README

Replication Package (Version 2) for

"Detecting Edgeworth Cycles" by Timothy Holt, Mitsuru Igami, and Simon Scheidegger

(Forthcoming in *The Journal of Law and Economics*)

November 14, 2023

The Digital Object Identifier (DOI) of this replication package is: 10.5281/zenodo.10126406

1. Overview

This replication package contains the data and the code to generate the paper's main results in Table 2 ("Performance of Automatic Detection Methods") and Figure 2 ("Gains from Additional Data").

After installing the relevant software and setting up the computational environment (**Steps A through D** in Section 6 below), the replicator should expect the code to run for *less than 20 minutes* to "train and test" all of the 10 methods using a regional/country dataset of choice—on a regular desktop personal computer and under some *basic, minimalist settings*.[1] The total amount of time to perform the same task in all three datasets should be *less than an hour*.[2] See **Steps E through K** in Section 6 below for detailed instructions.

Additionally, this package also allows the users to apply these "trained" models to new/external data of their choice. As an example of such a "new" dataset, we include the entire German dataset available at the time of our research (2014:Q4–2020:Q4), including both manually labeled and unlabeled subsamples. See **Steps L and M** in Section 6 below.

Finally, this package includes tools to facilitate the acquisition and formatting of the most recent data from Germany, which is updated every day on the Tankerkoenig website (at the time of the

[1] The "basic, minimalist settings" in the above refer to (i) a single run of "sample split" (i.e., steps 2 and 3 in the paper's section 4.3) and (ii) the training of Methods 8 and 10 (LSTM and E-LSTM) for only 10 "epochs" (see Online Appendix A.1).

[2] A *complete* replication of Table 2 (subject to stochastic differences due to the randomness in the sample-split procedure) requires a considerably larger amount of time or computational resources (up to approximately 1,000 times) because our reported results are based on (i') the repetition of the same steps *101 times* (see step 4 in the paper's section 4.3) and (ii') the training of Methods 8 and 10 for *100 epochs*.

preparation of this replication package). See **Steps N through Q** in Section 6 below.

2. Data Availability

The paper uses five sources of data. The first three sources are *FuelWatch*, *FuelCheck*, and the Market Transparency Unit for Fuels of the *Bundeskartellamt*, which publish data on retail gasoline prices in Western Australia (WA), New South Wales (NSW), and Germany, respectively:

https://www.fuelwatch.wa.gov.au

https://www.fuelcheck.nsw.gov.au

https://www.bundeskartellamt.de/EN/Economicsectors/MineralOil/MTU-Fuels/mtufuels_node.html

The fourth and the fifth sources are the Australian Institute of Petroleum and the Argus Media group's *OMR Oil Market Report*, which publish regional wholesale prices in Australia and Germany, respectively.

https://www.aip.com.au

https://www.omr.de/

The first four sources make their data publicly available at no cost. The fifth source is a commercial data provider and has allowed us to use the data for academic purposes, with a permission to publish our derived statistics (profit margins) in this replication package. Hence, *we certify that the authors of the manuscript have legitimate access to and permission to use the data used in this research*.

3. Dataset List

(A) Main Data

The following data files are included in the replication package:

`wa_label_db.json`,

`nsw_label_db.json`, and

`german_label_db.json`.

These files contain region/country-specific data from WA, NSW, and Germany, respectively.

For each gasoline station in each calendar quarter in each region/country, these files record the following information: (i) the manual-classification labels by our Research Assistants (RAs)

("human_labels"), (ii) daily retail prices ("price"), (iii) daily retail-wholesale margins ("detrended_price"), (iv) year, month, and date of its first day ("start_date"), (v) the number of days ("days"), (vi) postal code ("post_code"), (vii) longitude and latitude ("geotag"), and (viii) brand name ("brand").

The "human_labels" part codifies our RAs' decisions as "0" (not cycling), "0.5" (maybe cycling), and "1" (cycling). The prices and margins are measured in Australian cent (WA and NSW) or euro cent (Germany). The contents of the other variables should be self-explanatory.

