year', and 'Item' columns are grouped by with the sum of the 'Value' columns to take the total fertilizers used for each country and year. Pivot tables are performed the same. 'total_fertilizers' is added as the sum of the total fertilizers across all items to the dataset. Lastly, merge the dataset with the previous Exchange rate dataset on 'Year' and 'Area'. There are 78 columns in total after the merge.

- Food balances indicators:

There are two main categories, 'Element' and 'Item', to represent each food balance indicator value so the concatenation of both columns is necessary. Columns 'Domain Code','Domain','Area Code (M49)','Element Code','Element','Item','Year Code','Unit','Flag','Flag Description' are dropped for no contribution of what's interested. The new column 'food_balances_Item_Element' is used with 'Year' and 'Area' to perform group by on the Value column to get the sum of each item. There are 163 columns after the merge with the previous dataset fertilizers use.

- Foreign direct investment:

Columns 'Domain Code','Domain','Area Code (M49)','Element Code','Element','Item Code','Year Code','Unit','Flag','Flag Description','Note' are dropped as the same reason. 'Area','Year', and 'Item' columns are grouped by with the sum of the 'Value' columns to take the total foreign direct investment for each country and year. Pivot tables are performed the same. ' total_foreign_direct' is added as the sum of the total fertilizers across all items to the dataset. Lastly, merge the dataset with the previous Food balances indicators dataset on 'Year' and 'Area'. There are 170 columns in total after the merge.

- Land temperature change:

Columns 'Domain Code','Domain','Area Code (M49)','Element Code','Months Code','Unit','Flag','Flag Description' are dropped. The data type of the 'Year Cpode' column is changed to string since the later concatenation needs to be performed. The concatenation with 'Months' , 'Year Code', and 'Element' is done to get all the unique time periods for the value. Then, 'Area','Year', and 'Months_year_Element' columns are grouped with the mean of the 'Value' columns to take the average of land temperature change for each country and year. Pivot tables are performed the same. After merging with the previous dataset, 400 columns are there in the dataset.

- Land use:

Columns 'Domain Code','Domain','Area Code (M49)','Element Code','Element','Year Code','Unit','Flag','Flag Description','Note' are dropped. Gropby is used in the same way to summarize the value for each item in each country every year. The pivot table is performed the same. 'total_land_use' column is created to calculate the total amount of land use across all the categories of items. After merging with the previous dataset, the dataset has 421 columns.

- Pesticides use:

Domain Code','Year Code','Domain','Area Code (M49)','Element Code','Element','Item Code','Item','Unit','Flag','Flag Description','Note' are dropped. Groupby and the pivot table are performed in the same way. After merging with the previous dataset, there are 430 columns.

- Consumer price indicators:

'Domain Code','Year Code','Domain','Area Code (M49)','Element Code','Element','Item Code','Unit','Flag','Flag Description','Note','Months Code','Months' are dropped. Groupby and pivot tables are performed in the same way. After merging with the previous dataset, there are 432 columns.

1.3 Removing Duplicates

13705 duplicates were found after the final merge. Duplicates in the training data can lead to overfitting. This happens because the model might start to recognize and memorize the repeated patterns rather than learn the underlying general patterns needed to perform predictions on data outside the training set. I applied dop_duplicates() reviewed all columns in the DataFrame and removed rows that have identical values in all columns. The inplace=True argument modifies the DataFrame in place, meaning it does not return a new DataFrame but instead updates the current one.

1.4 Dealing with the Missing Values

Since I performed the merge function with 12 datasets to expand the total features, 23288823 missing values were found across all datasets. MLP models usually do not inherently handle missing values. If missing values are present in the input data, the MLP cannot process the data unless these values are either removed or imputed. final_df.fillna (method =ffill ,inplace=True) this line of code is used to forward fill the NaN values. This means that a NaN value in the final_df will be filled with the last preceding non-NaN value along the column. If there is no valid value before a NaN (i.e., if the NaN is at the beginning of the column), the NaN remains. As for this reason, the rest of the NaNs will be filled with -1. I didn't use 0 to fill with the NaNs is because 0 does exist in the dataset after the merge process. In order not to confuse the MLP model with the real value and the missing values, -1 is decided to represent the NaNs.

