# A Data Drift Approach to Update Deployed Energy Prediction Machine Learning Models

Hélder Teixeira[1,2 [0009−0001−9531−2976]], Arthur Matta[1,2 [0000−0002−4902−9483]], André Pilastri[1,2 [0000−0002−4380−3220]], Luís Ferreira[2 [0000−0002−4790−5128]], Pedro Pereira[1,2 [0000−0002−6169−8778]], Carlos Gonçalves[3,4,5 [0000−0001−8107−8747]], and Paulo Cortez[2,1 [0000−0002−7991−2090]]

[1] CCG/ZGDV ICT Innovation Institute, Universidade do Minho, Campus de Azurém, 4800-058 Guimarães, Portugal
helder.teixeira@ccg.pt, arthur.matta@ccg.pt, andre.pilastri@ccg.pt, pedro.pereira@ccg.pt

[2] ALGORITMI/LASI Center - Dep. Information Systems, Universidade do Minho, Campus de Azurém, 4800-058 Guimarães, Portugal
pcortez@dsi.uminho.pt, luis.ferreira@dsi.uminho.pt

[3] Efacec Energia - Máquinas e Equipamentos Eléctricos, S.A., 4466-952 S. Mamede de Infesta, Portugal

[4] ISEP, Polytechnic of Porto, Dr. António Bernardino de Almeida, 4249-015, Porto, Portugal

[5] SoftCPS - Software Technologies for Cyber-Physical Systems, Rua Dr. António Bernardino de Almeida, 4249-015 Porto, Portugal
carlos.goncalves@efacec.com

**Abstract.** While there is an increasing interest in Machine Learning (ML) based solutions, scarce research has been devoted to the deployment and monitoring of ML models. In this work, we address this research gap by proposing a new data drift ML update strategy that only considers changes in the input features. Using the realistic Growing Window (GW) and Rolling Window (RW) ML deployment simulation schemes, we propose two Drift variants (DGW and DRW), which are compared with three other ML update approaches: Single Training (ST) and Periodic retraining methods (PGW and PRW). Several computational experiments were held, using the XGBoost regression learner and 8 public-domain datasets related to energy production and consumption. Overall, when considering both the predictive performance and computational effort, the proposed DGW and DRW obtained competitive results. In particular, quality predictive errors were achieved (overall value of 1.33% for DGW and 1.43% for DRW), while requiring around half of the computational effort when compared with the periodic update versions (PGW and PRW).

**Keywords:** Data Drift · Machine Learning Deployment · Regression.

## 1 Introduction

Due to advances in Information Technology (IT), there is currently a worldwide increasing interest in the development of Artificial Intelligence (AI) and Machine Learning (ML) solutions [6]. Following this AI interest growth, there has been substantial research focused on data preparation, modeling, and evaluation stages of the ML pipeline workflow (e.g., proposal of new data processing functions, ML algorithms or evaluation measures). However, much less research has been set towards the ML deployment phase, in particular regarding supervised batch (thus offline) ML model update strategies [11]. As shown in Section 2, the few related works in this area mostly assume either a periodic model retraining or a performance monitoring trigger (retraining the model only when there is a decay in the quality of its predictions).

In this paper, we propose a novel data drift approach that only considers changes in the input features to trigger an ML retraining (update) need. When compared with the periodic update strategy, our approach requires less computation effort, since it leads to fewer ML updates. Moreover, when compared with the performance monitoring strategy, our approach can potentially trigger faster ML retraining (thus better overall predictions), since it does not require a predictive test set with target values. We particularly address the energy application domain, which is linked with a green transition context. Using eight public-domain energy production and consumption datasets and an eXtreme Gradient Boosting (XGBoost) regression learner, we perform several comparative computational experiments in order to compare five ML update strategies: a Single Training (ST) baseline, a Periodic Growing Window (PGW) or Rolling Window (PRW) retraining scheme, and our data Drift approach, under GW and RW variants (DGW or DRW). For each ML update strategy, we measure the predictive performance (considering a large number of test set batches) and the computational effort (in terms of the required number of model updates).