(B) Supplementary Data

Additionally, we offer the entire German data for 2014:Q4–2020:Q4 (including the manually *unlabeled* part, as well as our Method 10's automatic classification results) that we examined in section 7.3 of the paper ("Exploratory Data Analysis"):
`ALL_detrended_price_windows.json`.

4. Computational Requirements

Software: Our code requires Python (version 3.8 or later).

Hardware: The code was last run and tested on a regular desktop personal computer with a 4-core Intel-based CPU, Windows 8.1 Pro operating system, and 32 GB of RAM.

Runtime: The time needed to reproduce the "basic, minimalist" outputs underlying Table 2 (**Steps E through J** in Section 6 below) is less than an hour. A "complete" replication requires up to 1,000 times more computational time/resources (see footnote 2 on page 1).

- The replication of Figure 2 (**Step K**) requires seven additional runs for each dataset.
- The runtime for the "new/external data" part (**Steps L and M**) depends on the size of the new dataset. For example, the "example" dataset is 10 times larger than the baseline (manually labeled) German data, and hence its analysis requires a proportionally longer runtime.
- Finally, the runtime for **Step P** will be at least several hours (if the user tries to clean the entire up-to-date German data that are newly downloaded).

5. Description of Main Programs

First, the setup of a suitable environment uses `replication_conda_environment.yml` to install required Python packages and their associated dependencies (see Section 8 below).

Second, a random subsample of station-quarter observations (including our RAs' manual labels) can be plotted for visual inspection by the script `plot_sample.py`.

Third, the main scripts for training and testing the 10 methods are:
`run_parametric_models.py`,
`run_rf_model.py`, and
`run_lstm_model.py`,
which implement Methods 1–7 (and their variants), Method 9, and Methods 8 and 10, respectively.

- They take the data file of the replicator's choice (one of the three datasets) as inputs.
- They display summary results on screen (the terminal window) and record more detailed outputs in new CSV files:
  `parametric_model_log.csv`,
  `random_forest_model_log.csv`, and
  `lstm_model_log.csv`.
- New lines are appended to the same CSV files when the scripts are run again.
- When the main scripts' '`save_model`' option is set to `True` (default), the models' parameter values are saved in PKL files in new subdirectories
  `/detecting_edgeworth_cycles/pretrained_models/(method)/(region/country)/(training set hash)/`.

Fourth, the main scripts for applying any pre-trained model to new/external datasets are:
`parametric_classify_external_data.py` and
`nonparametric_classify_external_data.py`,
which implement Methods 1–7 (and their variants) and Methods 8–10, respectively.

Fifth, the main scripts for formatting newly downloaded data from Germany are:
`de_rawdata_parse_postal_region.py` and
`convert_price_window_json_csv.py`.

6. Instructions to Replicators

The replicator can follow these steps to obtain the main results.

I. Preparation

A) **Download and Extract the Project Files.**
Download and extract the zip file that contains scripts and data. The current version of the replication package is enclosed with this ReadMe file.

B) **Extract and Move the Data Files.**
The data files in
`/detecting_edgeworth_cycles/label_databases/label_data_files.zip` and
`/detecting_edgeworth_cycles/label_databases/ALL_detrended_price_windows.zip`
need another extraction step. The extracted JSON files should be moved to the subdirectory
`/detecting_edgeworth_cycles/label_databases/`

C) **Download and Install Miniconda or Anaconda.**
The easiest way to create a suitable computational environment in Python is by using Miniconda or Anaconda:
https://conda.io/projects/conda/en/stable/user-guide/install/
https://docs.anaconda.com/free/anaconda/install/
Skip this step if either of these distributions has already been installed on your computer.

D) **Prepare a New Environment.**

- Navigate to the main folder of this replication project by entering:
  `cd C:¥Users¥...¥detecting_edgeworth_cycles¥`
  Note: The path to the main folder in the above (`C:¥Users¥...`) is just an example. Modify this part according to your local directory structure.