1.5 Feature Scaling/Normalization

- Normalization (Yeo-Johnson transformation)

  There exists heteroscedasticity in the X features. Heteroscedasticity is the concept in statistics, that the standard errors of a variable monitored over some amount of time are non-constant. The Breusch–Pagan test is used here to test for the null hypothesis that the variance of the residuals is constant. If this null is rejected, then there exists heteroscedasticity. The very low p-values in the test result ($p\text{-value} < 0.05$) suggest that the presence of heteroscedasticity requires further transformation. Because the X features contain zero or negative values and a wider range of data, I applied the Yeo-Johnson transformation (Grid 2) on the features X for training, validation, and testing data to stabilize the variance. Yeo-Johnson Transformation Generalization of Box-Cox Transformation, which can handle positive and negative values and outliers.

$$\frac{(y+1)^{\lambda}-1}{\lambda} \quad \text{if } \lambda \neq 0, y \geq 0$$

$$\log(y+1) \quad \text{if } \lambda = 0, y \geq 0$$

$$-\frac{\left[(-y+1)^{2-\lambda}-1\right]}{2-\lambda} \quad \text{if } \lambda \neq 2, y < 0$$

$$-\log(-y+1) \quad \text{if } \lambda = 2, y < 0$$

Grid 2 Yeo-Johnson

- Normalization (Log Transformation)
  Since y shows characteristics of exponential growth and has right-skewed data (Figure 1), Log Transformation is therefore appropriate. The definition of the log function on positive real numbers allows it to give real numbers. This compresses the range of large values and expands the range of small values. This defines the equation: $y' = \log(1 + y)$. Note that $1 + y$ will ensure that whatever the value of y at each point, the input to the log function is always greater than zero, in case y is zero or a very small positive number. This is because the logarithm of zero, or a negative number, is not defined.

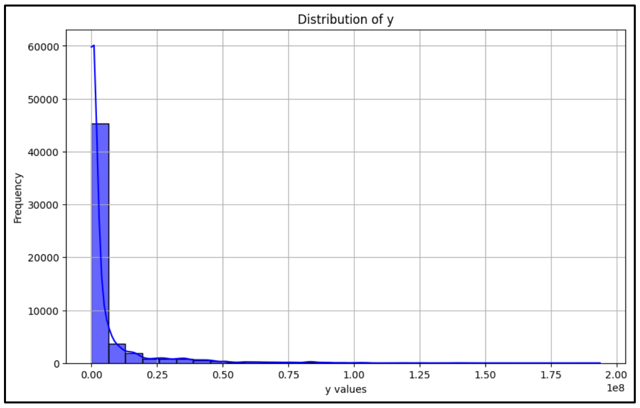


Figure 1 label y distribution

- Scaling
  194027 outliers have been found in the final_df after the merge. Outliers might cause an overfitting problem when training the MLP model since the model may overfit to the outliers, reducing its ability to generalize to new, unseen data. The scaler which works fine with many outliers is RobustScaler. It uses the median and interquartile range in scaling of data, making it more robust to outliers, which could then otherwise affect the process of scaling. RobustScaler is defined as:

$$X_{\text{scaled}} = \frac{X - \text{median}(X)}{\text{IQR}(X)}$$

  where $\text{median}(X)$ is the median of the feature and $\text{IQR}(X)$ is the interquartile range, which is the difference between the 75th and 25th percentiles. This is how RobustScaler is used to reduce the effect of the outliers on the scaling.

Features & Labels

2.1 Features Creation

The features were mainly drawn from each CSV file's category columns such as Element, Item, or Indicator. Each category will be transformed into the pivot tables with an appropriate aggregated function. After the transformation, the pivot table is then merged with the base dataset (Food trade indicators) on the columns Year and Area to form the final dataset for the MLP model training. With the future3year_total_y created to represent the future 3-year time export values, there are 433 features in total.