This paper is organized as follows. Section 2 discusses the related work. Then, Section 3 details the energy datasets, the ML base learner, the ML update strategies and evaluation measures. Next, Section 4 presents and analyzes the obtained results. Finally, Section 5 discusses the main conclusions and future work.

## 2 Related Work

The synthesis of the relevant research literature is presented in Table 1, in terms of: the Number of Datasets (#**Datasets**), the nature of the ML **Task** and several aspects regarding the **ML Update Strategy**, namely the **Training**, **Learning** and **Refreshing** modes. The last row of the table positions our approach. A varying number of datasets were adopted by the related works, ranging from one [12, 22] or two [7, 13] datasets to as many as 13 [11]. In our work, we explore 8 datasets, which corresponds to the third largest number. Regarding the ML task, most studies focus on classification goals [1, 4, 7, 9, 11, 13, 22], while we address a regression task (prediction of energy production or consumption).

Turning to the training modes, the most common procedure is the GW, also known as incremental retraining, followed by the RW scheme. Similarly, in some recent studies (e.g., [9, 11]), we adopt both GW and RW schemes in our research. As for the ML **Learning Mode**, half of the analyzed works assume an offline batch supervised learning [1, 4, 11, 22]. This type of learning is addressed in this paper since it suits the monthly time period in which the new energy data arrives. In effect, the offline batch training of the adopted XGBoost learner requires just a few minutes of execution time when using the adopted computer server (see details in Section 4). Finally, regarding the ML update trigger mechanism (**Refresh Mode**) most works either adopt performance-based [4, 12, 13, 22] or periodic [1, 7, 9, 11, 22] updates. While the former entails refreshing the model when performing below a predefined performance threshold, the latter involves model updates at specified time intervals (e.g., monthly). Noteworthy is a singular study utilizing the drift on the target variable as a refreshing trigger [12]. To our knowledge, there are no research works that measure drift on the input features only, as a determinant trigger to perform a model retraining (as executed in this work). The advantage of our approach when compared with target drift [12] or performance approaches (e.g., [13]) is that input features are often known before the target values. For instance, in our monthly datasets, the input attributes (e.g., energy sector, geographic region or even previous consumption values) are already available prior to the collection of the energy production or consumption values. Thus, the proposed input feature drift method can potentially trigger faster ML updates.

**Table 1.** Summary of the related works.

| Ref. | #Datasets | ML Update Strategy | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Task | | Training Mode | | | Learning Mode | | Refreshing Mode | | | |
| | | Cls. | Reg. | ST | GW | RW | Batch | Stream | Perf. | Time | Target | Feat. |
| [7] (2020) | 2 | ✓ | | ✓ | ✓ | | | ✓ | | ✓ | | |
| [22] (2021) | 1 | ✓ | | | ✓ | | ✓ | | ✓ | ✓ | | |
| [4] (2022) | 4 | ✓ | | | | ✓ | ✓ | | ✓ | | | |
| [12] (2022) | 1 | | ✓ | ✓ | ✓ | ✓ | | ✓ | ✓ | | ✓ | |
| [1] (2022) | 3 | ✓ | | | ✓ | | ✓ | | | ✓ | | |
| [13] (2022) | 2 | ✓ | | | ✓ | | | ✓ | ✓ | | | |
| [9] (2022) | 11 | ✓ | | | ✓ | ✓ | | ✓ | | ✓ | | |
| [11] (2024) | 13 | ✓ | ✓ | | ✓ | ✓ | ✓ | | | ✓ | | |
| This work | 8 | | ✓ | ✓ | ✓ | ✓ | ✓ | | | | | ✓ |

**Task**: **Cls.** - Classification; **Reg.** - Regression;
**Training Method**: **ST** - Single Training; **GW** - Growing Window; **RW** - Rolling Window;
**Refreshing Mode**: **Perf.** - Performance; **Feat.** - Features;

# 3 Materials and Methods

## 3.1 Energy Production and Consumption Data

This study utilizes 8 public-domain datasets from the United States of America (U.S.), extracted from the U.S. Energy Information Administration (EIA)[6]. Each dataset encompasses a monthly time-series structure, capturing data on energy production and consumption. Table 2 characterizes the energy datasets in terms of their identification (column **Dataset**), **Description**, **Data Range** (in years), number of examples (#**Instances**) and analyzed data features after applying the preprocessing procedure (#**Features**). The last column is set in terms of the number of adopted **Cat**egorical and **Num**erical features (which includes the energy target and its lagged values).

