Hack4Good: IMPACT Initiatives

Date: 09.12.2024
Team: Gabriel Häfner, Diana Korotun, Annamalai Lakshmanan
Mentor: Teodora Bujaroska
Person Of Contact: Francesco Gizzarelli
Developed in the context of Hack4Good, 7th Edition

Contents

Abstract....2
Introduction....2
About IMPACT Initiative and Its Mission....2
Minimum Expenditure Basket (MEB)....2
Concern....2
Missing Data....2
Planning ahead....2
Challenge solved by our program....3
Solution....3
Used Data....3
How to Run the Code....3
Instructions....3
To Note:....4
Code....5
Pipeline structure....5
Set-Up....5
Preprocessing....5
Imputation....5
Admin Level 1:....5
Admin Levels 2 and 3:....6
Whole Syria (Admin level 0):....7
Prediction (Forecasting)....7
Results....8

Abstract

This document outlines a comprehensive pipeline developed for the IMPACT Initiative to address challenges in humanitarian decision-making, specifically targeting data gaps and prediction needs. The pipeline focuses on imputation and forecasting of commodity price data across various administrative levels in Syria, collected between 2016 and 2024. By leveraging statistical models like local regression, ARIMA, and GARCH, the system ensures accurate data imputation and reliable price predictions, enhancing the capacity for Minimum Expenditure Basket (MEB) analysis.

The solution provides a modular framework capable of processing raw data, imputing missing values, and generating forecasts for user-defined timeframes. Key functionalities include geographical and regression-based imputations, time-series forecasting, and comprehensive data visualization. Results are output as CSV files and PNG plots, enabling actionable insights for humanitarian planning. This pipeline can significantly improve data reliability and predictive capabilities, addressing critical gaps in resource allocation and crisis response.

Introduction

About IMPACT Initiative and Its Mission

The **IMPACT Initiative** is a leading organization dedicated to improving decision-making and planning in humanitarian and development contexts. Founded to bridge information gaps, IMPACT specializes in data collection, analysis, and dissemination to support evidence-based responses to global crises. Its primary goal is to enhance the capacity of humanitarian factors by providing them with reliable, actionable insights that enable effective interventions and resource allocation. The organization has agents in various crisis-ridden countries acting as informants collecting first-hand information from these areas. Some of the information includes various produce prices such as eggs, kerosene, and sugar prices. This information is then used to calculate the mean expenditure basket, a significant factor in the evaluation of the status of the various regions that IMPACTS investigates.

Minimum Expenditure Basket (MEB)

One of the key tools utilized by the IMPACT Initiative is the **Minimum Expenditure Basket (MEB)**. The MEB represents the minimum set of goods and services a household requires to meet its essential needs during emergencies or crises. These include food, shelter, water, healthcare, and education costs. By defining the MEB, humanitarian organizations can assess the adequacy of financial support provided to affected populations, ensuring that aid aligns with local market conditions and the specific needs of communities. It serves as a crucial benchmark for designing cash-based assistance programs and gauging household economic resilience.

Concern

Missing Data

In humanitarian operations, the absence of accurate and timely data poses significant challenges. Missing data often arises due to communication difficulties, such as disrupted infrastructure or logistical barriers, and reliability issues caused by conflicts or natural disasters. This data gap can hinder efforts to design effective interventions, leading to resource misallocation or delays in aid delivery. For instance, without complete data, organizations may struggle to accurately identify vulnerable populations or forecast future needs.

Planning ahead

The ability to predict and model data offers transformative benefits for humanitarian work. Predictive analytics allows organizations like IMPACT to fill data gaps by extrapolating from existing information, enabling more accurate and comprehensive insights. This approach not only enhances the reliability of assessments but also improves crisis preparedness, resource optimization, and decision-making under uncertain conditions. By leveraging predictive data tools, IMPACT can ensure that humanitarian responses are both timely and impactful, ultimately saving lives and fostering resilience in affected communities.

Challenge solved by our program

As such, Impact has provided with the following two challenges::

- develop a system that will **impute the missing data**
- develop a system that will **predict future data**

The challenge provided was given specifically within the context of Syria. Impact has a long history of recording information on Syria, and as such has been able to provide information since 105. However, due to the scarcity of information in 2015, we have decided to focus on information from 2016 to 2024.

Solution

Our solution entailed providing a pipeline that takes a .csv of the data file and returns the imputed and predicted data in the form of .csv files corresponding to the task they just performed - imputation and prediction. Moreover, our solution entitled a program which can perform both Imputation and prediction - separately and combined - over the preferred time range as indicated by the user.

