Coursework Task Sheet 2

Data Analytics Pipeline: Implementation and Development

CMP5366: Data Management and Machine Learning Operations

Birmingham City University

Kacper Popis – 23161791 – [kacper.popis@mail.bcu.ac.uk](mailto:kacper.popis@mail.bcu.ac.uk)

[Step 1 2](#_Toc136221459)

[Data Ingestion 3](#_Toc429896329)

[Data Pre-Processing 4](#_Toc983513599)

[Model Development 6](#_Toc493373504)

[Model Deployment 7](#_Toc1736841906)

[Model Monitoring 9](#_Toc1612851182)

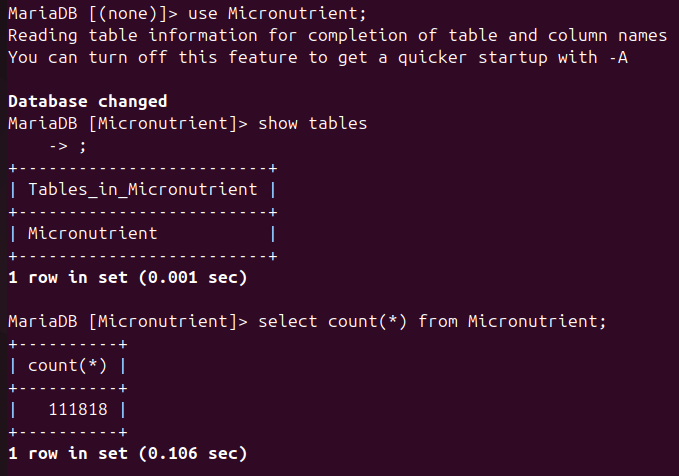
[References 11](#_Toc106078172)

# Step 1

## Data Ingestion

The data used for the Data Ingestion step of the pipeline is sourced directly from the website where the article is found, as the article’s publishers have made the data used publicly available. The files found on the website are a collection of 5 CSV files along with 5 different figures are available – but only the CSV files will be used. Those files have been downloaded into the Virtual Machine and are stored on the user’s home directory, from where the program responsible for applying pre-processing will access those files and convert them into a dataset, while removing any possible outliers, invalid data or missing data.

The approach for the data ingestion (as mentioned in Task Sheet 1) will be the **ETL** approach due to the large amount of data that is made available from the article. Currently, the focus is on the main CSV file (as shown in Task Sheet 1 – *Figure 3*) which is the S4 file, as this is the file that has the most references to the other CSV files from the article. This file is stored (as mentioned above) on the user's home directory – alongside the other CSV files from the article – which will then have pre-processing steps applied to it, and then is loaded into a newly created “Micronutrient” database to be accessed by the AI model. **ELT** is not being utilised as the data would have to be loaded into the databse and then later transformed using the pre-processing steps and once again loaded into the database, with new columns being created for the data that has been changed (like the country column which will be converted from alphabetical to numerical) will have to be added and the old columns would have to be dropped.



*Figure 1: Showing the dataset loaded into a database*

## Data Pre-Processing

The main CSV file contains no missing values for all the columns, however there is a clear disparity between the amount of times some of the countries have been noted down as the amount of data on some of the countries is fairly limited (not a lot of data is available due to data being collected over a short period of time. Additionally, there aren't many outliers within the data, and the ones present will only have a minor impact on the training of the AI model. There isn't an abnormal difference in the STD values, 75% values and the max values of each column, which means that the columns are missing outliers / there is a minor amount of outliers present in the columns.

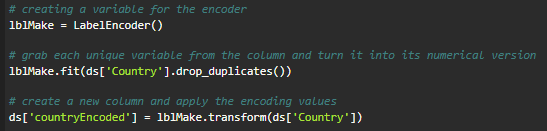
There are some redundant columns that had to be dropped from the dataset as they would provide no value to the predictions, or have similar data mentioned already. The columns that have been dropped are:

* Zone
  + The zone allocated to the country, which is redundant as the country is given
* ISO3
  + The 3 letter country code for the country, not needed as country is mentioned already
* Fortification
  + The column only contains 0 for each row, showing the column is redundant
* Tagname
  + The same content as Micronutrient column but an abbreviation



*Figure 2: Code snippet to show columns being dropped*

However, seen as the AI model is planned to be using regression models, the model will not be able to utilise non-numerical data for the training data. As such, columns containing text (such as Country, Micronutrient and Units columns) were purely alphabetical, which the model would be unable to register. These columns had label encoding applied to them to convert each unique instance of a value inside of each column into a numerical counterpart, for example, Afghanistan = 0, Albania = 1, Algeria = 2 would be the values converted with the encoder, and that data would be applied to each row appropriately inside of a new column “countryEncoded” - same is done for the other two columns, following the same principles. Once all of the steps have been applied to the dataset, then for clarity the column headers are renamed to make them easier to work with and to make the dataset look more clear and simpler, and then the dataset is ready for the next step of the pipeline plan. All these steps are found on the DAG file from Airflow.