- Create a new environment (set of installed packages) for the project.
  `conda env create -f replication_conda_environment.yml`

II. Activation

E) **Activate the New Environment.**
You will need to *start from this step* every time you open a new terminal/session. Enter
`conda activate edgeworth_replication_env`

III. Data Visualization

F) **Plot a Random Subsample of Station-Quarter Observations.**

Enter

```
python plot_sample.py arg1 arg2
```

where `arg1` is either `wa`, `nsw`, or `de` (i.e., region/country), and `arg2` is a positive integer such as `5` (the sample size). Only one plot is displayed at a time. You will need to close one to see the subsequent plot.

IV. Training, Testing, and Saving

G) **Train and Test Methods 1–7.**

Enter

```
python run_parametric_models.py arg1 arg2
```

where `arg1` is one of `{wa, nsw, de}` and `arg2` is one of `{PRNR, MIMD, NMC, MBPI, FT0, FT1, FT2, LS0, LS1, LS2, CS0, CS1, WAVY, all}` (i.e., Methods 1–7 and their variants).[3] Set `arg2` to `all` to implement all of them.

H) Train and Test Method 9 (Ensemble Random Forests).

Enter

```
python run_rf_model.py arg1
```

where `arg1` is one of `{wa, nsw, de}`.

I) Train and Test Methods 8 and 10 (Basic and and Ensemble LSTM).

Enter

```
python run_lstm_model.py arg1 arg2 arg3
```

where `arg1` is one of `{wa, nsw, de}`, `arg2` is a positive integer such as 10 or 100 (the number of training epochs),[4] and `arg3` is an indicator to choose between basic LSTM and

[3] The first four options of `arg2` (`PRNR, MIMD, NMC, MBPI`) correspond to Methods 1–4, respectively. `FT0` and `LS0` are Methods 5 and 6, respectively; `FT1` and `LS1` are their peak-based variants; `FT2` and `LS2` are their Herfindahl-Hirschman-Index-based variants. `CS0` is Method 7; `CS1` is its integral-based variant; `WAVY` is its discrete variant. See Online Appendix A.1 for details.

[4] A training epoch is a single run through the dataset. Model fit (prediction accuracy) will increase with the number of epochs until over-fitting is achieved. Less than 10 is not recommended; we set it to 100 to obtain our main results in the paper.

E-LSTM: `0` (Method 8) or `1` (Method 10).

J) Save the Trained Parameter Values.

To save the trained/estimated/optimized parameter values, go to the "parameters" section of these Python scripts (in G, H, and I above) and set `save_model = True` (default).

K) Change the Fraction of Training Subsample.

To replicate Figure 2 ("Gains from Additional Data") in the paper, go to the "set parameters" section of these Python scripts (in G, H, and I above) and change `train_fraction` from `0.8` (default) to `0.001`, `0.01`, `0.05`, `0.1`, `0.2`, `0.4`, and `0.6`, respectively.

V. Applying Pre-trained Models to New/External Data

L) Classify New Data by Methods 1–7 (Pre-trained in Our Data).

First, ensure that your new dataset (i) is in either JSON or CSV format and (ii) has the same structure as the data files listed in **Section 3** above (in the case of JSON format) or an output CSV file of '`convert_price_window_json_csv.py`' (in the case of CSV format—see **Step Q** below for the explanation of this Python script). Not all fields need to be present, but at least a price series and a unique-station-identifier column must be there.[5]

Second, go to the '`parametric_classify_external_data.py`' script and set the `external_data_path` parameter to *the path and file name of your dataset*. The default choice is `/label_databases/ALL_detrended_price_windows.json`—the example dataset.

Third, enter

```
python parametric_classify_external_data.py arg1 arg2
```

where `arg1` is one of `{wa, nsw, de}` and `arg2` is one of `{PRNR, MIMD, NMC, MBPI, FT0, FT1, FT2, LS0, LS1, LS2, CS0, CS1, WAVY, all}`. These two arguments specify the method(s) and the dataset used for pre-training it/them, respectively.

M) Classify New Data by Methods 8–10 (Pre-trained in Our Data).

First, ensure that your new dataset is in a compatible format (see **Step L** above).

