Diogo
Francisco Pedrosa

Melhorar o desempenho da previsão de consumos de água: deteção de desvios com o retreino de modelos de aprendizagem automática

Improving the performance of water demand forecasting: drift detection with machine learning model's retraining

Universidade de Aveiro
2023

Diogo
Francisco Pedrosa

Melhorar o desempenho da previsão de consumos de água: deteção de desvios com o retreino de modelos de aprendizagem automática

Improving the performance of water demand forecasting: drift detection with machine learning model's retraining

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Ciência de Dados , realizada sob a orientação científica do Doutor Sérgio Guilherme Aleixo De Matos, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Prof. Doutor Joaquim Arnaldo Carvalho Martins
Professor Catedrático da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor Sérgio Guilherme Aleixo De Matos
Professor auxiliar da Universidade de Aveiro (orientador)

Prof. Doutor Rui Pedro Sanches de Castro Lopes
Professor coordenador do Instituto Politécnico de Bragança

agradecimentos / acknowledgements

Um agradecimento ao meu orientador Sérgio Matos e ao professor Gil Campos pela disponibilidade.
Um agradecimento à equipa da SCUBIC, em especial ao Bruno pela oportunidade, ao Kevin e à Mariana, por toda a ajuda que me deram na realização deste trabalho.
Um agradecimento aos meus pais por todo o suporte que me deram no decorrer da vida.
Um agradecimento à Joana, ao Diogo, à Tatiana, à Nakov, à Beatriz e à Cláudia por todos os momentos que me proporcionaram.
Um agradecimento à Rita Oliveira pela sua genorosidade.
Um obrigado à Daniela, ao Grilo, ao João e à Sara por cá estarem a apoiar-me.
Um grande agradecimento ao Pedro por toda a ajuda que me deu ao longo da vida.

Palavras Chave

Previsão de Consumos, Aprendizagem Automática, Desvio de Conceito, Desvio nos Dados, Detetores de Desvio, Aprendizagem Incremental

Resumo

A previsão de consumos de água é um campo importante que auxilia as empresas de abastecimento de água a gerir eficientemente os seus recursos hídricos, resultando numa enconomia de custos e preservação dos recursos naturais. Este trabalho tem como objetivo encontrar uma abordagem automatizada para o retreino dos modelos de aprendizagem automática responsáveis pela previsão dos consumos de água. Essa necessidade surge da deterioração do desempenho dos modelos durante a produção, causada por desvio nos dados ou na relação entre os dados e a variável que se quer prever. Uma série de experiências foram conduzidas utilizando dados artificiais e reais, explorando três detectores de desvio: ADWIN, KSWIN e PH. Esses detectores de desvio foram testados em conjunto com o retreino dos modelos, quando os desvios são detectados, e a detecção de desvios foi também realizada tanto nos dados quanto na métrica R^2. A última opção, comumente utilizada na literatura, não teve um bom desempenho com dados de consumos de água, devido à alta variabilidade das previsões. Por outro lado, os testes mostraram que KSWIN e PH foram os detectores de mudanças mais eficazes, melhorando o desempenho em todos os dados. Além disso, uma exploração inicial de incremental learning revelou que tem potencial como uma alternativa melhor em relação ao retreino de modelos a partir do zero.

Keywords Demand Prediction, Machine Learning, Retraining, Concept Drift, Data Drift, Drift Detectors, Incremental Learning

Abstract Water demand forecasting is an important field that aids water supply companies in managing their water resources efficiently, leading to cost savings and preservation of natural resources. This work aims to find an automated approach for retraining machine learning models responsible for water demand predictions. This need arises from the deterioration of model performance during production, caused by data drifts or changes in the relationship between the data and the target variable. A series of experiments were conducted using artificial and real data, exploring three drift detectors: ADWIN, KSWIN, and PH. These drift detectors were tested in conjunction with retraining when drifts were detected, and drifts detection was performed on both raw data and the R^2 metric. The latter option, commonly used in the literature, did not perform well with water demand data due to the high prediction variability. Conversely, the tests showed that KSWIN and PH were the most effective drift detectors, improving performance across all data. Additionally, an initial exploration of incremental modeling revealed its potential as a better alternative to retraining models from scratch.

Contents

List of Figures

List of Tables

Glossary

ADWIN	Adaptive Windowing
ANN	Artificial Neural Network
ARF-Reg	Adaptative Random Forest Regressor
CNN	Convolution Neural Network
Conv1D	one-dimensional Convolution
Conv1D-GRU	one-dimensional Convolution-Gated Recurrent Unit
DDM	Drift Detection Method
DT	Decision Tree
DOF	Degree of Drift
ECDD	Concept Drift Detection
EDDM	Early Drift Detection Method
EIA	Error Intersection Approach
FEDD	Feature Extraction for Explicit Concept Drift Detection
GRU	Gated Recurrent Unit
GRUN	Gated Recurrent Unit Network
HDDMA	Hoeffding Drift Detection Method based on the moving average
HDDMW	Hoeffding Drift Detection Method based on the exponentially weighted moving averages
KS-test	Kolmogorov-Smirnov statistical test
KSWIN	Kolmogorov-Smirnov Windowing
LR	Linear Regression
LSSVM	Least Square Support Vector Machine
LSTM	Long-Short Term Memory neural networks
MAPE	Mean Absolute Percentage Error
MLP	Multi-Layer Perceptron
NSE	Nash-Sutcliffe Efficiency
PH	Page-Hincley test
R^2	coefficient of determination
RMSE	Root Mean Square Error
STEPD	Statistical Test of Equal Proportions
StWDF	Short-term Water Demand Forecasting
SVM	Support Vector Machine
S-H-ESD	Seasonal Hybrid Extreme Student Deviant
XGBoost	eXtreme Gradient Boosting

CHAPTER 1

Introduction

1.1 Context

Water is the most critical natural resource for human survival. Since the beginning of time, human settlements have been established near sources of water. Today, in developed countries, accessing potable water is as simple as turning on the tap, thanks to sophisticated water supply systems [1]. However, to maintain the sustainability, management and better operation of these systems, it is essential to accurately forecast water demand. This is where the field of water demand forecasting comes in [2].

Water demand forecasting involves predicting the future water usage based on past water demand data, as well as various other factors like population growth, economic development, and climate change [3] [4]. The forecasting can be either short-term or long-term, with different goals for each. Long-term forecasting, aims to inform water supply planning, infrastructure investment decisions, and environmental management strategies. By projecting future water demand, long-term forecasting can provide insights into the likely future demand for water in various sectors and help water management authorities plan for future infrastructure needs and allocate resources accordingly [5]. Short-term forecasting aims to ensure the optimal operation of water supply systems and to facilitate efficient resource allocation. It helps water utilities determine the amount of water that should be produced, stored, and distributed to meet customer demand, while minimizing the risk of shortages or surpluses [6]. This work focus on improving the forecast of Short-term Water Demand Forecasting (StWDF).