**Table 2.** Characterization of the energy datasets.

| Dataset | Description | Date Range | #Instances | #Features | |
|---|---|---|---|---|---|
| | | | | Cat. | Num. |
| 1 | Fuel consumption for electricity production by fuel type (e.g, in Short Tons, Mcf, Barrels, Billion Btu) | 2002 - 2010 | 76,586 | 3 | 15 |
| 2 | Energy consumed by economic sectors (in Trillion Btu) | 1973 - 2023 | 16,200 | 2 | 16 |
| 3 | Electricity consumption by sector (in Billion kWh) | 1974 - 2023 | 1,608 | 2 | 16 |
| 4 | Electricity production by sector (in Billion kWh) | 1974 - 2023 | 2,208 | 2 | 16 |
| 5 | Primary energy by source (in Quadrillion Btu) | 1974 - 2023 | 7,548 | 2 | 16 |
| 6 | Primary energy consumption by source (in Quadrillion Btu) | 1973 - 2023 | 6,948 | 2 | 16 |
| 7 | Renewable energy consumption by primary energy (in Trillion Btu) | 1974 - 2023 | 5,052 | 2 | 16 |
| 8 | Renewable energy production by primary energy (in Trillion Btu) | 1974 - 2023 | 2,304 | 2 | 16 |

**Features: Cat.** - Categorical; **Num.** - Numerical.

**Units:** Thousand cubic feet (Mcf), British thermal unit (Btu), kilowatt-hour (kWh).

In terms of data preprocessing, in order to improve the predictive results by capturing monthly seasonal patterns, we applied the popular sliding time window transform [8] to the output energy target ($y$), resulting in 12 lagged numeric inputs that were added to all datasets and that correspond to the energy production or consumption values of the previous 12 months (one year). After applying this procedure, there were a few examples (the oldest ones) for which there were no previous lagged values and thus were discarded from the datasets.

While lagged numeric inputs are included in the datasets, they also include several categorical and numeric features. Thus, rather than adopting a pure univariate time series forecasting approach, in this work, we assume an ML regression task. The categorical and numeric features were treated as follows. The

[6] https://www.eia.gov/totalenergy/data/annual/index.php

energy values ($y$) are used as the numeric target and also to compute the 12 lagged inputs. The temporal features (month and year) were only used for two purposes: to execute the sliding window preprocessing step; and to chronologically order the examples, allowing to simulate the GW or RW procedures. We note that the month and year were not used as inputs by the ML models, in order to avoid a memorization that would prevent a proper predictive generalization. For instance, if the year input was adopted, it would imply a strong ML extrapolation when performing predictions for the year 2024 since there are no training examples for the same year. All other data features were kept in the datasets, such as the energy type, sector or geographic region. One of the datasets (1) contained different unit values for the energy target ($y$). This was handled by keeping the original target values ($y$) while adding a categorical input that defined the unit type (e.g., Barrels, Billion British thermal unit - Btu).

### 3.2 ML Update Methods

For this study, we adopted the XGBoost base learner, which is an implementation of an ensemble of gradient-boosted decision trees designed to be computationally efficient and provide good predictive performance [3]. This ML algorithm is quite popular given that it won numerous ML competitions [17].In order to provide fair unbiased comparison and also to reduce the computational effort, in this work we adopt the default XGBoost hyperparameters for all tested ML update strategies, as implemented using the `xgboost` Python module.

In terms of the evaluation procedure, a time order holdout split was applied to all datasets. Each dataset was thus divided into an initial training set with $W$ of the oldest examples (70%) and an evaluation set with the most recent records (30%). The evaluation sets were further partitioned into monthly batches (with $T$ test examples). The total number of evaluation batches ($B$) varies from 30 to 184 (as shown in column #**Batches** of Table 4).

In this study, we compared five different strategies for updating the models, namely Single Training (ST), Periodic Growing Window (PGW), Periodic Rolling Window (PRW), Drift-based Growing Window (DGW), and Drift-based Rolling Window (DRW). The ST strategy is used as the baseline and consists of a single training of the model with the initial data (70% of the oldest instances) and never refreshing it.