Used Data Structure

The data we have worked on deals with data collected in Syria from 2015-03-21 to 2024-03-03, and contains columns such as uuid, country, round, round_date, date, month, year, admin1_code, admin1_label, urban_rural currency_code, sugar_price, green_tea_price, chicken_meat_price, laundry_soap_bar_price, dishwashing_soap_price, sanitary_pad_price, rice_price, etc. During preprocessing, the date, month, and year columns are condensed into the *year_months* column during preprocessing, the list of relevant products is extracted, and the admin levels are divided into the appropriate data frames in the code.

The organization "Impact" collects price data across multiple geographical regions, structured into four administrative levels. An instance of an Admin Level specifies an entity within a given admin level (e.g., "Idleb" is an instance of Admin Level 1). Admin level 1 corresponds to broad regions and admin level 2,3,4 to really small individual places. Each administrative level has time-series price data for various products (e.g. rice price, sugar price, etc.). The goal is to ensure that every instance within Admin Levels 1 to 3 has complete time series data for forecasting purposes. Admin Level 4 **i**s excluded from the data imputation process due to the lack of sufficient data.

For each Admin Level instance and each product, the average overall available individual timestamps are taken to generate one single time series.

How to Run the Code

Instructions

To run the program following steps:

- **clone** the repository
- ensure that all the **packages** listed in environment.yml are installed.
    - note that there might be an incompatibility between *pmdarima* and *numpy* in some Python environments, and may require downgrading the numpy version. For more on this issue check prime-GitHub issues[1].
- **upload** your raw data file as data/raw/dataframe_raw.csv.
    - If you wish to change the address where the raw data is uploaded from, it can be done by changing the constant impact/set_up/addresses_constants.py/ADRS_RAW_DATA
    - The program support .parquet and .xlsx. However, there were few tests done with the former, and .xlsx was taking almost an hour to load, as opposed to half a minute with CSV.
- **Adjust Parameters** in impact/set_up/constants.py as you wish to run the pipeline. The adjustable parameters include:
    - ADMIN_LOCATIONS: a dictionary of admin level which will be **imputed**
    - PRODUCTS_LABELS: a list of products which will be imputed
    - PREDICT_ADMIN: a String value of the admin level that will be predicted
    - PREDICT_CITIES: a String of cities which will be predicted
    - PREDICT_PRODUCTS: a list of products that will be predicted
    - CHOSEN_MIN_DATE: a String in YYYY-MM format for the minimum month of the analyzed data.
    - CHOSEN_MAX_DATE: a String in YYYY-MM format of the maximum month of the analyzed data. If data is being predicted this value needs to be larger than the original most recent month of the raw data.
        - note, that at least one of the dates needs to be in range of the raw data. Otherwise, the program will request that you change their value.
- **Run main.py**
    - once run you will be prompted to choose the task that will be performed by the pipeline:
        - **1**: only imputation
        - **2**: imputation followed by prediction
        - **3**: only prediction
- **Results** can be found in the data/processed/<unix_timestamp> and the plots of the processed data can be found in data/plot/<unix_timestamp>

To Note:

- The pipeline was developed specifically for Syria data collected from 2015-03-21 to 2024-03-03. To run the **pipeline on a data set in a different region, with raw data from a time frame, different product list, or a different MEB product quantity** please change the new parameters accordingly in impact/set_up/labels.py as this file contains the description of the raw data.

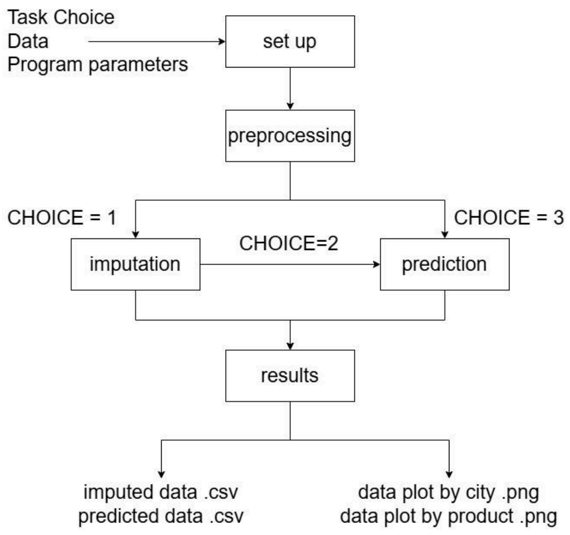
[1] https://github.com/alkaline-ml/pmdarima/issues/577