1.2 Objectives

As discussed in Chapter 2, there is currently a significant amount of research focused on enhancing the performance of StWDF. The objective of this dissertation is to explore methods for automating the retraining of machine learning models utilized in StWDF, thereby ensuring their continuous optimization. In addition to this primary goal, an alternative approach is

investigated, where the model is updated incrementally, on a daily bases or on batches, rather than undergoing a complete retraining process from scratch, this approach is usually called incremental learning.

This work was conducted within the environment of SCUBIC, a company that forecasts short-term water demand and utilizes optimization algorithms to manage water supply systems. This allows their clients in Brazil, Spain, and Portugal to save money by guiding them to pump water only when energy prices are low, save water by pumping only the necessary amount, and save energy by pumping water efficiently.

The company encountered the challenge of determining the appropriate timing for retraining their models developed for StWDF. Currently, their strategy involves manual tracking of the models and initiating retraining when a decline in prediction accuracy is observed. However, due to the growing number of clients, it has become impractical to manually monitor the consumption forecasting for every data point of each client. Therefore, the objective of this research was to investigate methods to determine the optimal timing for model retraining and, if feasible, propose an automated solution for the retraining process.

The decay in the model's performance occurs due to a phenomenon called drift. This is a well-studied phenomenon; however, the nomenclature for drift is not clear in the literature. Therefore, throughout this dissertation, "concept drift" is referred as the change in the relationship between the input variables and output variable, and "data drift" as the changes in the data distribution. In Chapter 3, you can find a more detailed explanation of these concepts and also the types of drifts that exist.

In the field of StWDF, drift can be caused by several reasons. One of the most obvious examples is the COVID-19 pandemic, which has changed demand patterns not only in this field but also in almost every existing field. Nowadays, there are still many reasons for drift to occur, and here is a list of possible reasons:

- Changes in the population – a change in the number of users utilizing water;
- Changes in water usage patterns – such as efforts by the population to reduce water usage;
- Changes in the water supply infrastructure, such as the addition or removal of water storage facilities or changes in water treatment processes, that can lead to changes in water demand patterns;
- Hardware problems like leakage problems;
- Changes in the sensors that measure water demand.

1.3 Guide of Lecture

This work is structured as follows:

- Chapter 1: An introduction to the field is provided, along with an explanation of why the field of StWDF is important, and a definition of the problem that this dissertation aims to solve;
- Chapter 2: A review of the most recent literature on StWDF, as well as concept drift detection and retraining modeling, is presented;

- Chapter 3: The methodology of this work is described, and the choices made are explained;
- Chapter 4: The results and discussion of the experiments conducted with artificial and real-world datasets are presented;
- Chapter 5: The conclusions of the work and future directions for research are presented.

CHAPTER 2

State of the Art

This chapter provides an overview of recent studies conducted on water demand forecasting. Although not the main focus of this thesis, it is essential to address the existing literature to provide contextual background for the field.

2.1 Water Demand Forecasting

Recent studies in the literature have concentrated on enhancing the performance of StWDF. Chen et al. [7] identified a gap in the field where automated feature extraction methods were lacking. This limitation restricts the data mining and self-adaptability capabilities of the models, rendering them incomplete. To address this issue, they proposed a new framework for StWDF. Their framework utilizes a Seasonal Hybrid Extreme Student Deviant (S-H-ESD) method for data preprocessing and a one-dimensional Convolution-Gated Recurrent Unit (Conv1D-GRU) forecasting model. This approach ensures efficient data utilization and assesses the impact of different training dataset lengths on forecasting. Automatic feature extraction is performed and subsequently utilized in the Gated Recurrent Unit (GRU) model. Comparative analyses with benchmark models demonstrated the superiority of S-H-ESD over the Z-score method, and the Conv1D-GRU model achieved a better Nash-Sutcliffe Efficiency (NSE) than other two benchmarks: Gated Recurrent Unit Network (GRUN)) model and the Artificial Neural Network (ANN) model. These experiments were conducted for one-step forecasting.

Another study aimed at improving the forecast accuracy of StWDF was conducted by Pu et al. [8]. They proposed a hybrid approach incorporating wavelets, Convolution Neural Network (CNN), and Long-Short Term Memory neural networks (LSTM) to enhance data predictability. This hybrid approach initially extracts key temporal features and subsequently employs LSTM to encode the temporal dependencies present in the time series. The results obtained using this approach were compared to benchmark models such as ANN, one-dimensional Convolution (Conv1D), GRU, and LSTM, which are commonly employed in recent studies

that they had analysed. The proposed method was able outperform all of these benchmark models.

Salloom et al. [9] present a method aimed at mitigating errors at extreme points, where water demand significantly deviates from the adjacent periods' average demand. They propose a new, lower complexity model for StWDF compared to existing methods in the literature. The model is based on deep learning and consists in two blocks: a dense block that captures the relationship between water demand values and classes, and a GRU block that focuses on the sequential relationship within the water demand data. The proposed method demonstrates favorable results when compared to other state-of-the-art methods, exhibiting improved prediction accuracy while reducing model complexity by a factor of six.

Wu et al. [10] strive to enhance StWDF by introducing an error correction module to the initial forecasting model, which is based on the Least Square Support Vector Machine (LSSVM). The error correction module utilizes chaotic time series derived from the difference between initial forecast values and observed values. The hybrid model, incorporating the error correction module, is compared with two other methods: a simple forecasting model and a hybrid model employing Fourier series for error correction. The proposed hybrid model and the Fourier model outperform other models in StWDF. However, during periods of frequent and disordered peak fluctuations in the error time series, the performance of these models is not satisfactory, indicating the need for further improvements.

Pacchin et al. [11] conducted a study comparing the performance of various StWDF models, rather than focusing on improving existing ones. The study considered factors such as the forecasting technique, determinism versus probability, and the data required for calibration. The models examined include an ANN-based model, a model replicating periodic water demand patterns (both requiring calibration), and two models based on moving windows of previously observed data. Additionally, a probabilistic model employing markov chains and six naive benchmark models were tested across seven real-life cases. The study concluded that both data-driven and pattern-based techniques can achieve similar forecasting accuracy. Models requiring calibration exhibited a decrease in accuracy from the calibration year to the validation year, while models based on moving windows demonstrated higher and more consistent accuracy.

Rosen et al. [2] conducted a comprehensive review of StWDF methods used in the literature from 2010 to 2021. Analyzing over 100 articles, they provided guidelines for future researchers in the field. The choice of method, according to the authors, depends on the study's objectives, considering temporal scope, analysis objectives, and available technology as crucial factors. Among the commonly used open-source software options, Julia, R, and Python, they found no significant advantages in selecting one over the others and recommended choosing based on personal preference. The review emphasized the superior performance of hybrid models compared to individual methodologies and highlighted the importance of incorporating parameter tuning phases with meta-heuristic algorithms for enhanced accuracy. Overall, Rosen et al.'s study serves as a valuable reference for researchers in StWDF, assisting in method selection based on specific research goals and available technology.