The PGW and DGW strategies follow the same base Growing Window (GW) approach [15], which consists of iteratively training a model on increasing amounts of historical data. Initially, the PGW model is trained with a smaller window of past data (70% of the oldest examples). Then, it is updated (retrained) with additional data in subsequent iterations (one for each evaluation batch), gradually expanding the training window size by monthly batch increments (as shown in Fig. 1). The PGW scheme realistically simulates a supervised offline usage of an ML over time, allowing the ML model to periodically adapt to new incoming data (every month). As for the DGW approach, it is similar to the PGW strategy except that the ML model is only retrained when there is an in-

put data drift (as further explained in this subsection). Thus, the DGW method tends to produce fewer ML updates when compared with PGW.

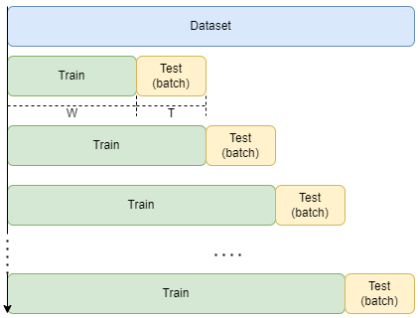



**Fig. 1.** Growing Window scheme.

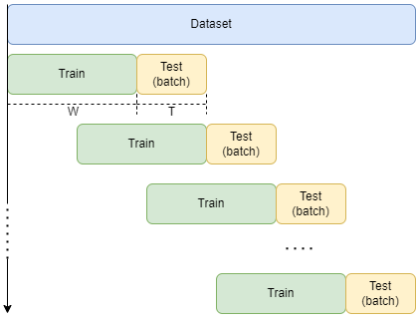



**Fig. 2.** Rolling Window scheme.

Turning to the PRW and DRW methods, they assume the same Rolling Window (RW) scheme [21], which involves creating a moving window of fixed size along a time-ordered dataset. In the first iteration, the ML algorithm is trained with the $W$ oldest elements and tested on subsequent $T$ ones. Each time a new incoming batch of data arrives (in our case, one month), the training set is updated by discarding the oldest $T$ records and adding newer $T$ ones, thus rolling the training window (as shown in Fig. 2). The PRW is more suited for higher temporal dynamic changes since it discards the oldest training records. Also, it is more computationally scalable when compared with PGW, since the training effort is the same (i.e., same training size of $W$) across the distinct RW iterations. Similarly to what was previously described, the DRW approach follows the same PRW scheme (e.g., training size of $W$) except that the ML is only retrained when an input data drift is detected.

The proposed input data drift trigger works as follows. Let $x_{i,tr}$ and $x_{i,ts}$ denote the $i$-th input feature values of the training (of size $W$) and test sets (of size $T$), respectively. The input feature drift measure ($DF_i$) is computed as: $DF_i = drift(x_{i,tr}, x_{i,ts})$, where $drift$ denotes the drift measurement function. In this work, we adopt a recently proposed tool to compute this function, namely the `Evidently` Python library (https://docs.evidentlyai.com). In a recent benchmark study [16], this tool provided competitive results when compared with other data drift detection tools (e.g., `NannyML`, `AlibiDetect`). We adopt the default `Evidently` configuration, which assumes:

**numeric data** – a two-sample Kolmogorov-Smirnov test [2] (if less than 1000 observations in the reference set, which corresponds to the training data – $x_{i,tr}$) or a Wasserstein Distance measurement [19] (otherwise);

**categorical data** – a proportion difference test for independent samples based on the Z-score (if a binary feature and less than 1000 observations in the reference set), a chi-squared test [20] (if multi-class feature with less than

1000 observations) or a Jensen–Shannon divergence measurement [14] (otherwise);

