A Reproducibility information

A.1 Published Dataset

The published dataset describes the weekly average prices and demand of $M = 100$ product groups across $N = 10$ stores, for a period of 156 weeks. All values have been anonymized and scaled. The data file is available publicly on GitHub via https://github.com/aabdulaal/gerd. For any access-related problems, please contact the authors to request an email copy of the file. The following are the list of features included:

date_id : Integer representing the order of the week of a given sample.
location_id : Integer distinguishing the different stores.
product_id : Anonymized product label as a mix of letters and digits. The purpose of the letter characters is to approximate potential relevancies across products without using real values or names. B is for the brand, T is for the product type or category, C is unit count per product, M is for the product or unit size, and V is to separate different variations of the product.
economic_factor : A continuous number acting as a proxy for macroeconomic, seasonal, and trend variations.
price : The normalized price.
demand : The scaled demand.

From the dataset, the following features should be extracted in order to reproduce the study results:

R_1 : A 1-dimensional vector of the seasonal-week order. Obtained via iteratively subtracting 52 from values larger than 52 in the date_id column, until the highest value in the column is 52. This vector is encoded as a categorical vector of 8-bit integers.
R_2 : The economic factor, also a 1-dimensional vector.
S : A 1-dimensional vector of the label-encoded location id, in 8-bit integer format.
P^C : The prices unstacked into a 2-dimensional vector of size $(156 \times N) \times M$.
Q : The demand unstacked into a 2-dimensional vector of the same size as P^C.
$\check{P}, \check{P}^{LB}, \check{P}^{UB}$: Arbitrary assigned markups/markdowns and their limits of the same size as P^C and Q. All the values of $\check{P}$ are set to 1, while all the values of $\check{P}^{LB}$ and $\check{P}^{UB}$ are set to 0.85 and 1.20 respectively.

A.2 Other Public Datasets

The ORII dataset is available via the Univeristy of California, Irvine machine learning repository: https://archive.ics.uci.edu/ml/datasets/Online+Retail+II. There is also a copy on Kaggle. This dataset in its raw format is unsuitable for dynamic pricing research and requires further processing as follows:

Remove all samples with invoice number beginning with 'C', which indicates a canceled purchase.
Remove all samples with non-positive quantities.
Remove all samples with missing descriptions or customer IDs.
Remove all samples with the following descriptions: '85123a mixed', '21733 mixed', 'MISSING', 'found', '?', 'smashed', 'damaged', 'missing', 'broken, uneven bottom', 'mouldy', 'damages', 'wonky bottom/broken', 'broken', 'damages', 'wrong code', 'amazon', and 'faulty'.
Remove all samples with stock code of 'POST'.
Unstack The stock code feature, which is the product identifier, such that the data is converted from long format to wide multivariate time series format.
Down-sample to the daily or weekly levels, using mean aggregations for prices and summation for quantities.
Limit the dataset to only United Kingdom and the top 100 items in overall quantities, since the remaining dataset are extremely sparce and lack sufficient price variations.

A.3 Demand Estimation Benchmark Hyperparameters and Settings

In addition to using recommended parameter values by the models' developers, we explored a fair range of values using the hyperparameter optimization framework in [1]. For all the deep-learning based models, the following ranges for sampling parameter values were applied across all models:

Dropout rate : $U(0, 0.5)$ for each layer, encoder, or decoder.
Batch size : $DiscreteUniform(32, 256)$.
Learning Rate : $DiscreteUniform(0.1, 0.001)$.
Patience : $DiscreteUniform(5, 20)$ for early stopping based on validation loss or learning rate reduction after stagnation.

As for the model-specific parameter value sampling ranges were applied:

LSTNet :
Lookback period : $DiscreteUniform(7, 49)$ or $DiscreteUniform(4, 52)$ for daily or weekly frequencies respectively.
GRU units : $DiscreteUniform(50, 150)$.
GRU units for skip connection :

$DiscreteUniform(25, 75)$.
time-steps for skip connection :
$DiscreteUniform(1, 3)$.
lag time-step : $DiscreteUniform(1, 3)$.
Regularizer : $[L1, L2]$.
Regularizer : $U(0.001, 0.1)$.
LSTM-AR and LSTM-VAR :
RNN hidden layers : $DiscreteUniform(1, 10)$.
RNN cells per layer : $DiscreteUniform(10, 250)$.
Rank : $DiscreteUniform(1, 100)$, applies to multivariate normal distibution loss in LSTM-VAR only.
TFT :
Lookback period : $DiscreteUniform(7, 49)$ or
$DiscreteUniform(1, 4)$ for daily and weekly frequencies respectively, and also known as encoder length.
Hidden layers : $DiscreteUniform(8, 32)$ and
$DiscreteUniform(4, 16)$ for categorical and time-series (i.e. continuous) features respectively.
This paper :
Hidden layers : $DiscreteUniform(1, 5)$ for each feature encoder as well as the final decoder.
Latent dimension : $DiscreteUniform(1, 250)$ for each feature encoder.

For the holdout set, we use the last 1 week or 7 days period in the time series, since the highly dynamic pricing in shared economy expects frequent retraining and price adjustment.