2.2 Retraining Modeling and Drift Detection

Upon conducting research, it has become evident that comprehending the concept of drift in machine learning is crucial for determining when to retrain models.

In a survey on concept drift and data drift conducted by João Gama et al. [12], three requirements for predictive models were outlined: the ability to detect and adapt to drift, the capability to distinguish noise from real changes, and operating instantly while utilizing a fixed amount of memory in storage. To achieve adaptive learning, they proposed four modules: memory, change detection, learning, and loss estimation.

The first module involves managing data by either defining a window size where older data is continuously discarded, and with the arrival of new data, a new model is built, or utilizing a gradual forgetting method that assigns weights to the data.

In the second module, various types of detection methods were described, including sequential analysis, drift detectors based on statistical process control, drift detectors based on the distributions of two different time windows, and a contextual approach.

The third module deals with two types of learning modes: retraining of the model or incremental updates of the model. Decisions can be made blindly, without any criteria, or with the knowledge of a concept drift algorithm. The authors also discussed two model management options: using a single model or an ensemble of models.

In the last module, they defined two types of models: dependent and independent. They suggested having a measurement for the computational cost of the mining process and a statistic for class taking into account class imbalance, such as using kappa-statistic. They also introduced criteria for evaluating change detection methods, including the probability of true changes, probability of false alarms, and delay of detection.

Overall, the survey by João Gama et al. provides a comprehensive overview of the different approaches and methods used for detecting and adapting to drift in machine learning, which is important for ensuring the continued accuracy and effectiveness of predictive models.

Firas Bayram et al. [13] provided an overview of performance-aware drift detectors. They highlighted the issue of different terms in the literature that define the same type of drift. The authors summarized the terms used in the literature to describe similar types of drift. Additionally, they presented numerous drift detection methods proposed up until the date of their research. Finally, they discussed how the performance of models is validated and utilized to track and detect concept drift. Their findings indicate that existing works related to fields other than classification problems are limited, and further clarification is required to determine the most suitable drift detector for each specific problem.

In another study, Paulo Gonçalves et al. [14] conducted experiments on twelve datasets to compare the performance of eight drift detectors: Drift Detection Method (DDM), Early Drift Detection Method (EDDM), Statistical Test of Equal Proportions (STEPD), Degree of Drift (DOF), Adaptive Windowing (ADWIN), paired-learners, Page-Hincley test (PH) and

Concept Drift Detection (ECDD). The study concluded that the overall best method in terms of metrics average rank and accuracy was the DDM method. However, in real-world datasets, PH achieved high accuracy, and the paired-learners method had the lowest average rank.

Lidia Kidane et al. [15] conducted a pilot study on cloud management systems to investigate the optimal timing and methodology for retraining machine learning models, using LSTM. Three methods were evaluated, including two well-known drift detection methods: ADWIN and PH, which were applied to the data to detect drift and trigger retraining. The third method was based on prediction error and involved setting a threshold, with retraining triggered when this threshold was exceeded. Additionally, the authors analyzed the amount of data necessary for retraining the models. The study found that regardless of which detection method was used, retraining always improved performance. The ADWIN method was found to require more data to detect changes than the PH. The prediction error method was found to be more sensitive to outliers and noise. In terms of the amount of data needed for the retraining phase, the study concluded that the data used should consist of between 70% and 100% new data.

Baier et al. [16] proposed a novel approach for handling concept drift in regression problems. Although their method does not specifically address the topic of retraining models, it still provides an interesting contribution to the field. The approach is applied to a real-world dataset of taxi demand and is based on paired learners. The authors build two prediction models: a simple forecast model and a complex neural network model. They use an approach called Error Intersection Approach (EIA) that switches between the two models triggered by an intersection of their error curves. This approach is particularly effective because the complex model cannot provide accurate predictions when the demand pattern deviates from its usual trajectory, while the simple model adapts better to such changes. The authors compare the results of their approach with other drift detection methods, such as PH, ADWIN, and EDDM. When drift is detected, the switch between the two prediction models is performed. The results demonstrate that their method, EIA, outperforms the well-known drift detectors. However, the method has some limitations, such as its effectiveness only when sudden drift occurs.

Cavalcante et al. [17] proposed a method for detecting drift in time series by monitoring time series features, which differs from the more common approach of detecting drift based on the error. Their method, called Feature Extraction for Explicit Concept Drift Detection (FEDD), consists of two main modules. The first module extracts the features of the time series, while the second module monitors the evolution of the time series features and tests for the occurrence of drifts. The drift detector implemented in FEDD is ECDD. The authors focused on analyzing the accuracy of their drift detection method and compared it to other methods such as ECDD, DDM and PH, which were applied to the error and triggered retraining when drift was detected. As they used an artificial dataset, it was possible to

determine the accuracy of the drift detection, however this may me a limitation since no real world dataset was tested. The authors demonstrated that their approach showed better drift detection accuracy compared to the commonly used approach of detecting drift based on the error.

Tomas Cabello-Lopez et al. [18] applied drift detection to improve time series forecasting of wind energy generation. The study utilized three drift detectors: ADWIN, PH, and Kolmogorov-Smirnov Windowing (KSWIN). The authors conducted a grid search to determine the optimal parameters for each drift detector. Whenever drift was detected, the authors retrained a Linear Regression (LR) model. The results of the study revealed that the performance improved by 7.88% to 33.97%, depending on the drift detector applied, compared to a no-retraining approach.

Lucas Baier et al. [19] proposed a novel approach to handling concept drift in prediction problems in business process mining. They detected concept drift using ADWIN and PH and addressed the issue of which data to use in the retraining phase after detecting drift. They developed three approaches: the first approach involved collecting the next batch of instances after detecting drift, and the retraining is performed on that batch. The second approach used the data before and after the drift detection, while the third approach used only past data from when the drift detection was made. The authors also tested an incremental learning approach, which is a rare feature implemented by only a few machine learning algorithms, with Naive Bayes being one of them, and the selected for the experiments. They also examined the effect of four batch sizes (500, 1000, 2000, 5000) on performance. The combination of incremental learning with retraining when drift was detected provided the best results. Furthermore, the approach that retrained the model with past data after detecting the drift showed better performance than the other two approaches. The batch size that achieved the best results was 500.

Felipe Neri [20] conducted a study in the field of financial time series to examine how drift detectors behave in this area. He clarified that although drift detectors are commonly applied to measure the accuracy of the model, they can also be applied to the raw data. Neri used the most common drift detectors in the literature, such as EDDM, ADWIN, DDM, KSWIN, PH, Hoeffding Drift Detection Method based on the moving average (HDDMA), and Hoeffding Drift Detection Method based on the exponentially weighted moving averages (HDDMW), as well as three novel drift detectors specifically tailored to the field of financial time series: myTanDD, MINPS, and mySD. The workings of these detectors can be found on page four of his article. He studied the interactions between the learners, the drift detectors, and their input. Although the study has many details to analyze, only a few are mentioned. Neri presented results that explained whether to input the raw data or the accuracy into the drift detectors. The results showed that it depends on the combination of the dataset and the drift detector. Some detectors, like KSWIN, performed well on both the data and the Mean Absolute Percentage Error (MAPE) measure of accuracy, while others, like

ADWIN, performed well only with the MAPE. Others, like PH, performed well only on the data. All of the detectors worked well with any of the learners used. The main conclusion of the study was that drift detectors should be analyzed together with the learning systems.