Second, go to the '`nonparametric_classify_external_data.py`' script's "set parameters" section, and specify: (i) the path and file name of your dataset,[6] (ii) one of Methods 8–10 in `{`lstm_basic', 'rf', 'lstm_ensemble'}`, (iii) the *hash* of the desired pre-trained

[5] As an example of a new dataset, we include the entire German data that was available at the time of our research (2014:Q4–2020:Q4) in our preferred format and data structure.

[6] The default choice is the same "Germany 2014:Q4–2020:Q4" example dataset as in Step L.

model,[7] and (iv) the output file's name (with either JSON or CSV extension).

Third, enter

```
python nonparametric_classify_external_data.py
```

VI. Downloading and Formatting New Data from Germany

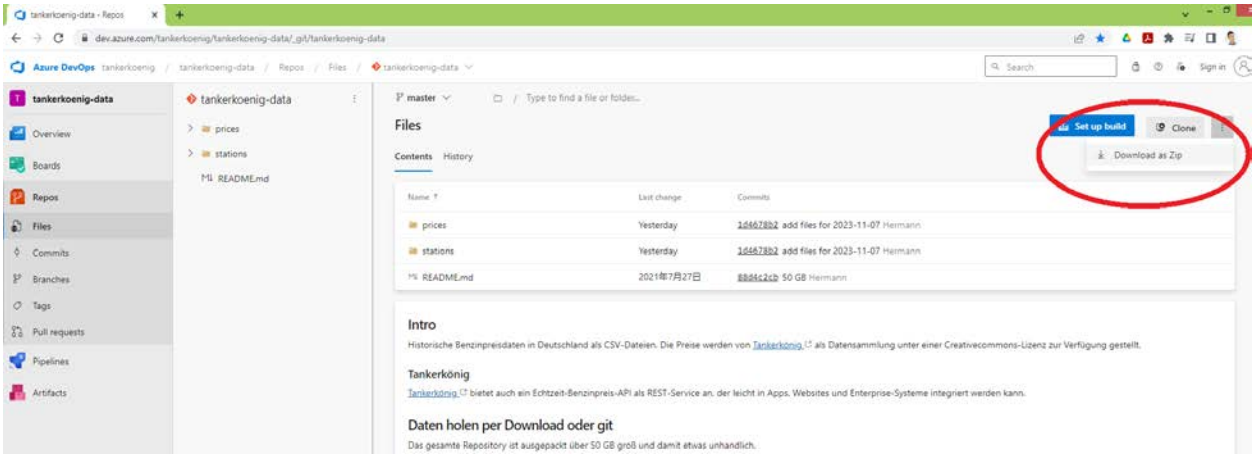
N) **Download New Data.**

First, download the Tankerkoenig raw data (updated daily) by either (i) entering

```
git clone https://tankerkoenig@dev.azure.com/tankerkoenig/tankerkoenig-data/_git/tankerkoenig-data
```

or (ii) copying the following URL to a web browser and choose "Download as Zip"

https://dev.azure.com/tankerkoenig/tankerkoenig-data/_git/tankerkoenig-data



Note the entire database is quite large (e.g., more than 80GB of disk space is needed to store all data files between 2014 and 2023). If you intend to use only a specific subsample, you might want to consider clicking on the "prices" and "stations" folders on their website and download specific year-month subfolders.[8]

---

[7] The "hash" is an alphanumeric string that identifies (a specific sequence of pseudo-random numbers that is used to generate the specific training subsample that is used to generate) the specific "trained model" (parameter values). The users can find it either in the execution log of "training and testing" (**Steps G through I**) or in the name of the subfolder that stores the parameter values (**Step J**).

[8] If you choose to selectively download specific subsamples, make sure to *replicate the Tankerkoenig website's folder structure on your local computer* before running our Python script in **Step P**. That is, the top folder should be named 'tankerkoenig-data' and placed immediately below the main directory of the replication project. Individual CSV files should be organized into 'prices' and 'stations' subfolders, which should nest 'year' and 'month' subfolders inside.

O) Adjust "Start Date" and "End Date" in Script to Match New Data.

Go to the `/framework/de_data_parsing` subfolder.

Open '`observation_windows_functions.py`'.