Below are the details of the category columns used to form the features from each CSV file:

1. Food trade indicators:
   - Item column is used. Item column includes 'Cereals and Preparations', 'Fats and Oils (excluding Butter)', 'Meat and Meat Preparations', 'Sugar and Honey', 'Fruit and Vegetables', 'Dairy Products and Eggs', 'Alcoholic Beverages', 'Non-alcoholic Beverages', 'Other food', 'Non-food', 'Non-edible Fats and Oils', 'Tobacco'.
   - Element coulumn is used to get both Export Value and Import Value.
   - total_y is created for the summation of the total food trade indiactor value for each country every year.
2. Crops production indicators:
   - Item column is used. Item column includes 'Cereals, primary', 'Citrus Fruit, Total', 'Fibre Crops, Fibre Equivalent', 'Fruit Primary', 'Oilcrops, Cake Equivalent', 'Oilcrops, Oil Equivalent', 'Pulses, Total', 'Roots and Tubers, Total', 'Sugar Crops Primary', 'Treenuts, Total', and 'Vegetables Primary'.
   - total_crop_production is created for the summation of the total crop production value for each country every year.
3. Emissions:
   - Item and Element are used, which include 'All Crops', 'Cropland organic soils', 'Grassland organic soils' in the Item column.
   - 'Crops total (Emissions N2O)', 'Crops total (Emissions CH4)', 'Emissions (N2O)', 'Emissions (CO2)' in the Element column.
   - total_emissions is created for the summation of the total emissions value for each country every year.
4. Employment:
   - Indicator and Source columns are used, which include 'Mean weekly hours actually worked per employed person in agriculture, forestry and fishing', 'Employment in agriculture, forestry and fishing - ILO modelled estimates' in the Indicator column.
   - 'Household income and expenditure survey', 'ILO - ILO Modelled Estimates', 'Labour force survey', 'Population census', 'Employment surveys', 'Household survey', 'Official estimates' in the Source column.
5. Exchange rate:
   - Only the Value column is used for the merge for the exchange rate feature.
6. Fertilizers use:
   - Item column is used. Item column includes 'NPK fertilizers', 'Urea', 'Ammonium nitrate (AN)', 'Ammonium sulphate', 'Calcium ammonium nitrate (CAN) and other mixtures with calcium carbonate', 'Diammonium phosphate (DAP)', 'Monoammonium phosphate (MAP)', 'Other NP compounds', 'PK compounds', 'Potassium chloride (muriate of potash) (MOP)', 'Potassium nitrate', 'Potassium sulphate (sulphate of potash) (SOP)', 'Sodium nitrate', 'Superphosphates above 35%', 'Superphosphates, other', 'Ammonia, anhydrous', 'Phosphate rock', 'Urea and ammonium nitrate solutions (UAN)', 'Fertilizers n.e.c.', 'Other nitrogenous fertilizers, n.e.c.', 'Other phosphatic fertilizers, n.e.c.', 'Other potassic fertilizers, n.e.c.', 'Other NK compounds'.
   - total_fertilizers is created for the summation of the toal fertilizers value for each country in every year.