Overall, the studies mentioned in this part provide valuable insights into the field of retraining modeling and drift detection in machine learning. They explore different approaches, methods, and detectors for detecting and adapting to drift, as well as the impact of retraining on model performance. These studies contribute to the understanding of how drift affects predictive models and provide guidance for ensuring the continued accuracy and effectiveness of such models.

2.3 Incremental Learning

An alternative to retrain the models is implementing an algorithm that allows incremental learning, where the model updates its weights according to the new data that is received. Bellow its an overview study about incremental learning.

Lina Alberg et al. [21] presented an overview of regression tree methods for mining data streams. The three main methods discussed in their work include batch algorithms, incremental algorithms, and batch-incremental algorithms.

Batch algorithms involve training the model on the available data, and if any modifications are required, the model needs to be retrained from scratch.

Incremental algorithms, on the other hand, allow for the construction and updating of the existing model as new instances become available, ensuring the model stays up to date.

The batch-incremental approach utilizes a window of new instances, which can be either fixed or flexible. This window is used to update the model, incorporating the latest information. In their review, they focused on identifying the distinct features of different models and their ability to adapt to a time-changing environment. They analyzed five methods from the literature and summarized their findings in a table, which assessed the models based on various criteria.

These criteria included whether the models were capable of operating online, scaling linearly with the arrival rate of new examples, being domain-specific, competitive with state-of-the-art batch methods, responsive to distribution changes, and other relevant factors.

Comparing these incremental tree methods to existing batch models, they found that the incremental approaches were able to compete with traditional batch learning methods. However, there is still room for improvement in the existing incremental models, as most of them failed to meet five out of the ten criteria analyzed by Alberg et al.

This highlights the potential for further advancements in incremental learning algorithms, particularly in addressing the limitations identified in the existing models. Improving these models to meet the criteria identified by Alberg et al. would enhance their competitiveness and effectiveness in handling time-changing environments and data streams.

2.4 Incremental vs Retraining modeling

Tao Chen [22], made an empirical comparison on the previous ways of modeling that I referred on this dissertation. On his review of the literature he noticed that the papers do not provide any evidences on why they choose one method instead of the other. He referred that the general belief is that the incremental modeling is chosen when a faster training is needed and the retrained modeling is chosen when an high accuracy is needed. However he proved that this general belief is inaccurate.

To made the experience, he used eight machine learning algorithms: LR, Decision Tree (DT), Support Vector Machine (SVM), Multi-Layer Perceptron (MLP), Bagging - LR, Bagging - DT, Boosting - LR and Boosting - DT.

He collected the data from running adaptable software and tested the data of for each algorithm in a retrained version and an incremental version. For the retrained version he was extreme and retrained the algorithms each time a new sample arrived.

He find out that the accuracy is more affected by which is the learning algorithm than if it's implemented in an retrained manner or an incremental manner. For example, for the incremental modeling is better with MLP and the retrained model is better with SVM. For the ensembles algorithms the incremental modeling had a better accuracy on bagging and the retrained on boosting.

About the time that each algorithm spent, although the incremental modeling spent statically less time, for the majority of the learning algorithms the difference is negligible.

Tao Chen, state that the choice of modeling should be a trade off between training time and accuracy.

CHAPTER 3

Methodology

This chapter presents the methodology applied in the company and utilized for this work. Initially, an overview of the general methodology used by SCUBIC is provided, followed by a description of the specific methodology employed for this study.

3.1 Methodology at SCUBIC

The general methodology employed by SCUBIC can be summarized as follows:

- Data preparation for training involves applying a transformation to the newly arrived data, creating half-hour time-steps. This is achieved by calculating the mean of the received values within each half-hour interval.
- Values outside the range between the mean and a margin of 3 times the standard deviation are considered outliers.
- Outliers are replaced with the last non-outlier value from the corresponding hour on the previous day.
- Negative values and repeated values resulting from system errors are removed.
- After data preprocessing, a parallel approach is employed using multiple models trained on the last 100 days of available data.
- The model that exhibits the best performance in the past 14 days is selected for predicting the next 24 hours.
- The input features used for training the models include historical data from the past 14 days, the mean and median values from that period, and a feature indicating whether the day is a weekday or weekend.

Due to computational limitations, this study focus solely on utilizing the eXtreme Gradient Boosting (XGBoost) model, despite the company utilizing various other models. XGBoost was chosen based on its demonstrated superior performance within the company. It is important to note, however, that other models may exhibit different performance levels when subjected to the retraining conducted in this study.

XGBoost, introduced by Tianqi Chen and Carlos Guestrin [23], has emerged as one of the most popular algorithms in recent years. Before delving into explanations about its functioning, it is important to grasp the concept of boosting.

Boosting is a type of ensemble method, which utilizes a collection of models to obtain a final prediction [24]. In the case of boosting, the objective is to leverage a set of weak learners to create a stronger learner. This is accomplished by sequentially training the weak learners while considering the error produced by the previously trained model [24].

In XGBoost, the weak learners are decision trees. The training process continues to create a sequence of models until the error falls below a predefined threshold value or a specified number of models have been trained. At the end of the training phase, a weighted average of all the models is computed, assigning greater weight to models that yielded smaller errors.

XGBoost offers numerous hyperparameters that can be optimized, although the focus is on adjusting four of them:

- Learning rate: This parameter governs the algorithm's learning speed. As the value approaches zero, the algorithm becomes more precise but also more computationally expensive. The value should be in the interval $]0, 1]$.
- Max depth: This parameter determines the maximum depth of the tree. Increasing this value makes the model more complex and prone to overfitting. Conversely, reducing this value leads to underfitting. The values can be in the interval $]0, \infty]$.
- Min child weight: This parameter sets the minimum sum of instance weight required in a child. A low value results in a more complex model but also increases the risk of overfitting. The values can be in the interval $]0, \infty]$.
- Number of estimators: This parameter specifies the number of trees utilized. The values can be in the interval $]0, \infty]$.

The XGBoost implementation used in this work can be accessed here [25]. Ideally, experiments would involve training XGBoost with a tuning technique such as grid search, which aims to compute optimal values for the hyperparameters. However, due to computational constraints, this technique was only applied once for each dataset in this study.