Go to Lines 200–202 and modify '`start_date`', '`end_date`', and '`sparsity_threshold`' to match your new (downloaded) data and your data-cleaning criteria. These three lines determine whether each station-quarter observation is processed (and included in the final dataset as a valid observation).

P) Clean and Organize the Data.

Format the data by entering

```
python de_rawdata_parse_postal_region.py arg1
```

where `arg1` is one of {`0, 1, 2, ...9, all`}. This argument specifies one (or all) of the 10 "postal regions" of Germany.

Three types of output files will appear in a new folder '`de_databases`':

(1) **Station Info Databases**

(`sta_info_db.json` and `sta_info_db.pkl`)

—contain information on individual gasoline stations;

(2) **Price Station Databases**

(`0_price_sta_db.pkl`, `1_price_sta_db.pkl`, …, `9_price_sta_db.pkl`)

—contain a serialized dictionary of gasoline-stations objects, each of which contains a list of prices and time stamps for a given station; and

(3) **Price Windows Files**

(`0_price_windows.json`, `1_price_windows.json`, …, `9_price_windows.json`)

—contain the daily average price at each station, in each of the 90 days of each calendar quarter (recall our basic unit of analysis is a station-quarter pair).[9] This is the main "new/external data" file to be analyzed in **Steps L and M** above. If these files are generated but empty, go back to **Step O** and make sure the data-cleaning criteria match your download files (and are not too stringent).

Q) Convert the Main Data Files from JSON to CSV (or Reverse)

If you wish, you can convert the "Price Windows Files" between JSON and CSV formats by entering

```
python convert_price_window_json_csv.py arg1
```

[9] If you download only a small subsample that is *insufficient to construct 90 consecutive days of price data* (organized by calendar quarter), the price-windows output files will be empty and become a source of errors in **Step Q**.

where `arg1` is *the path and file name of the target data file*. This script allows the users to (i) "import" their own new/external data (in CSV) to the JSON format that is compatible with our programs and (ii) "export" our/their data files (in JSON) to the CSV format, so that they can analyze these data in other software packages.

7. Description of Other Programs

(A) Training/Estimation Part

The package contains the following scripts in the `/main/framework/` subdirectory:

- `Estimation_Framework.py`: main code defining the parametric models and feature interfaces.
- `RF_Framework.py`: main code defining the random-forest model.
- `LSTM_Framework.py`: main code defining the LSTM models.
- `Label_Class.py`: defines the data structure underpinning individual observations, as well as collections of observations.
- `Model_Loader.py`: a convenient interface for loading data and splitting into training and test sets.
- `model_settings.py`: dictionaries and lists for some important default parameters to make the framework function.

(B) Formatting New German Data

The following scripts in `/main/framework/de_data_parsing/` subdirectory help the formatting and other handling of the most recent data from Germany.

- `sta_info_functions.py`: functions to parse station info files (excluding prices).
- `price_db_functions.py`: functions to parse station price files.
- `observation_windows_functions.py`: functions to create station window observations of quarterly average daily prices.
- `Station_Class.py`: data structure representing a single gas station with all relevant data.
- `Timer_Utility.py`: convenient utility for recording timing and performance in Python code.
- `parameters.py`: set of default parameters for parsing German raw data.

8. Python Packages Used

The following packages and their associated dependencies were used at the time of development:

```
matplotlib (3.7.1)
numpy (1.24.3)
pandas (1.5.3)
python (3.10.11)
scikit-learn (1.2.2)
scipy (1.10.1)
seaborn (0.12.2)
tensorflow (2.10.0)
```

9. Suggested Citations

If any part of this package becomes useful for your research project, please cite our research paper and the replication package:

(A) Holt, Timothy, Mitsuru Igami, and Simon Scheidegger. 2023a. "Detecting Edgeworth Cycles." *The Journal of Law and Economics*, forthcoming.

(B) Holt, Timothy, Mitsuru Igami, and Simon Scheidegger. 2023b. "Replication Package for: Detecting Edgeworth Cycles (Version 2)." Available on Zenodo: 10.5281/zenodo.10126406.