7. Food balances indicators:
    - Both Element and Item columns are used, which include 'Import Quantity', 'Export Quantity', 'Losses', 'Other uses (non-food)', 'Food' in Element.
    - 'Cereals - Excluding Beer', 'Starchy Roots', 'Sugar Crops', 'Sugar & Sweeteners', 'Pulses', 'Treenuts', 'Oilcrops', 'Vegetable Oils', 'Vegetables', 'Fruits - Excluding Wine', 'Stimulants', 'Spices', 'Alcoholic Beverages', 'Meat', 'Eggs', 'Milk - Excluding Butter', 'Fish, Seafood' in Item.
    - total_food_balances_indicators is created for the summation of the toal food balances value for each country in every year.
8. Foreign direct investment:
    - Item column is used, which includes 'Total FDI inflows', 'Total FDI outflows', 'FDI inflows to Agriculture, Forestry and Fishing', 'FDI inflows to Food, Beverages and Tobacco', 'FDI outflows to Agriculture, Forestry and Fishing', 'FDI outflows to Food, Beverages and Tobacco'.
    - total_foreign_direct is created for the summation of the foreign direct investment value for each country in every year.
9. Land temperature change:
    - Months, Year Code, and Element are used, which includes 'Dec–Jan–Feb', 'Mar–Apr–May', 'Jun–Jul–Aug', 'Sep–Oct–Nov', 'Meteorological year' in Months.
    - 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016, 2017, 2018, 2019, 2020, 2021, 2022 in Year Code.
    - 'Temperature change', 'Standard Deviation' in Element.
10. Land use:
    - Item column is used, which includes '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'.
    - total_land_use is created for the summation of the land use value for each country in every year.
11. Pesticides Use:
    - Both Element and Item are used, which includes 'Agricultural Use', 'Use per area of cropland', 'Use per value of agricultural production' in Element.
    - 'Pesticides (total)', 'Insecticides', 'Herbicides', 'Fungicides and Bactericides', 'Fungicides – Seed treatments', 'Insecticides – Seed Treatments', 'Rodenticides' in Item.
12. Consumer prices indicators:
    - Item column is used, which includes 'Consumer Prices, Food Indices (2015 = 100)' and 'Food price inflation'.

2.2 Numerical Transformation

After extracting and merging all the features and columns from each CSV, I converted categorical string data into a model-understandable numerical format. Categorical columns such as ‘Area ’ are performed. The LabelEncoder from sklearn.preprocessing is used. This encoder assigns a unique integer to each unique string label in the order they are encountered. For example, if Area includes ['Japan', 'Taiwan', 'India'], they might be encoded as [0, 1, 2].

2.3 Feature Selection Method

It may not be the case that all the features in the dataset from df_final are important for predicting the value of exports. L1 regularization helps in automatic feature selection through the shrinking of the coefficient of less important features to zero for reducing overfitting. L1 adds a penalty equal to the absolute value of the magnitude of coefficients to the loss function. L1 can also help the model be more robust to outliers, which is best suited to the df_final dataset which has many features and outliers. L1 regularization is defined as:

$$\text{Objective} = \text{RSS} + \alpha \sum_{j=1}^{p} |\beta_j|$$

where:

- RSS is the residual sum of squares.

- α is the regularization parameter.

- β_j are the coefficients of the features.

At a higher value of α, the regularization term has a more significant impact, meaning that more coefficients are shrunken to zero. It effectively removes them from the model and represents feature selection. I used LassoCV for the choice of α. The choice of the best alpha in scikit-learn's LassoCV by cross-validation involves balancing the model between bias and variance in an individual dataset.

2.4 Label

Firstly, total_y is calculated by summing up all the unique categories for export values in the food trade CSV file. This represents the total amount of the export values for each country in a certain year. Then, future3year_total_y column is calculated to represent future predictions about 3 years ahead data as the label y in the model. Groupby ('Area') is used to make sure the calculation only happens within the same country instead of the entire dataset. For each group, the values are calculated by applying ['total_y'].shift(-3) with a negative parameter -3 to shift the column values of total_y up for 3 rows, indicating the three year time in the future for each country. At the end, I cleaned the data by removing rows where future predictions cannot be made due to insufficient data (The year in the dataset only exists up until 2022, meaning 2019 is the last year the model can predict.) The column total_y is defined as:

$$\text{Label}_Y = \text{ExportValue}_{Y+3}$$

MLP model

I used the sequential model from Tensorflow Keras to create a linear stack of layers.