**confidence level in all tests** – 0.95.

For each $i$-th input feature, the adopted tool returns a binary value, thus $DF_i \in \{0, 1\}$. Since a predictive ML model uses several input features ($i \in \{1, 2, ..., I\}$), where $I$ denotes the total number of inputs, in this work we propose a global input drift measurement ($D$) that is based on a weighted sum when considering the ML input feature importances, namely:

$$D = \sum_{i=1}^{I} FI_i \times DF_i \tag{1}$$

where $FI_i \in [0, 1]$ denotes the input relevance of the $i$-th input feature when analyzing the last trained ML model. To compute this relevance, we adopted the `xgb_model.feature_importances_` values that are already computed by the `xgboost` Python module. For other ML algorithms, different input relevance methods could be adopted, such as the sensitivity analysis proposed in [5] and that is applicable to any supervised ML algorithm. Finally, the ML update trigger is tested for all $B$ batch iterations of the GW or RW procedure and executed (launching a model retraining) when $D > Th$, where $Th \in [0, 1]$ is a threshold (in this work we adopt the middle range value, namely $Th = 0.5$).

### 3.3 Evaluation Measures

After obtaining the ML predictions for a particular test batch (of size $T$), the Normalized Mean Absolute Error (NMAE) predictive performance measure is computed:

$$MAE = \frac{1}{T} \sum_{i=1}^{T} |y_i - \hat{y}_i| \tag{2}$$

$$NMAE = \frac{MAE}{\max(y) - \min(y)} \tag{3}$$

where $y_i$ denotes the target value for the $i$-th test example, $\hat{y}_i$ the corresponding ML prediction, $\max(y)$ and $\min(y)$ represent the minimum and maximum target values (when considering all data). The $NMAE$ measurement has two main advantages [18]. Firstly, it is a scale-independent measure, allowing the comparison of the results related to different datasets. Secondly, it is easy to interpret, since it is expressed as a percentage value (the lower the value, the better the predictions). The $NMAE$ value is computed for each test batch. The overall results (considering all $B$ batches) are aggregated by using the median measurement. Statistical significance (p-value below 0.05) is obtained by computing a paired Wilcoxon non-parametric test [10].

Regarding the computational effort, we also measure the total number of ML updates ($U$). Thus, a larger value of $U$ would imply a higher computational effort (since more ML models are trained). For ST, PGW and PRW, the number of updates is fixed and equal to: $U = 0$ (ST) and $U = B - 1$ (PGW and PRW).

## 4 Results

All experiments conducted in this paper were executed using Python code that was executed in a Linux server running Ubuntu 20.04 with 32Gb of memory and 16 cores. Table 3 presents the median NMAE values (for all $B$ test batches) achieved by each update strategy on each dataset, as well as the global values (median value over all datasets, as shown in the last row of the table). Globally, XGBoost updated using the DGW method achieved the lowest NMAE result (1.33%), followed by DRW (1.43%) and GW (1.45%) methods. On the other hand, the PRW method resulted in the worst predictive performance, with a global median NMAE of 2.91%, and the ST the second worst, with 2.90%. In particular, XGBoost using the DGW method presented the best predictive performance for datasets 1, 6 (tied with PGW) and 8, with 0.03, 0.85 and 1.81 NMAE values, respectively. As for the DRW method, it resulted in the best median NMAE values for datasets 3 (3.13%), 4 (2.08%), 5 (0.67%) and 7 (6.23%), while GW method lead XGBoost to perform better predictions on datasets 2 and 6 (tied with DGW), with, respectively, 0.56% and 0.85% median NMAE values.

**Table 3.** Comparison of NMAE (in %) results (best values in **bold**).

| Dataset | Method | | | | |
|---|---|---|---|---|---|
| | **ST** | **PGW** | **PRW** | **DGW** | **DRW** |
| 1 | 0.04 | 0.04 | 0.11 | $^{\bullet}$**0.03** | 0.04 |
| 2 | 0.63 | $^{\star}$**0.56** | 2.01 | 0.61 | 0.65 |
| 3 | 5.51 | 3.17 | 9.77 | 3.17 | $^{\bullet}$**3.13** |
| 4 | 2.68 | 2.26 | 10.4 | 2.26 | $^{\bullet}$**2.08** |
| 5 | 3.11 | 0.72 | 1.55 | 0.69 | $^{\circ}$**0.67** |
| 6 | 1.24 | $^{\bullet}$**0.85** | 2.33 | $^{\bullet}$**0.85** | 0.86 |
| 7 | 9.65 | 6.36 | 10.94 | 6.51 | $^{\bullet}$**6.23** |
| 8 | 8.79 | 2.04 | 3.49 | $^{\star}$**1.81** | 1.99 |
| **Global** | 2.90 | 1.45 | 2.91 | $^{\bullet}$**1.33** | 1.43 |

**Method**: **ST** - Single Training; **PGW** - Periodic Growing Window; **PRW** - Periodic Rolling Window; **DGW** - Drift Growing Window; **DRW** - Drift Rolling Window.