    - ➢ labels should only be updated if the primary raw data set changes. For changes in pipeline parameters please refer to constant.py as instructed above.
- ■ If you decide to perform **any kind of prediction,** please note that in softwares where constants are not overridden well - such as Jupiter Notebook - impact/set_up/constants.py/N_MONTH will not be updated per the difference between the user-chosen max_month and raw data max month. This might require the variable to be changed manually.
- ■ If you decide to **only perform prediction**, ensure that fully imputed data exists in a file of the following format: data/processed/imputed_admin<insert level>_label_full.csv
    - ➢ To generate a fully imputed file please run one of the pipeline's imputation options first.
    - ➢ To load imputed data from a different location update impact/set_up/addresses_constants.py/ADRS_IMPUTED_ADMIN<insert level>
- ■ When exporting results, note that by default the UNIX timestamp is *000000000*. As such, when the code runs with Jupyter Notebook or other setups where constant might not update properly, the results will be exported to data/processed/000000000 and data/plot/000000000.

Code

Pipeline structure

The pipeline is divided into 5 primary stages: set-up, preprocessing, imputation, prediction, and results. Where the pipeline takes in a .csv/.xlsx/parquet file address, user input, and parameters - defined in the set-up directory - and outputs 2 .csv files containing the imputation data and prediction data.



Set-Up

This stage is stored in the set_up directory. This directory contains the code that takes in the task number decision from the user with the following options:

- **1**: only imputation
- **2**: imputation followed by prediction
- **3**: only prediction

Where for choice 3 imputed data needs to be preloaded in the appropriate directory. Moreover, this stage will import the parameters used to process the rest of the pipeline and check their validity. The parameters are stored in the following files:

- **addresses_constants.py**: This folder contains the relevant addresses for the program. Including where the raw data, full imputed data, imputation, and prediction output addresses, along with the time stamp variable.
- **labels.py**: This folder contains the parameters describing the original raw data file. This file is to be changed ONLY if there are changes in the database itself. Otherwise, please change the appropriate parameters in the constant.py file. This file contains parameters such as the raw

minimum and maximum months, full product lists, and full region lists of the different admins.

- **constants**: This folder contains the parameters which are used for data processing by the pipeline. Its contents are checked with the content of the label.py to ensure that the pipeline doesn't attempt to hallucinate any regions, or products which did not exist before. This file includes admin location and product lists relevant for imputation and prediction, the minimum and maximum date the time range we wish to analyze, and the number of months which will be predicted.

Preprocessing

This stage is stored in the preprocessing directory. It is in charge of loading data into the panda's framework, filtering the columns so that the only price columns left are the relevant products, followed by the replacement of all None and blank cells with NAN - for improved statistical modeling further on, concatenation of the date, month, year columns into a singular year_month column for better clarity, removal of outlier data for better analysis performance, and finally the separation of the data into admin level, were admin level 0 is the monthly average of the products across all of Syria. At this stage, each admin is processed to contain a monthly of each region within that admin level.

Outlier Detection process:

In the beginning, a simple outlier detection is performed to filter out wrongly inputted data.

Calculate the 70% Interquartile Range (IQR) for a given month and product price across all admin levels combined. Then, define an outlier as any value lying beyond 10 times the 70% IQR.
This soft rule removes only extreme outliers. Hereby it preserves variability among different admin levels, which is especially relevant in admin levels with limited data that might still experience a massive price increase. Now remove detected outliers from the dataset.

Imputation

This stage is stored in the modeling_permutation directory.

Admin Level 1:

The following methods are successively applied to the admin level 1 dataset:

1. **Local Regression Imputation**:
   - For each instance and product, its time series is analyzed individually.
   - Small gaps in the time series are filled using polynomial regression of degree 3, capturing local trends and patterns.

2. **Geographical Imputation**:
   - For each Admin Level 1 instance, identify the four nearest instances using geographic coordinates.
   - Use the available price values from these neighbors and compute the average to fill missing values.
   - Repeat the geographical imputation process a second time:
     - In the first round, some instances may not have had any neighbors with valid data. If their neighbors are successfully imputed in the first round, they can propagate their newly imputed data in the second round.
     - Note: Instances imputed successfully in the first round are not modified in the second round.

3. **Global Regression Imputation**:
   - For any remaining missing values, perform a global polynomial regression across the available data.
   - This step ensures that even instances with no valid neighbors in both geographical imputation rounds are filled. Only if there is still not enough data available overall, NaN values will remain, which means that the imputation was not successful. This should only happen if one tries to perform imputation with solely data from the past few months. In that case, please also include more past data into the imputation.
   - This method is rougher but should only affect a small number of instances due to prior imputation steps.