3.1 Input Layer

The input layer receives the input features from the Robustscaler scaled value after the Yeo Johnson transformation for X features and log transformation for y label. The number of neurons in the input layer equals the number of input features, which has 108 features in total (106 Lasso selected features with 2 customized features, Year and Area.) Inputs for theMLP is defined as:

$$\text{input} = \text{number of selected features} + 2 \text{ (for Year and Area)}$$

Columns Year and Area are added separately since the purpose of this project is to forecast the export value for a region three years into the future. Thus, these columns can’t be filtered by Lasso.

3.2 Dense Layers

The dense layer is the standard layer of a neural network; each neuron receives input from all neurons of the previous layer, making it fully connected.

- First Hidden Layer:
  1. Dense layer with 100 neurons. Generally, the first dense layer has more neurons in number. This is because this layer is directly attached to the input layer. This layer learns a wide range of features and patterns from the input data. The use of 100 neurons enables one to capture a wide variety of interactions and complexities present in the input features. This is of high importance for the trade dataset, with a wide diversity and potentially some nonlinearity of the relationship between all the agricultural metrics over various years and areas.
  2. Activation function: The Leaky ReLU allows a small non-zero gradient for negative inputs, while the normal ReLU takes the output as zero for negative values. This helps in prevention of dying ReLU, which occurs due to a problem in which a lot of neurons' output become zero for every input given. Zero and negative values are there in the input features of final_df. A neuron will most likely generate the zero when the ReLU activation is present, rendering it inactive. Leaky ReLU is defined as:

     $$\text{LeakyReLU}(x) = x \text{ for } x > 0 \text{ and } \text{LeakyReLU}(x) = \alpha x \text{ for } x \leq 0.$$

     The activation function outputs the input value scaled by the factor α α.

  3. In this MLP model, I added batch normalization and 0.3 dropouts to prevent overfitting. The trade dataset contains a range of values for both imports and exports of agricultural products. These methods help stabilize the training process and prevent the problem of overfitting with noisy data and many dimensions. Batch normalization is one way to normalize the input throughout the deep neural network to stabilize and speed up model training. Batch normalization is defined as:

     $$\hat{x} = \frac{x - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

  4. where (μ) is mean, (σ^2) is variance, and (ϵ) is a small constant for numerical stability. y is the normalized inputs to each layer. Therefore, batch normalization serves to stabilize the learning process in the network, preventing gradients from exploding or being squashed to zero. At each stage in training, dropout drives a fraction of the input units to zero. It helps to make the model not completely dependent on any one neuron. I used a dropout of 30% for this model. In this case, during each training step, approximately 30% of the node activations are not added to the next layer. dropout is defined as:

$$With\ probability\ (\ p\ ), the\ output\ is\ 0.$$

$$With\ probability (1-p), the\ output\ is (\frac{x_i}{1-p})$$

- Second Hidden Layer:
  1. The second dense layer consists of 50 neurons, which is actually less than the first dense layer. Factually, the decrease in the number of neurons in the subsequent layers aids in learning essential features from the layers in front of them. The fact that the number of neurons is reduced for the subsequent layers also contributed to this factor of reducing the risk of overfitting due to a too-complex network.
  2. Activation function: Leaky ReLU (α = 0.01 α=0.01).
  3. Batch normalization and dropout (rate = 0.3) for regularization.

3.3 Output Layer

Dense layer with 1 neuron with an activation function, linear which outputs a single continuous value for each input sample as the prediction. Linear function:

$$\text{Output} = w \cdot x + b$$

where (w) is the weight vector, (x) is the input vector, and (b) is the bias.

3.4 Optimizer

The Adam optimizer is made to work effectively with sparse gradients on noisy problems in the df_final dataset. Adam determines individual adaptive learning rates for different parameters and is helpful, considering that the dataset is most likely to have features with different scales, like export values and import values, and different agricultural use metrics.