$\star$: Statistically significant compared with all.

$\circ$: Statistically significant compared with ST, PGW and PRW.

$\bullet$: Statistically significant compared with ST and PRW.

Complementary to these results, Table 4 presents the total number of batches ($B$) and model updates ($U$) for the DGW and DRW methods on each dataset. DRW and DGW present the same number of ML model updates for datasets 3, 7 and 8, and different in the others. In general, both methods have a similar

computational effort when compared against each other. However, when comparing the input-drift-based methods to PGW and PRW, there is in general a significant computational effort decrease. For example, around half of the updates are required by DGW and DRW for five of the energy datasets (2, 3, 6, 7 and 8). Overall, both the proposed methods (DGW and DRW) have shown the best balance between the predictive performance and computational cost, when compared with the remaining ones (ST, PGW and PRW).

**Table 4.** Number of model updates.

| **Dataset** | **#Batches ($B$)** | **#Updates ($U$)** | |
|---|---|---|---|
| | | **DGW** | **DRW** |
| 1 | 30 | 1 | 1 |
| 2 | 184 | 82 | 64 |
| 3 | 180 | 87 | 87 |
| 4 | 180 | 179 | 173 |
| 5 | 180 | 142 | 130 |
| 6 | 184 | 63 | 65 |
| 7 | 180 | 87 | 87 |
| 8 | 180 | 87 | 87 |

Both DGW and DRW methods obtained high quality performances (as shown in Table 3 and Fig. 3), with no clear winner. While requiring the same computational effort, DGW presented the best overall predictive performance (yet not statistically significant when compared with DRW) and 3 best dataset results. As for DRW, it obtained a quality global median NMAE (1.43%) and 4 best dataset results. Nevertheless, DRW is more scalable over time than DGW and thus it is more useful for larger ML deployment stages.

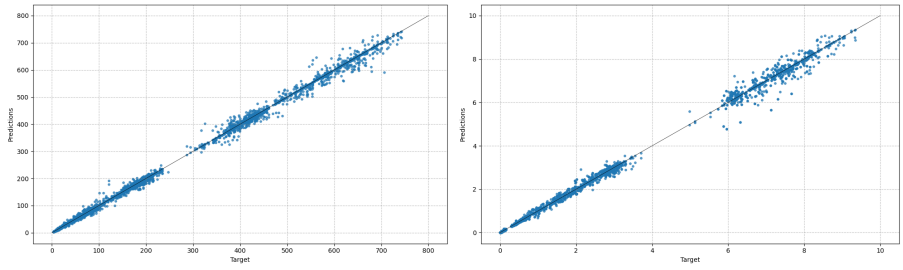



**Fig. 3.** DGW predictions (blue dots, $y$-axis) for all test batches versus the target values ($x$-axis) for datasets 3 (left) and 6 (right). The diagonal line denotes the perfect regression.

For demonstration purposes, Fig. 4 plots the NMAE ($y-$axis) evolution values for different batches ($x$-axis) and update strategies for dataset 6.

In Fig. 4, the DGW and DRW methods tend to obtain the smallest NMAE values.