*Figure 3: Code snippet showing the encoding of the “Country” column*

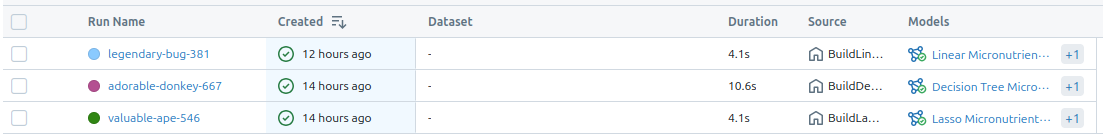
## Model Development

Due to the large quantity of data from the collection of CSV files, the AI model will have plenty of data to be trained on and the rows of data that will be selected to train the model will be selected randomly, rather than one after another. The way the test data and training data will be split will be in an 80 – 20 split, where 80% of the data will be used to train the model, and 20% of the data will be used to test the model to give majority of the data as training data to develop a strong basis on predictions while reducing the bias. However, as mentioned previously, there is a disparity in the quantity of data stored related to each country which could potentially alter the prediction accuracy for countries with a smaller amount of data on them.



*Figure 4: Code snippet showing the split of data*

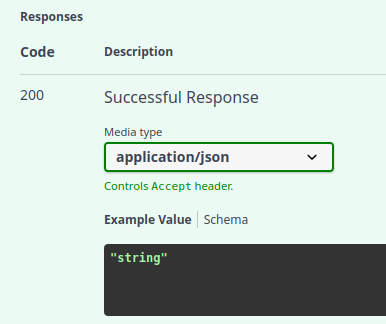
The AI model was planned to utilise 4 different regression techniques (linear, decision tree, random forest and lasso regression), but due to random forest regression struggling with the amount of data in the dataset, this regression technique will not be used for the model, the remaining 3 regression techniques will be trained the same, with an 80-20 split of data and a **random state** of 42 (where possible) to keep the training of the models as fair as possible.

*Figure 5: MLflow processes showing the different regression techniques*

## Model Deployment

Once each of the regression models have been trained on the dataset, the models are stored using the pickle python library, this allows for the models to be serialized (an object is converted into a character stream – a method of containing the necessary data to reconstruct the object) **[1]** and can later on be de-serialized (loaded) without having to retrain the model ever single time when its called across different sources like MLflow or Airflow. Currently, the main usage of the main usage of the pickle models is on MLflow, where an API is used (Fast API) to allow for user input via HTML. For testing purposes, **/docs** is being utilized as the endpoint gateway for testing if the:

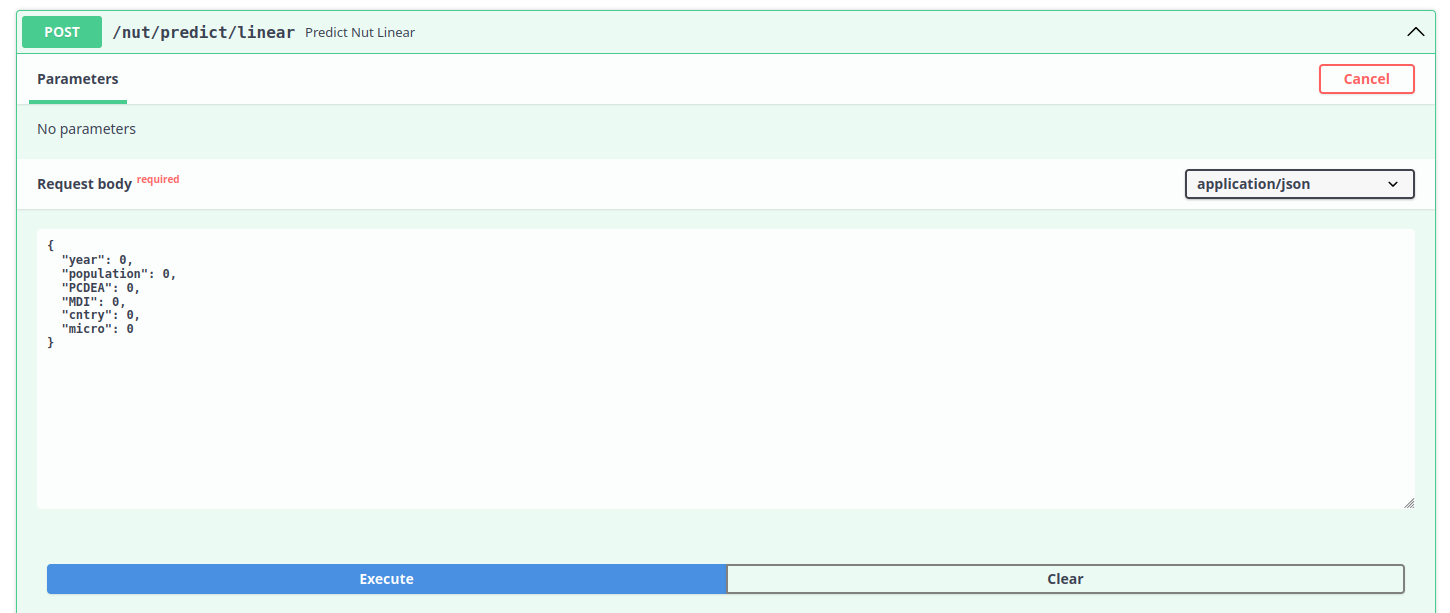
1. API works and is able to read the pickle models