- Adam optimizer
  - ◆ Calculate the Gradients: Compute the gradients of the loss for the weights, (g_t), for each time step (t).

$$g_t = \nabla_{(\theta)} J(\theta_t)$$

  - ◆ Update Biased First Moment Estimate:

$$m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t$$

Where (β_1) is the exponential decay rate for the first moment estimates.

  - ◆ Update Biased Second Raw Moment Estimate:

$$v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2$$

Where (β_2) is the exponential decay rate for the second-moment estimates.

  - ◆ Correct Bias in First Moment:

$$\widehat{m_t} = \frac{m_t}{1-\beta_1^t}$$

This correction helps counteract the bias towards zero in the initial steps of the algorithm.

- Correct Bias in Second Moment:

$$\widehat{v_t} = \frac{v_t}{1 - \beta_2^t}$$

Similar to the first moment, this step corrects the bias in the second moment estimate.

- Update the Weights:

$$\theta_{t+1} = \theta_t - \frac{\alpha \cdot \widehat{m_t}}{\sqrt{\widehat{v_t}} + \epsilon}$$

Where:
- θ_t are the parameters (weights) at time step t.
- α is the learning rate.
- ϵ is a small constant (e.g., 10^{-8}) added to improve numerical stability.

- Loss Function: The Mean Squared Error calculates the average of the squared differences between predicted values and actual values, indicating how well the model fits the data. Loss Function is defined as:

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

where (y_i) is the actual value and $\widehat{y_i}$ is the predicted value.

3.5 Early Stopping

It is a regularization form to avoid overfitting at the time of training a learner with any iterative method, like gradient descent.The function is defined as:

$$\text{Stop training if} \quad \text{val\_loss}_{\text{current}} > \min\left(\text{val\_loss}_{\text{past}}\right)$$

- This callback will halt the training if the validation loss does not improve over 10 epochs in a row (patience=10).
- restore_best_weights=True ensures that the weights of the model are reverted to the best ones to have the minimum loss in validation; so, in short, it reverts back to the best model iteration during training.
- The model.fit() function is trained with a fixed number of epochs (epochs = 100) on the data.
- batch_size=10 indicates the number of samples per gradient update for training.
- validation_data is to be used at the end of an epoch to evaluate the loss and any model metrics. Training on this data will not be done.
- verbose=1 is to print a progress bar for each epoch during training.

3.6 Cross-Validation K-Fold Cross-Validation

This is done in order to make sure that the model could perform consistently across all subsets of the data. The data is then divided into k folds, and the model is trained and evaluated k times—each time, with a different fold acting as the validation set and the remaining folds as the training set. K-Fold is defined as:

$$\text{Average Score} = \frac{1}{k}\sum_{i=1}^{k} \text{Score}_i$$

Performance

4.1Performance Metrics

- Mean Squared Error (MSE):

The average of the squared differences, or mean of the squared difference between the approximate and real values, is called the mean squared error, or MSE. It is calculated in this way:

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

where yi are the true values and $\hat{y}_i$ are the predicted values from the model.

- ■ Result: 0.003867091456151034
- ■ Interpretation: The fact that the average squared error (MSE) is relatively minimal suggests that the model's predictions are reasonably close to the actual value.

- Mean Absolute Error (MAE):

MAE measures the average magnitude of errors in a set of predictions, disregarding direction. It is defined as the average over the test sample of absolute differences between forecast and actual observations, where each difference has equal weight.

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

- ■ Result: 0.030317992240630046
- ■ Interpretation: With an MAE of 0.1306, the model's average forecast error is 0.13 units off the actual value. This tiny inaccuracy indicates a high degree of forecast accuracy.

- Root Mean Squared Error (RMSE):

RMSE is the square root of the mean of the squared errors:

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

- ■ Result: 0.062185942592767975

- Interpretation: The model projections are fairly near to the actual values, as seen by the RMSE of 0.1867, which indicates that the typical prediction error is approximately 0.19 units, which is a very low figure.