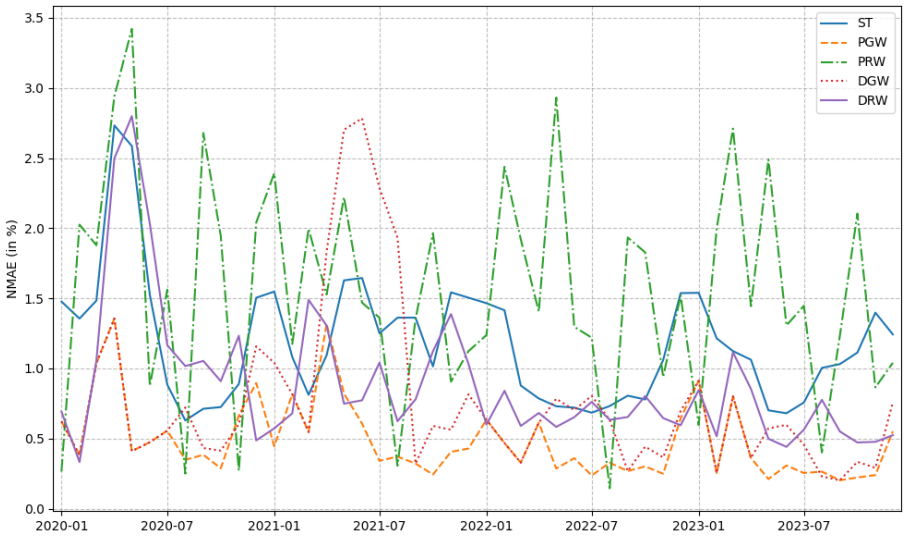



**Fig. 4.** Evolution of the NMAE values for dataset 6.

## 5 Conclusions

In this paper, we propose a new input feature data drift approach, called Drift Growing Window (DGW) and Drift Rolling Window (DRW), for updating energy prediction deployed Machine Learning (ML) models. Extensive experiments conducted on eight public datasets from the U.S. energy sector demonstrated that the proposed DGW and DRW methods achieved a competitive predictive performance (e.g., overall predictive error of 1.33% and 1.43%) when compared to other ML update strategies, such as Single Training, Periodic Growing Window, and Periodic Rolling Window. Also importantly, our drift detection approach resulted in a significant reduction in the computational effort required, with an average decrease of over 40% (DGW) and 45% (DRW) in the number of model updates needed compared to periodic strategies.

The main implications of this work are twofold. Firstly, by reducing the number of model updates, the proposed approach can improve the efficiency of deployed ML systems, especially in resource-constrained environments. Secondly, by utilizing input data drift as the update trigger, the proposed approach

can potentially trigger model updates faster than performance monitoring-based strategies, thus avoiding a prolonged degradation in the predictive performance.

In terms of future research, we intend to adopt more sophisticated methods for detecting data drift, such as adaptive thresholds or online change detection techniques. We also aim to study the applicability of our approach to other real-world domains and tasks (e.g., classification).

**Acknowledgments.** This work has been supported by the European Union under the NextGenerationEU, through a grant of the Portuguese Republic's Recovery and Resilience Plan (PRR) Partnership Agreement, within the scope of the project ATE – Aliança para a Transição Energética, aiming at enhancing the competitiveness and resilience of energy sector companies, thus propelling Portugal to a leadership position on decarbonization and promoting an effective energy transition (Project ref. nr. 56 - C644914747-00000023).

## References

1. Adam, G.A., Chang, C.K., Haibe-Kains, B., Goldenberg, A.: Error amplification when updating deployed machine learning models. In: Proceedings of the 7th Machine Learning for Healthcare Conference (MLHC). Proceedings of Machine Learning Research, vol. 182, pp. 715–740. PMLR (Aug 2022), https://proceedings.mlr.press/v182/adam22a.html
2. Berger, V.W., Zhou, Y.: Kolmogorov–smirnov test: Overview. Wiley statsref: Statistics Reference Online (Sep 2014). https://doi.org/10.1002/9781118445112.stat06558
3. Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 785–794. ACM (Aug 2016). https://doi.org/10.1145/2939672.2939785
4. Chi, S., Tian, Y., Wang, F., Zhou, T., Jin, S., Li, J.: A novel lifelong machine learning-based method to eliminate calibration drift in clinical prediction models. Artificial Intelligence in Medicine **125**, 102256 (Feb 2022). https://doi.org/10.1016/j.artmed.2022.102256
5. Cortez, P., Embrechts, M.J.: Using sensitivity analysis and visualization techniques to open black box data mining models. Information Sciences **225**, 1–17 (Mar 2013). https://doi.org/10.1016/J.INS.2012.10.039
6. Darwiche, A.: Human-level intelligence or animal-like abilities? Communications of the ACM **61**(10), 56–67 (Sep 2018). https://doi.org/10.1145/3271625
7. Doak, J.E., Smith, M.R., Ingram, J.B.: Self-updating models with error remediation. In: Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications II. vol. 11413. SPIE (May 2020). https://doi.org/10.1117/12.2563843
8. Donate, J.P., Cortez, P.: Evolutionary optimization of sparsely connected and time-lagged neural networks for time series forecasting. Applied Soft Computing **23**, 432–443 (Oct 2014). https://doi.org/10.1016/J.ASOC.2014.06.041
9. Hinder, F., Vaquet, V., Brinkrolf, J., Hammer, B.: On the change of decision boundaries and loss in learning with concept drift. arXiv (Feb 2022). https://doi.org/10.48550/arXiv.2212.01223
10. Hollander, M., Wolfe, D.A., Chicken, E.: Nonparametric Statistical Methods. Wiley Series in Probability and Statistics, John Wiley & Sons, Inc. (Jul 2015). https://doi.org/10.1002/9781119196037