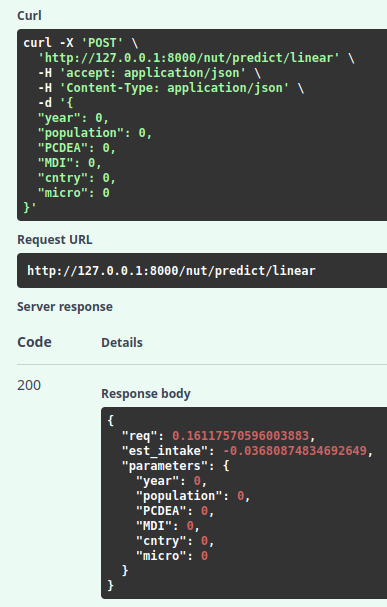
*Figure 6: Showing the response from the API*

When testing the endpoints for the 3 different models, all three send out a HTTP status code of 200 letting us know that the process has completed successfully, which will allow the user to interact with each of the models.

1. The user is able to interact with the models with their own data using a JSON input format

*Figure 7: Showing the POST method to be used to interact with a model*

For the user to interact with the model, they are required to complete a JSON input, which contains all the parameters used to train the models, once all 6 parameters have been satisfied, the user is able to execute the POST command and send the data, and shortly recieve an output back showing the predicted data (required micronutrient intake and estimated micronutrient intake)



*Figure 8: Showing the output of the POST method*

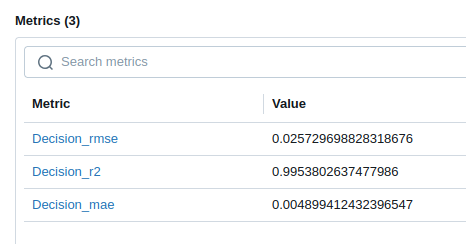
In the output, the user can see the predicted parameters, alongside the parameters used for the prediction.

## Model Monitoring

After the deployment of a model, one of the methods of checking the performance of the model would be to retrain the model after a certain period of time – this could be done every 2 months, or when there is a new major influx of data inside of the database. If the chosen method was to retrain the model when new data is added, this will require manual retraining of the model every time new data has been added – but this can also be a benefit as the model will be trained on the new data as soon as its added and can then provide better predictions, without a backlog of unused data. However. If the model was to be retrained every X amount of time, this would provide an automated approach and can then utilize services like Airflow for the automation of tasks, which these AI models use to be retrained on (with a schedule of every day) - when initializing the models, the parameter **random\_state** is missing as if it was present then the model would be trained on the same data every single iteration.

When each model is trained, their accuracy is also evaluated (where the X & Y test data is utilized) using the following factors:

* Mean Squared Error (MSE)
  + The average squared distance between the predicted values and actual values **[2]**
* Mean Absolute Error (MAE)
  + The absolute value of the difference between predicted and actual values **[3]**
* R2 Score
  + “The proportion of the variation in the dependent variable that is predictable from the independent variable” **[4]**



*Figure 9: Showing the stored metrics (from MLflow) of the decision tree model*

The decision tree model has the highest R2 score compared to the other two models, where both of the models average an R2 score of around 0.2, showing that the decision tree model is the most accurate model and should be the one used to get the most accurate predictions – this does not mean that the other two models are redundant as with an influx of data, the R2 score of them can potentially increase. As such, each model will have their metrics closely monitored, and will be inspected over time to see if there are any drastic changes to the accuracy metrics – if an unexpected drop in accuracy will happen then the model will have to be retrained, or potentially the data will have to be inspected for possible outliers, restarting the whole pipeline process.

# References

[1] - GeeksforGeeks (2017). Understanding Python Pickling with example. [online] GeeksforGeeks. Available at: https://www.geeksforgeeks.org/understanding-python-pickling-example/. [Accessed. 20th May 2025]

[2] - Frost, J. (2021). Mean Squared Error (MSE). [online] Statistics By Jim. Available at: https://statisticsbyjim.com/regression/mean-squared-error-mse/. [Accessed. 20th May 2025]‌

[3] - Acharya, N. (2023). Choosing Between Mean Squared Error (MSE) and Mean Absolute Error (MAE) in Regression: A Deep Dive. [online] Medium. Available at: https://medium.com/@nirajan.acharya777/choosing-between-mean-squared-error-mse-and-mean-absolute-error-mae-in-regression-a-deep-dive-c16b4eeee603. [Accessed. 20th May 2025]

‌[4] - Wikipedia Contributors (2019). Coefficient of determination. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Coefficient\_of\_determination. [Accessed. 20th May 2025]

‌