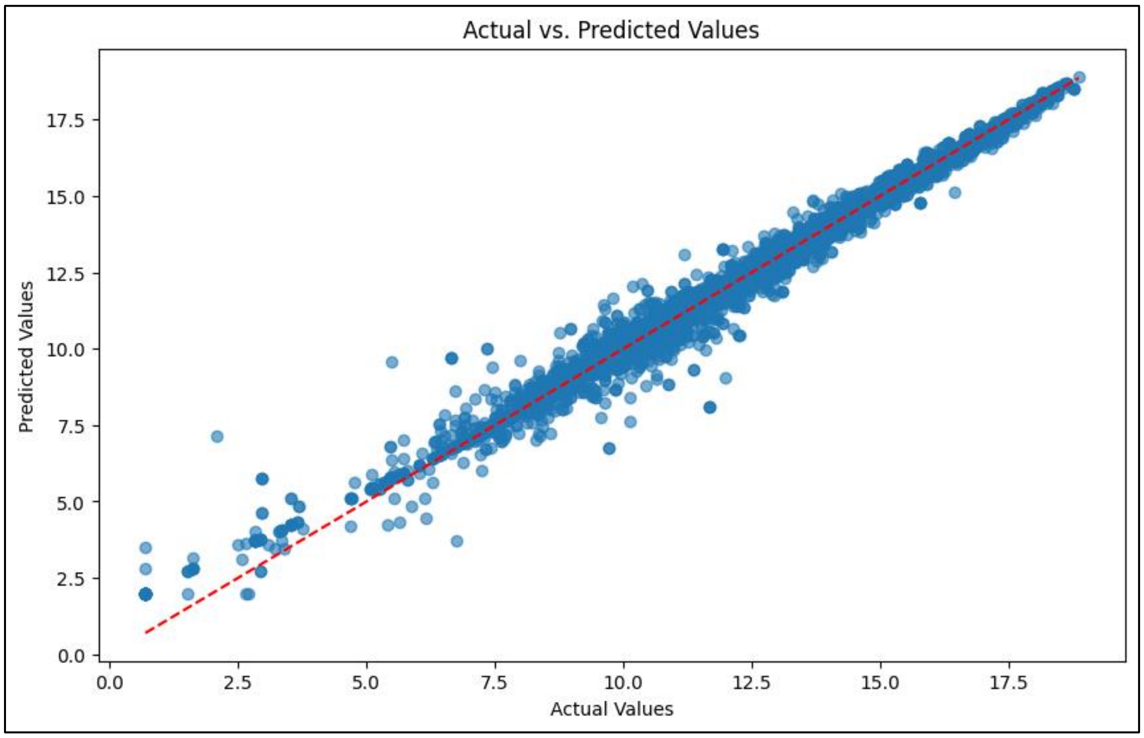
- R-squared (R^2): R^2 indicates how well the model uses the proportion of variance to predict samples that have not yet been seen.

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y_i})^2}{\sum_{i=1}^{n}(y_i - \overline{y})^2}$$

where $\overline{y}$ is the mean of the observed data.

  - Result: 0.9927655222966913
  - Interpretation: With an R2 value of 0.9431, the target variable's variation is roughly 94.31% explained by the model. Such high scores would indicate that the data's underlying patterns are being well-captured by your model.

- The figure below shows how well the predicted values is to the actual value for the MLP.



4.2 Data Splitting and Instances Used

4.2.1 Total Instances:

- Every data point is represented by 5602 rows and 433 columns in the dataset final_df. There are a total of 2420064 data points.

4.2.2 Training and Test Set Split:

- The first split divides the entire dataset into a training and a test dataset in a 70-30 ratio, where 70% of data goes into training and 30% is used for testing.
- Test set: 30% of 5602 is 1680 instances.

- Training set (before validation split): 70% of 5602 is 3921 instances.

4.2.3 Further Split for Validation:

- The training data is divided again into creating a validation set.
- Validation set: 20% of 3921 is 785 instances.
- Training set (after validation split): 80% of 3921 is 3136 instances.