11. Ilic, M., Ivanovic, M., Kurbalija, V., Valachis, A.: Towards optimal learning: Investigating the impact of different model updating strategies in federated learning. Expert Systems with Applications **249**, 123553 (Feb 2024). https://doi.org/10.1016/j.eswa.2024.123553
12. Kidane, L., Townend, P., Metsch, T., Elmroth, E.: When and how to retrain machine learning-based cloud management systems. In: International Parallel and Distributed Processing Symposium Workshops (IPDPSW). pp. 688–698. IEEE (Jun 2022). https://doi.org/10.1109/IPDPSW55747.2022.00120
13. Kim, M., Lim, B., Lee, K., Kwon, H.: Effective model update for adaptive classification of text streams in a distributed learning environment. Sensors **22**(23), 9298 (Nov 2022). https://doi.org/10.3390/s22239298
14. Menendez, M.L., Pardo, J.A., Pardo, L., Pardo, M.C.: The jensen-shannon divergence. Journal of the Franklin Institute **334**(2), 307–318 (Mar 1997). https://doi.org/10.1016/S0016-0032(96)00063-4
15. Montgomery, D.C., Jennings, C.L., Kulahci, M.: Introduction to time series analysis and forecasting. Wiley Series in Probability and Statistics, John Wiley & Sons, Inc. (Apr 2015)
16. Müller, R., Abdelaal, M., Stjelja, D.: Open-source drift detection tools in action: Insights from two use cases. arXiv (Apr 2024). https://doi.org/10.48550/arXiv.2404.18673
17. Nielsen, D.: Tree boosting with xgboost - why does xgboost win "every" machine learning competition? Master's thesis, NTNU (Dec 2016), http://hdl.handle.net/11250/2433761
18. Oliveira, N., Cortez, P., Areal, N.: The impact of microblogging data for stock market prediction: Using twitter to predict returns, volatility, trading volume and survey sentiment indices. Expert Systems with Applications **73**, 125–144 (May 2017). https://doi.org/10.1016/J.ESWA.2016.12.036
19. Panaretos, V.M., Zemel, Y.: Statistical aspects of wasserstein distances. Annual Review of Statistics and its Application **6**, 405–431 (Mar 2019). https://doi.org/10.1146/annurev-statistics-030718-104938
20. Strahler, A.N.: Quantitative slope analysis. Geological Society of America Bulletin **67**(5), 571–596 (May 1956). https://doi.org/10.1130/0016-7606(1956)67[571:QSA]2.0.CO;2
21. Tashman, L.J.: Out-of-sample tests of forecasting accuracy: an analysis and review. International Journal of Forecasting **16**(4), 437–450 (Oct 2000). https://doi.org/10.1016/S0169-2070(00)00065-0
22. Züfle, M., Erhard, F., Kounev, S.: Machine learning model update strategies for hard disk drive failure prediction. In: International Conference on Machine Learning and Applications (ICMLA). pp. 1379–1386. IEEE (Dec 2021). https://doi.org/10.1109/ICMLA52953.2021.00223