The values that were considered the best for each dataset were then used in the retraining phase. The values that were taken into account in the first phase were:

- Learning rate: $[0.005, 0.01, 0.02, 0.05, 0.1]$
- Max depth: $[2, 3, 5, 7, 10]$
- Min child weight: $[250, 500, 1000]$
- Number of estimators: $[1, 3, 7, 10]$

3.2 Types of Drifts

With the structure of the learning model done, it's important to understand the types of drifts that exist. As previously mentioned, there are two types of drifts: concept drift and data drift. The definitions of both have already been provided in Chapter 1. In general, drifts can be defined mathematically as follows: for time points t_0 and t_1,

$$\exists X : p_{t_0}(X, y) \neq p_{t_1}(X, y), \tag{3.1}$$

where p_{t_0} is the joint distribution between the model features X and the target variable y. More specifically, concept drift can be defined as

$$\exists X : p_{t_0}(y|X) \neq p_{t_1}(y|X), \tag{3.2}$$

where t_0 in this case can represent a set of time points and p_{t_0} is the conditional probability between y and X. In this case, changes can occur with or without changes in $p(X)$. Here, $p(X)$ represents the data distribution. In practice, this type of drift is detected by tracking some measure of accuracy.

On the other hand, data drift can be defined as follows: for a set of t_0 points and t_1 points that, in a time series problem, represent a sequence of t_0:

$$\exists X : p_{t_0}(X) \neq p_{t_1}(X). \tag{3.3}$$

This type of drift is tracked in the data. However, it is often not tracked because a change in the data distribution does not necessarily imply that the concept drift has occurred. Furthermore, most of the vast literature on detecting drift is based on classification problems where the evaluation metrics are binary, making it easier to track changes in concept drift.

In this work, most of the experiments were conducted on data drift because the accuracy metrics, independent of each other, that resulted from the real-world datasets made it almost impractical to track concept drift.

Regardless of which type of drift we are tracking, it is important to understand the different kinds of drift that can occur in the data. The explanations of each one were based on the article by João Gama et al. [12].

The first type of drift is sudden drift, which occurs when the existing concept changes abruptly. In StWDF, this can happen when an existing water demand sensor is replaced, and the new one has a different calibration.

The second type of drift is incremental drift, in which the concept slowly changes. For example, a water demand sensor that slowly wears off and becomes less accurate.

The third type is recurring concept drift, which is like two sudden drifts that occur due to seasonal changes or other time-dependent factors.

The last type is gradual drift, which occurs when the concept changes gradually. For example, in winter months, when temperatures are cooler and outdoor activities are limited, water demand may be lower. However, as the climate changes and winters become milder, there may be a gradual increase in water usage during these months as people spend more time outdoors.

Visual aids can help to understand these four types of drift more easily. Below, its presented an image of each type of drift.

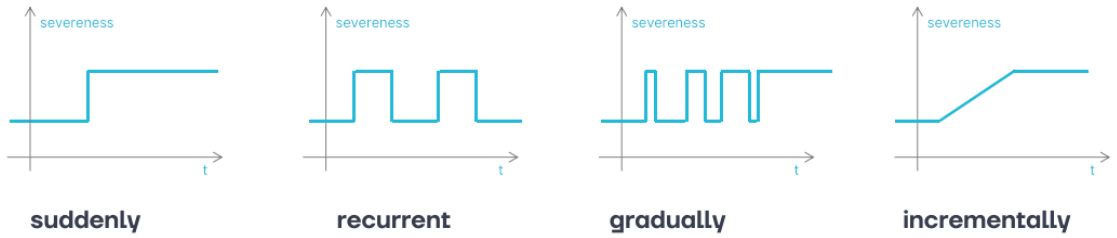

Figure 3.1: Four types of drifts that happens in concept drift and data drift. Image from [26]

3.3 Drift Detectors

To determine which type of drift detectors to use, an analysis of the existing methods in the literature was conducted. Since there is no clear consensus in the literature on which methods are superior, three commonly used methods were chosen: ADWIN, PH, and KSWIN. These methods were selected because they are widely used in the literature and have open-source implementations in scikit-multiflow [27].

3.3.1 Adaptive Window

The Adaptive Window (ADWIN) is one of the most commonly used methods in the literature and was first introduced by Albert Bifet and Ricard Gavaldà in 2007 [28]. This method can be used on both the data and error-rate. The algorithm utilizes sliding windows, whose size is recomputed every time it detects a drift. The algorithm cuts the window at different points and analyzes the average of some statistic over the two windows that resulted from the cut. If the absolute value of the difference between the averages of the two windows surpasses a predefined threshold, a change is detected at that point and all data before that time is discarded. Note that this algorithm cuts the sliding window multiple times according to a predefined criteria each time new points are received.

The test used to compare the averages of the two sub windows W_0 and W_1 can vary and must satisfy some criteria that is described in [29] as:

- If W_0 and W_1 were generated from the same distribution (no change), then with probability at least $1 - \delta$, the test says
  no change, where δ is the confidence value that $\in (0, 1)$.
- If W_0 and W_1 were generated from two different distributions whose average differs by more than some quantity $\in (W_0, W_1, \delta)$, then with probability at least $1 - \delta$, the test says "change".

The test used in the implementation that was used was the Hoeffding's independence test. In [30] its a detailed explanation of this test.

3.3.2 Page-Hincley Test

PH [31] is a variant of the cumulative sum test (CUSUM). The main objective of the PH is to detect whether the mean of an input value is deviating from the mean of previous values [29]. The way this test works is as follows: firstly, from a sequence of observations x_t, we

define

$$z_t = (x_t - \mu)/\sigma \tag{3.4}$$

where μ is the expected value of x_t and σ is the standard deviation in normal conditions. If these are not known a priori, they are estimated from the sequence. Then, the following is calculated:

$$g_0 = 0, \tag{3.5}$$

$$g_t = (g_{t-1} + z_t - k), \tag{3.6}$$

$$G_t = \min(g_t, G_{t-1}) \tag{3.7}$$

where k represents a threshold, and it is advised to set it to half the value of the changes to be detected (measured in standard deviations) [32]. Then, the following test is performed:

$$g_t - G_t > h \tag{3.8}$$

where h is the parameter that the user usually defines, but it is also advised to use $h = \ln(1/\delta)$, where δ is the false alarm rate. If the test is true, a drift is declared, and the process starts from the beginning.

The problem with this drift detector is that it is implemented only for one side. However, a change in the code of the scikit-multiflow library was made to enable the test to be performed on both sides.

