Technical Appendix: Assessing Prophet automated change point detection method ability to recognise NOx, PM2.5 and O3 fluctuations produced by COVID-19 lockdown policies

SUMMARY:

This technical appendix guides the reproducibility of the results and work done for the Extended Research Project: Assessing Prophet's automated change point detection method's ability to recognise NOx, PM2.5, and O3 fluctuations produced by COVID-19 lockdown policies. It contains a first high-level overview of the process, the details of the data inputs, methods and information about the most important hyperparameters used. Ultimately, it contains information about the coding repository with an overview of its use.

GITHUB REPOSITORY: https://github.com/tgarciar/ERP_ProphetACPDM_Pollutants

OVERVIEW OF THE PROCESS

This research sought to understand the ability of the Prophet forecast model, specifically the automated change point detection method that comes with it, to detect changes in pollutant trends (NOx, PM2.5, and O3) during COVID-19 lockdown policies in Manchester, UK. In addition, it aimed to understand the practical effect of using different weather normalisation methodologies to understand how these decisions affect the later use of Prophet. The process consisted of three practical stages: (1) Data collection and EDA, (2) Weather Normalisation, and (3) Conducting experiments. FIGURE 1 illustrates the process at a practical high level.

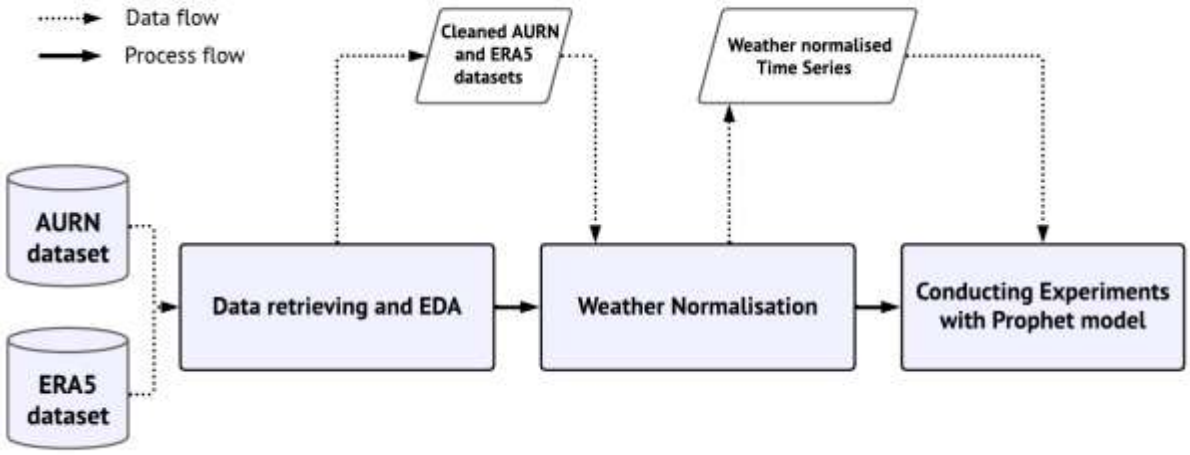

FIGURE 1: General process Extended Research Project

DATA INPUTS

The project used two data sources to obtain the information needed for the weather normalisation and Prophet experiments.

- **AURN Dataset:** Hourly measurements of pollutant concentrations (NOx, PM2.5, O3) and modelled meteorological variables (Temperature, Wind Direction, Wind Speed) at five stations near Manchester. The data on AURN stations was obtained directly from the DEFRA (Department for Environment, Food & Rural Affairs, UK) website and is in R format. Data from January 2016 to December 2023. TABLE 1 summarises the information from the stations. Data link: https://uk-air.defra.gov.uk/interactive-map

TABLE 1: AURN stations information. Source: ERP Report

Station	Lat/Lon	Background	Pollutants available
Manchester Piccadilly	53.48 / -2.24	Urban	NOx, PM2.5, O3
Manchester Sharston	53.37 / -2.23	Suburban	NOx, O3
Salford Glazebury	53.46 / -2.47	Rural	NOx, O3
Salford Eccles	53.48 / -2.33	Urban Traffic	NOx, PM2.5
Bury Whitefield	53.56 / -2.29	Urban Traffic	NOx

- **ERA5 Dataset:** Hourly meteorological data for the Manchester area, including Surface Pressure, Relative Humidity and Boundary Layer Height. Data from January 2016 to December 2022. Data is available in a csv format and shared in the repository. Data link: https://www.ecmwf.int/en/forecasts/dataset/ecmwf-reanalysis-v5

EXPLORATORY DATA ANALYSIS (EDA)

The EDA process consisted of performing different analyses to improve understanding of the pollutants. This analysis was done at each station and was intended to profile the behaviour of the pollutants. The GitHub repository has multiple EDA images presented as results for comparison.

WEATHER NORMALISATION:

This process aims to create the weather normalised (WETNOR) time series that will be used as the primary input for the Prophet experiments. The weather normalisation process has two stages. The first one consists of training a machine-learning model with the collected data. For this process, the dependent variable is the pollutant concentration, and the independent variables are the other variables already mentioned in the data input. In the second part, several resamplings of the variables are made, and the model trained in step 1 is used to predict the time series. Finally, the results are averaged, and the result is the WETNOR time series.