4. **ARIMA-Based Refinement**:
   - Fit an ARIMA model to the completed time series for each instance and product price.
   - During earlier imputation steps (steps 1–3), track where missing values originally existed.
   - Use the ARIMA model to re-estimate those missing values based on the time points preceding and following the gaps.
   - Replace the previously imputed values with these ARIMA-predicted values, leveraging the model’s understanding of the time series structure.
   - In summary, steps (a)-(c) serve to fill all missing values in the first round. In the second round, an ARIMA model refines the imputed values based on the full dataset, improving the accuracy of the imputation. Cross-validation tests have demonstrated that this approach enhances the quality of the imputation.

Admin Levels 2 and 3:

Admin Levels 2 and 3 follow a similar approach but with some differences due to data sparsity and hierarchical dependencies.

1. **Local Regression Imputation**:
   - Perform a local regression imputation to fill small gaps in the time series for each instance. A second local regression is deployed that also incorporates a bit wider gaps but is still rather restrictive.

2. **Geographical Imputation**:
   - Each Admin Level 2 instance is linked to a corresponding Admin Level 1 instance, which now serves as the nearest neighbor for imputation. For admin level 3, the same applies with respect to admin level 2 linking.
   - When imputing a missing value for a previously imputed admin level, the fully imputed data from this corresponding Admin Level instance is used to fill in the gaps, ensuring all missing values are imputed.

Whole Syria (Admin level 0):

The dataset for overall Syria is generated by taking the average for each time period across all available data before the imputation for other admin levels is calculated. Only a small amount of data is missing and needs to be inputted. The imputation process consists of a local regression followed by a global regression.

Disclaimer:

For a few products, such as bread, only limited data is available. Therefore, the imputed data should be treated carefully. Additionally, since there are over 2000 time series in total, some may not be successfully imputed (e.g., negative price values may occur). We did not adjust these values, as they reflect the significant data scarcity that leads to uncertainty. Moreover, due to limitations in the machine learning models, unrealistic imputed data points may improve forecast performance compared to assigning those values to a constant.

Prediction (Forecasting)

This stage is stored in the modeling_prediction directory. For forecasting commodity prices, we selected GARCH (Generalized Autoregressive Conditional Heteroskedasticity) as our primary model due to its simplicity, and robust handling of volatility patterns in time series data. While we explored other models, including ARIMA and S-ARIMA, GARCH outperformed them in terms of accuracy for calculating the MEB, as shown by lower Mean Absolute Percentage Errors (MAPE). The ARIMA-based models sometimes struggle to capture the dynamic volatility of the data, which is crucial for forecasting commodity prices.

We chose not to use deep learning models like LSTMs or Transformers as these methods usually require high-quality ground truth data to generalize effectively, and in our case, the available data was limited and mostly imputed. This imputation process introduces uncertainty, making the data less suitable for deep learning methods, which are prone to overfitting in such scenarios. As a result, GARCH emerged as a better choice when considering reliability. The following work was done in this section.

1. The data is chosen in a way such that it considers only from the second year (due to the incomplete nature of the data) and uses the imputed data from the previous step.
2. Three models were tried: ARIMA, S-ARIMA, and GARCH with a hyperparameter to change between the models.
3. The optimal parameters for each of the ARIMA, S-ARIM,A and GARCH models were chosen using a greedy approach.
4. For each price + each city we have a separate model to forecast prices.
5. If MEB is required to be forecasted, we forecast every price, weigh it, and then calculate it.
6. We have a hyperparameter n_months to control how far we predict into the future. We do rolling forecasts i.e. only one step is predicted at a time and to predict the next step, we use the output of the previous step as well.

MEB forecasting for the whole of Syria should be performed with only admin 0 dataset. It doesn’t consider the rest of the admin_levels (including level 1,2,3) as the imputation is performed later.

Results

This stage is stored in the results directory. It is the final stage of the pipeline where that data is output into two CSV files -imputed data and predicted data - along with various PNG files.

The CSV files are stored in the data/processed/<UNIX_Timestamp> directory where for better clarity the predicted data and imputed data are completely separated. The files combined contain all the data within the parameter range request in constant.py. Moreover, the exported files correspond to teh task choice. This means that for the imputation-only task, imputed data only will be exported. Meanwhile, for the prediction-only task, only the predicted data will be exported, and when both tasks are performed, both CSV are exported. Furthermore, the exported files will also be separated by admin level, as data was processed on different levels of admin data.

Lastly, all the data that had been generated will be plotted into PNG files stored in data/plot/<UNIX_timestamp>. These plots are divided into two types - by product and by region. This means there is a plot for each processed product where the prices of that product are displayed across different regions. The second plotting type displays different graphs by different regions, with each region demonstrating the price of different products in that region. As such imputed data and predicted data are displayed within the same graph, while different admins would always be displayed in different plots.