3.3.3 Kolmogorov-Smirnov Windowing

KSWIN [33] is a test based on the KS-test. The KS-test is a non-parametric test used in statistics to compare two distributions and determine if they come from the same underlying distribution [34]. For two samples, an empirical distribution function is calculated for each sample, where each sample consists of n independent and identically distributed (i.i.d) observations X_t. The empirical distribution function is defined as:

$$F_n(x) = \frac{1}{n} \sum_{t=1}^{n} 1_{(-\infty,x]}(X_t) \tag{3.9}$$

where p is the number of elements in the sample. After the empirical distribution function is calculated for both samples, the Kolmogorov-Smirnov statistic is calculated:

$$D_{n,m} = \sup |F_{1,n}(x) - F_{2,m}(x)| \tag{3.10}$$

where $F_{1,n}$ and $F_{2,m}$ are the empirical distributions of the two samples, and sup is the supremum function. The null hypothesis is that the two samples are drawn from the same distribution. For a given significance level α, the null hypothesis is rejected if:

$$D_{n,m} > c(\alpha)\sqrt{\frac{n+m}{n \cdot m}} \tag{3.11}$$

where $c(\alpha) = \sqrt{-\ln(\frac{\alpha}{2})/2}$.

In KSWIN, a sliding window of fixed size is maintained, and the two samples are defined by the last m observations and the remaining $n - m$ observations in the window. The value of m is the "statistic size", which represents the number of most recent observations that the KSWIN test considers.

Figure 3.2 shows a visual example of the KS-test.

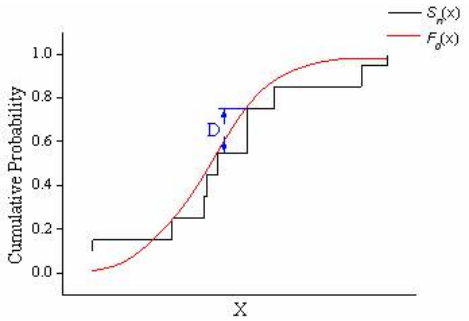

Figure 3.2: Example of how the KS-test works. Image from [35]

KSWIN uses this test, and the way he defines the two samples are by maintaining a sliding windowing of fixed size. Then a stat size is defined, and this represents the last samples that had arrived in the window. Then the other window is the remaining values of the fixed size window.

3.3.4 Approach with Drift Detectors

The drift detectors are applied in different ways. There wasn't much literature regarding the seasonal component that time series may have, which could influence false alarms by the drift detectors. Especially in the field of StWDF, this may be an important consideration, as demands during the weekend or holidays are often different from those during the week.

Therefore, the drift detectors are applied in four different ways:

- Traditional approach (see Figure 3.3): each type of drift detector tracks day by day if a drift has occurred.
- Weekly approach (see Figure 3.4): the drift detectors accumulate seven days and only check if a drift has occurred when all seven days of the week have been received.
- Weekend day approach (see Figure 3.5): two drift detectors (one for weekend days and one for other days) are used in this approach.
- Error-rate approach (see Figure 3.6): drift awareness is determined by tracking the R^2.

For each of these approaches, an adaptation of the parameters of the drift detectors was made. The precision of the drift detection parameters was determined through trial and error using artificial data.