There is a debate on two questions: Which is better, intensive model tuning or non-intensive model tuning? Which resampling approach is better?

Given these questions, there are 4 combinations that were investigated in this research:

1. WETNOR pollutant time series with intensive model tuning with ALLVAR resampling.
2. WETNOR pollutant time series without intensive model tuning with ALLVAR resampling.
3. WETNOR pollutant time series with intensive model tuning with METVAR resampling.
4. WETNOR pollutant time series without intensive model tuning with METVAR resampling.

The following models were used for the model tuning process. TABLE 2, TABLE 3, and TABLE 4 show the parameters used in the hyperparameters grid.

- **XGBooster**, **Random Forest with RandomizedSearchCV:** Intensive model tuning. TABLE 2 and TABLE 3 show the grid parameters.
- **Normet-AutoML:** Non-Intensive model tuning. TABLE 4 display the grid parameters.

TABLE 2: XGBooster - Principal hyperparameters Grid. Source: ERP Report

Hyperparameter	Interval
Number estimators	[100-500]
Max depth	[5-75]
Max leaves	[100-1000]
Learning rate	[0.001-1]
Reg. Alpha and Lambda	[0.001-10000]

TABLE 3: Random Forest - Principal hyperparameters Grid. Source: ERP Report

Hyperparameter	Interval
Number estimators	[50-500]
Max features	[“sqrt”,”log”]
Max depth	[5-75]
Bootstrap	[“True”]

TABLE 4: Normet-AutoML - Principal hyperparameters Grid. Source: ERP Report

Hyperparameter	Value
Time Budget	90
Metric	r2
Estimator list	[“rf”,”xgboost”]
Task	regression
Split method	random
Number resampling	1000

Resampling approaches:

- **ALLVAR**: 1000 Resampling of all independent variables except the UNIX variable.
- **METVAR:** 1000 Resampling of only the weather independent variables.

EXPERIMENTS – PROPHET:

This research had three experiments:

Experiment 1: Understanding how weather normalisation strategies impact Prophet.

Evaluate how different methods in the weather normalisation process impact Prophet's ability for detect the trend, and changes of pollutants concentrations from the Manchester Piccadilly station. Flexible trend with weekly possible changepoints to compare results.

Experiment 2: Weekly changepoints - Flexible trend

Assess Prophet ability to detect changes during the 2020-2021 lockdown policies using a flexible approach - weekly posible changepoints.

Experiment 3: Given changepoints - Inflexible trend

Evaluate Prophet ability to detect changes during the 2020-2021 lockdown policies using an inflexible approach - given changepoints on key dates.

Experiments 2 and 3 follows this process:

1. Modelling in Prophet using the ALLVAR_RandomSearch WETNOR time series.
2. Creating a new dataset with the Prophet trend results. Linearly interpolate the results to regularise the gaps between changepoints and avoid overestimated changes.
3. Calculate the rate of change and the accumulated rate of change sum (CUMSUM) using the changepoints and the Prophet trend values.
4. Calculate the slope and percentage (%) changes for the four periods studied(TABLE 5).
5. Visual inspection of changes in the slope on dates close to the different lockdown policies investigated (TABLE 6).

TABLE 5: Periods of study. Source: ERP Report

Period	Start	End
Period 1: Pre-Covid19 and First Lockdown	01/01/2020	10/05/2020
Period 2: Reopening	10/05/2020	05/11/2020
Period 3: Second and Third Lockdown	05/11/2020	22/02/2021
Period 3: Reopening	22/02/2021	01/07/2021

TABLE 6: Specific lockdowns to investigate. Source: ERP Report

Policy	Start	End
First Lockdown	26/03/2020	10/05/2020
Non-essential shops reopen	15/06/2020	-
Eat out to help out	03/08/2020	-
Second Lockdown	05/11/2020	02/12/2020
Third Lockdown	06/01/2021	22/02/2021
Non-essential shops reopen	12/04/2021	-

TABLE 7: Hyperparamters used in Prophet for Experiment 2 and 3.

Hyperparameter	Used	Why
growth	“linear”	To use Piecewise Linear Model
interval_width	1	To use all the time series
changepoint_range	1	To use all the time series
changepoints	Weekly or given changepoints	Flexible or inflexible, it depends on the Experiment
freq	“h”	To get hourly data results

CODE REPOSITORY – GITHUB

This extended research project has a code repository with all the significant pieces of code used during it. Although this paper worked with different software's, such as Google Collab, Jupyter Notebook, and Visual Studio, among others, what has been done is a repository with Python scripts that facilitate the understanding and follows the sequence of the research. For this, the code was cleaned, annotated, organized, and automated to follow the main parts of each process.

The first thing to understand is that it is critical to read the README file. The user will find important information on how to install the repository, the virtual environment, and the necessary libraries in it. It also contains the licenses for both data sources. The repository is organised as follows:

- The main directory includes folders and Python scripts. FIGURE 2 shows a screenshot of the directory.