A.4 Evolutionary Optimization Metaheurisitc Hyperparameters

We used the following settings for the optimization metaheurisitic, consistently across all methods to insure parity: population size of 100, max generations $K = 100$, crossover ratio of 70%, mutation ratio of 30%, mutation rate of 10%, maximum patience $O = 100$ (i.e. disabled), discretization step size of $\eta = 0.01$ (except for ablation or continuous optimization tests), revenue maximization as the objective function, preset potentials to $\rho_{s,i} = 1$ for all values of s and i, selection based on stochastic universal sampling, limiting number of mating parents to 2, and only the elite of a current generation is kept in the next generation (inapplicable for simple EA method). It must be noted that the stochastic universal sampling process is implemented differently in PyGAD, DEAP, and our in-house implementation. In some implementations, the population is sorted before the sampling process. Therefore, we import a custom universal sampling process without sorting in all 3 tools for the purpose of parity. It must also be noted that the described setup is specific only to the reported evaluation tests. This is independent of the production implementations at Shipt.

A.5 Software Packages

Independent of the production implementation, a Python environment with the following package versions was used for benchmark and prototyping:

- Packages for the proposed solution:
  - python 3.9.12
  - numpy 1.25.0
  - pandas 1.5.3
  - tensorflow 2.11.0 (cpu build)
  - keras 2.11.0
- Additional packages for benchmark models:
  - deap 1.4.1
  - lightgbm 3.3.4
  - lstnet-tensorflow 0.1.0
  - pygad 3.2.0
  - pytorch-forecasting 1.0.0
- Additional packages for tuning and ablation studies:
  - optuna 3.5.0
  - shap 0.44.0

A.6 Generalized Hyperparameters and Settings for the Additive Deep Learning Model

The additive deep learning framework presented in this paper is generalizable to any type of DNN architectures and their combinations. This section describes the specific architectures and parameters used for the benchmark test with the GERD dataset. Albeit performing hyperparameter tuning, we found that the proposed additive deep learning framework with simple encoder architectures performs consistently well across a wide range of parameters. We provide the following setup to establish a rule-of-thumb for selecting default parameters with

the feature dimensions of the GERD dataset as a reference. It must be noted however that the below setup is independent from the production-implemented setup. The input and output layers of each model in the additive DNN framework are summarized below

A.6.1 First pre-training encoder Θ_{R_1}.

- Embedding: size 1 -> $\frac{52+1}{2}$
- Linear: size $\frac{52+1}{2}$ -> $\frac{52+1}{2} + (\frac{100-\frac{52+1}{2}}{3})$
- ReLU
- Linear: size $\frac{52+1}{2} + (\frac{100-\frac{52+1}{2}}{3})$ -> $\frac{52+1}{2} + 2(\frac{100-\frac{52+1}{2}}{3})$
- ReLU
- Linear: size $\frac{52+1}{2} + 2(\frac{100-\frac{52+1}{2}}{3})$ -> 100
- ReLU

A.6.2 Second pre-training encoder Θ_{R_2}.

- Linear: size 1 -> $1 + \frac{100-1}{3})$
- ReLU
- Linear: size $1 + (\frac{100-1}{3})$ -> $1 + 2(\frac{100-1}{3})$
- ReLU item Linear: size $1 + 2(\frac{100-1}{3})$ -> 100
- ReLU

A.6.3 Third pre-training encoder Θ_S.

- Embedding: size 1 -> $\frac{10+1}{2}$
- Linear: size $\frac{10+1}{2}$ -> $\frac{10+1}{2} + (\frac{100-\frac{10+1}{2}}{3})$
- ReLU
- Linear: size $\frac{10+1}{2} + (\frac{100-\frac{10+1}{2}}{3})$ -> $\frac{10+1}{2} + 2(\frac{100-\frac{10+1}{2}}{3})$
- ReLU
- Linear: size $\frac{10+1}{2} + 2(\frac{100-\frac{10+1}{2}}{3})$ -> 100
- ReLU

A.6.4 Fourth pre-training encoder Θ_P.

- Multiply: $P = P^C \cdot \check{P}$, size M, M -> M
- Linear: size M -> $M + (\frac{100-M}{3})$
- ReLU
- Linear: size $M + (\frac{100-M}{3})$ -> $M + 2(\frac{100-M}{3})$
- ReLU
- Linear: size $M + (\frac{100-M}{3})$ -> 100
- ReLU

A.6.5 Fine-tuning decoder Θ'.

- Concatenate: size 100, 100, 100, 100 -> 400
- Linear: size 400 -> $400 + (\frac{M-400}{3})$
- ReLU
- Linear: size $400 + (\frac{M-400}{3})$ -> $400 + 2(\frac{M-400}{3})$
- ReLU
- Linear: size $400 + 2(\frac{M-400}{3})$ -> M

For each of the pre-training and fine-tuning steps, use a batch size of 32, 20 epochs (for a total of 100), a mean squared error loss function, and the adaptive moment estimation (Adam) optimizer.

References

[1] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*. 2623–2631.