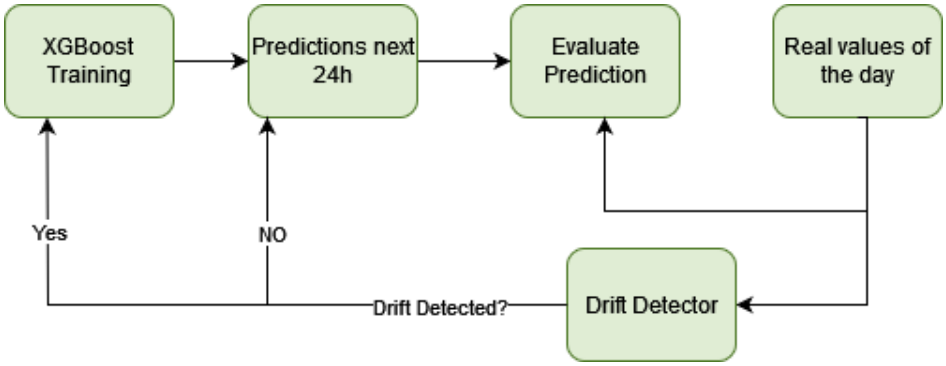

Figure 3.3: Scheme of the traditional approach experiences

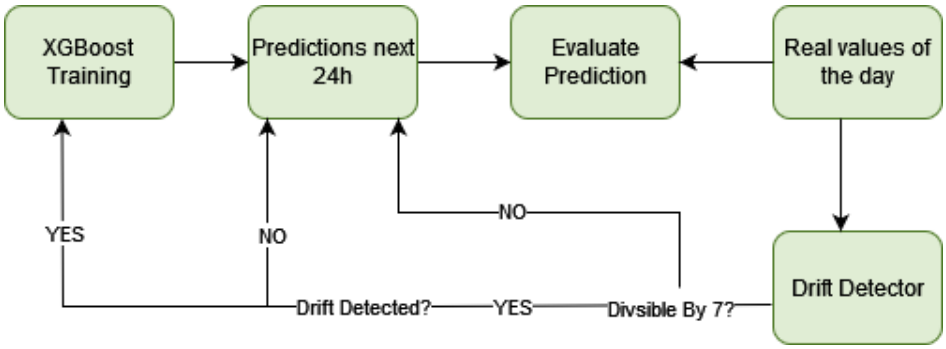

Figure 3.4: Scheme of the week approach experiences

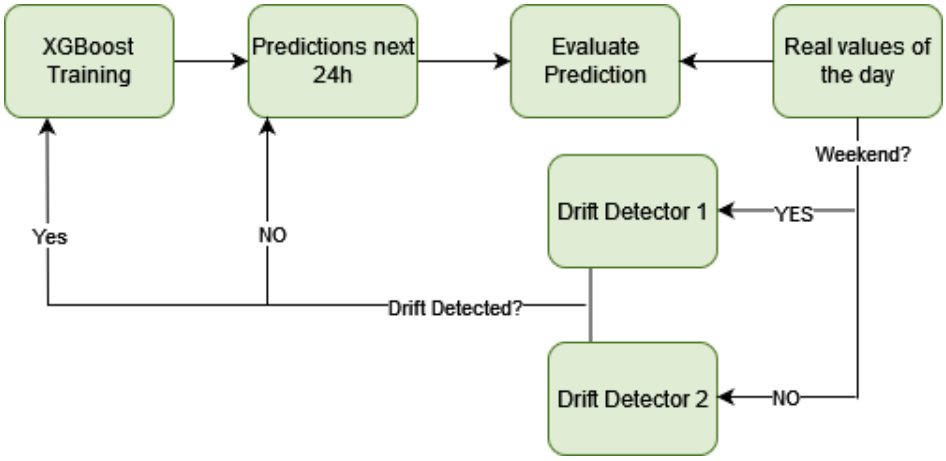

Figure 3.5: Scheme of the weekend day approach experiences

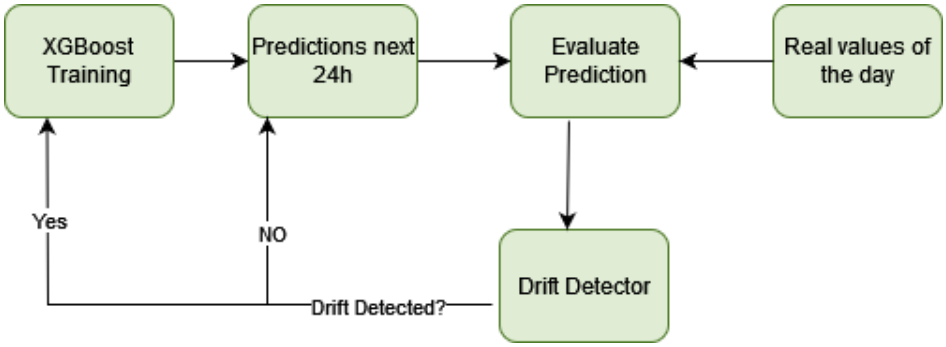

Figure 3.6: Scheme of the error-rate approach experiences

3.3.5 Retraining phase

Every time a drift is detected, the XGBoost model is retrained. However, this approach includes data that represents the previous concept, meaning that the model incorporates training data that does not account for the drift. In order to tackle this issue, two delayed retraining approaches were tested, in which the model is retrained only after 7 or 14 days following the detection of the drift.

A small study was conducted to ascertain the optimal amount of data for retraining purposes. The study examined the effectiveness of retraining with 30, 50, and 100 days of data. While many approaches in the literature suggest retraining with all available data, this can be cost-prohibitive in a production setting. Moreover, retraining with a smaller dataset may enhance the model's ability to adapt to new and evolving concepts. Consequently, this study also investigates the impact of varying the amount of data utilized for retraining.

3.4 Incremental Learning Approach

In the final part of each section, an analysis of an algorithm enabling incremental learning is conducted. However, it should be noted that XGBoost currently does not support this feature. Presently, XGBoost permits model updates with new data but necessitates the availability of the old data for updating purposes. Despite the faster retraining process in this scenario, numerous tests have indicated a significant decrease in performance compared to all other experiments. Therefore, an alternative model was chosen to explore incremental learning.

The alternative model utilized in this study is also tree-based and specifically designed for incremental training. It is important to acknowledge that direct comparisons at a 100% level cannot be made between the findings of each approach. Different machine learning models may respond divergently to incremental learning and retraining. This initial experiment lays the groundwork for future investigations and aims to explore the potential benefits that incremental learning may offer in this specific domain.

The model choosen is the Adaptative Random Forest Regressor (ARF-Reg), that was proposed by Heitor Gomes et al. [36] and is an adaptation of the adaptative random forest for classification tasks that was also proposed by those authors [37].

This algorithm is also based on decision trees, but instead of boosting it is a bagging algorithm, i.e., generates decision trees and calculates their predictions in parallel. It was designed for regression of evolving data stream, so it can update its model by adding new trees to the ensemble or modifying the existing trees based on the new information. Since the model is updating incrementally it may adapt better to the concept drift.

This algorithm has multiple important parameters, one of them is the option of using or not the drift detector ADWIN. Throughout the experiments of this work, conclusions are made to discard the use of this parameter. So the model is updated without the information that a drift have happened.

The algorithm offers a range of parameters that can be optimized, but for this study, default parameters were chosen. The only parameter that was changed was the drift detector, which was set to "none" instead of ADWIN.

Originally, the base learner of the algorithm was the Fast and Incremental Model Trees with Drift Detection algorithm. However, for this dissertation, a different base learner called the Hoeffding Tree Regressor, obtained from the skmultiflow library [27], was utilized. This base learner continuously updates and refines the tree structure as new data arrives.

As said before, this part of the dissertation its just an initial approach with a incremental learning algorithm, so a further study is needed to take more reasoned conclusions.

Two experiments were conducted using this algorithm: one where the model is updated daily and another where it is updated on a weekly batch.

3.5 Data Sets

Regarding the data sets, this work have two phases. The first phase involves creating multiple artificial data sets to which various methods for detecting drift are applied to evaluate the performance of the drift detectors. In the second phase, the best drift detection methods are applied to real-world data sets to simulate a real-world approach.

3.5.1 Artificial data sets

The artificial data sets were created to test the drift detectors for each type of drift present in Figure 3.1. In addition, a set of experiments was conducted to analyze the effect of retraining on this kind of data.

To simulate the seasonality of one day of StWDF, a day from a real-world data set was extracted and multiplied by 173 days. As each day has 48 points, there were 8,304 points in total. This baseline was used to create each type of drift.

Two types of sudden drift were created, one with a sudden drift upwards and one with a sudden drift downwards, to test if the adaptation made in the PH test had been successfully implemented.

After creating the five simulations, artificial noise was added to the data using the NumPy function to generate random numbers from a normal (Gaussian) distribution. In the Figures 3.7, 3.8,3.9, 3.10 and 3.11 you can see the generated data sets.