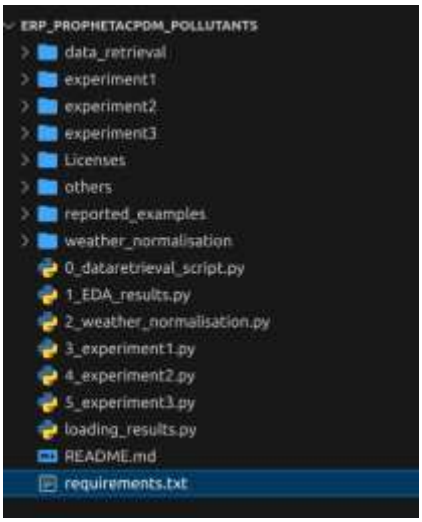

FIGURE 2: Main directory of the repository

- The Python scripts that are visible have a number that signifies the order in the ERP process. For example, data_retrieval has the number 0 because it is the first step in the ERP process. It is key to understand that there are some previous installation steps before running the ERP. These Python scripts on the main directory are meant to be the easiest way to run the process.

- The folders, which are part of the ERP process, contain different scripts inside of them. In FIGURE 3, for example, it can be seen that the experiment2 folder has a saved results folder and two Python scripts with initials 41 and 42. These two scripts together are part of the 4_experiment2 process. In this case, the user could run it from the main directory called python 4_experiment2.py, and the whole process would run. If users want to be more curious and go deeper into the code, they can open 41_experiment.py and run it. This flexibility allows the users to delve into the part of the process that catches their attention. Inside the scripts, users can also change hyperparameters.

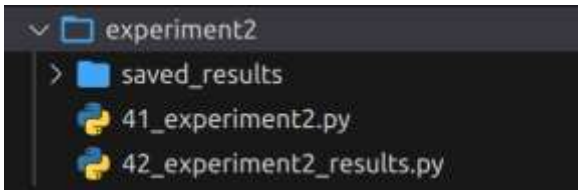

FIGURE 3: Files inside 4_experiment2 folder

- The scripts have some saved results, as there are processes, like the weather normalisation process, that took more than 60 hours to run. In that case, the user can decide whether to use the results from the ERP (more convenient) or try to run the weather normalisation and see the results. This is specified inside the Python scripts.
- When running the different Python Scripts, folders will appear containing the results of the process, and of the ERP.

EXTRA: PROPHET TECHNICAL DETAIL

Created as an appendix for the ERP report but with an accurate and short explanation of how Prophet works. Useful to share in the technical appendix.

This technical detail is based on the Prophet's documentation (Prophet, 2024), and the Prophet research paper (Taylor and Letham, 2017) .

Prophet is an open-source forecasting model that handles complex time series. A characteristic of Prophet is that it is particularly effective when managing seasonal patterns, holidays, and anomalies in data, such as outliers or missing data.

Prophet models a time series $y(t)$ as an additive composition of three main elements:

$$y(t) = g(t) + s(t) + h(t) + \epsilon(t)$$

- $g(t)$: trend function modelling changes in the time series
- $s(t)$: seasonal components
- $h(t)$: holiday effects
- $\epsilon(t)$: normally distributed error term.

Trend component:

The trend $g(t)$ is modelled using either a piecewise linear model or a logistic growth model. In this research, the piecewise linear model was used as the pollutants data had an overall linear trend.

Piecewise Linear Model (Linear Trend with Changepoints): The trend is represented as a series of linear segments with changes in slope at possible changepoints.

$$g(t) = (k + a(t)^T \delta) \cdot t + (m + a(t)^T \gamma)$$

- k: initial growth rate.
- m: offset parameter (intercept).
- δ and γ: vectors of adjustments at the changepoints (rate and offset). *
- $a(t)$: indicator function activated after each changepoint. *

* The complexity on how δ, γ, and $a(t)$ work together is well explained in the research paper (Taylor and Letham, 2017).

Seasonality and holiday components:

The seasonality is modelled using Fourier series to provide a flexible model with periodic effects. On the other hand, the holiday effects are modelled by assigning specific periods during which the time series shows anomalous conduct. The paper provides details on the mathematics behind this.

Automated Changepoints in Prophet:

Prophet identifies potential change points across the historical data based on the change's magnitude and the trend's slope. A practical consideration on this point is that there are two ways of giving the potential changepoints: (1) assigning a number of changepoints and (2) giving the dates of the changepoints.

After this, a changepoint matrix is created, and these values are used within the piecewise linear model. Prophet uses an L1 Lasso regularisation to prevent overfitting. Moreover, Prophet uses a gradient-based optimisation algorithm to minimise the loss function, which includes both the fitting error and the regularisation penalty. During this process, the model balances the need for flexibility in the trend with the need to avoid overfitting with many small changepoints activated.

REFERENCES:

Prophet (2024) *Prophet - Documentation / Quick Start*, *Prophet - Python Documentation*. Available at: http://facebook.github.io/prophet/docs/quick_start.html (Accessed: 16 August 2024).

Taylor, S.J. and Letham, B. (2017) ‘Forecasting at scale’. Available at: https://doi.org/10.7287/peerj.preprints.3190v2.