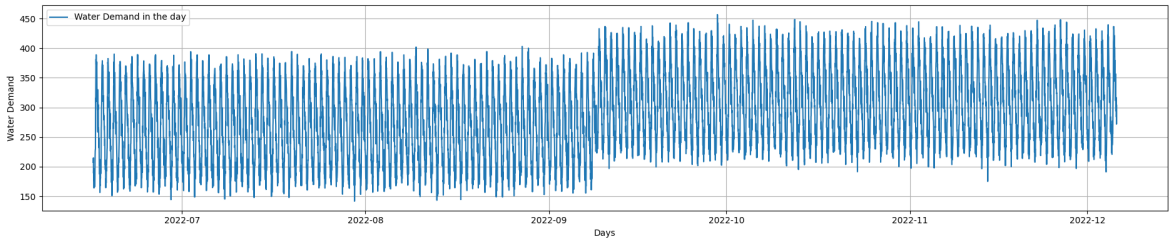

Figure 3.7: Artificial data with a sudden increased drift

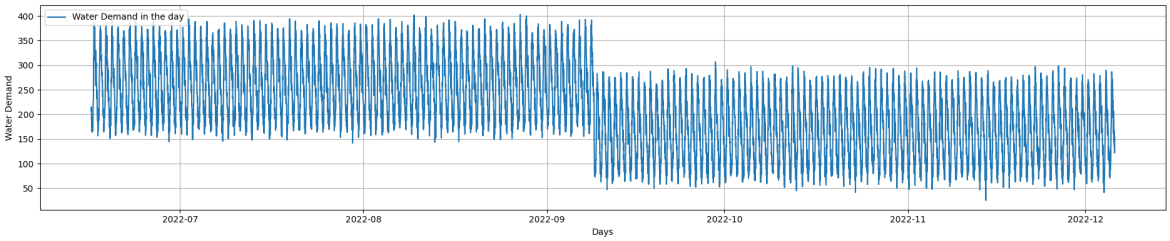

Figure 3.8: Artificial data with a sudden decrease drift

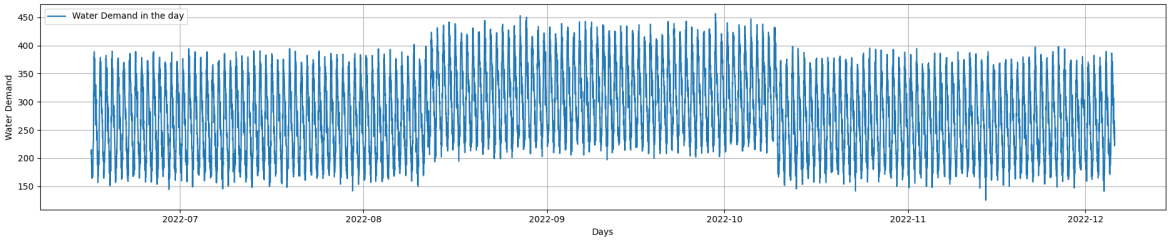

Figure 3.9: Artificial data with recurring drift

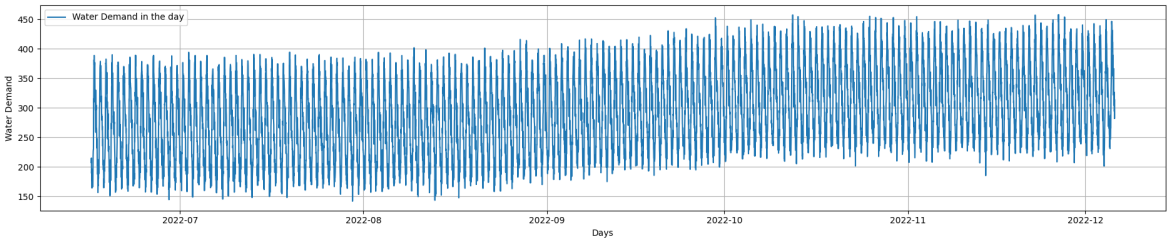

Figure 3.10: Artificial data with incremental drift

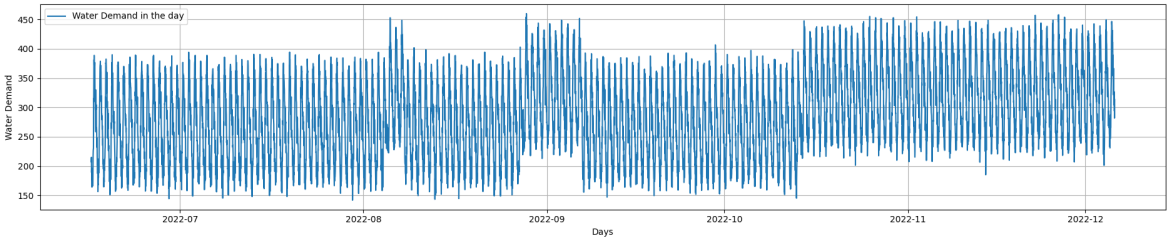

Figure 3.11: Artificial data with gradual drift

3.5.2 Real-world data sets

For the real-world data sets, we obtained consumption data from three points selected by SCUBIC. The criteria for selecting these points were as follows: one point where there were few visual changes in the data, another point where there was a clear change in the data, and a third point where the data was constantly changing. The data for each of these three points spans from June 1st, 2022 to April 11th, 2023. So in each data set 294 days are tested, and since in each day have 48 points, a total of 14112 points are tested.

Throughout this work I'll refer to those data as real case 1 (Figure 3.12), real case 2 (Figure 3.13) and real case 3 (Figure 3.14).

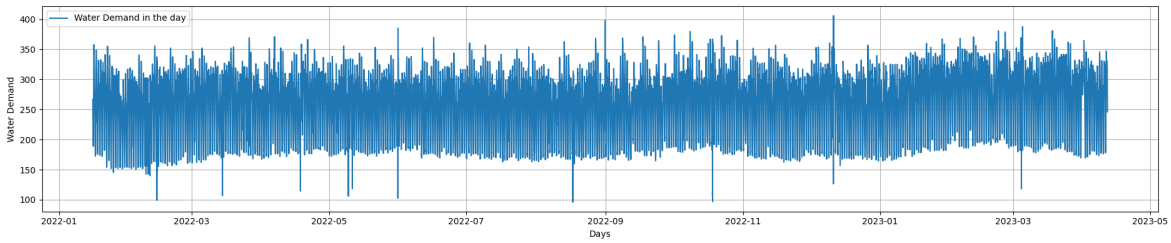

Figure 3.12: Real case data with few changes

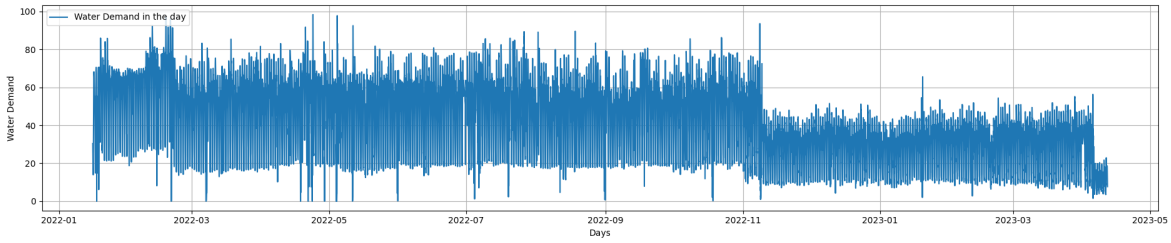

Figure 3.13: Real case data with a clearly change

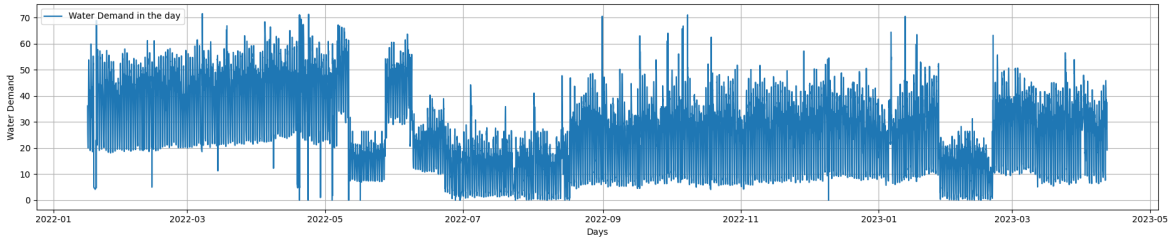

Figure 3.14: Real case data with lots of changes

3.6 Results evaluation

Two metrics that are usually used in a time series problem are the Root Mean Square Error (RMSE) and coefficient of determination (R^2).

The RMSE measures the difference between the predicted value and the real one, and can be defined as

$$RMSE = \sqrt{\frac{\sum_{t=1}^{T}(\hat{y}_t - y_t)^2}{T}} \tag{3.12}$$

where $\hat{y}_t$ is the value predicted, y_t is the real value and T is the number of variables that were observed.

The R^2 determines the proportion of variance in the dependent variable that can be explained by the independent variable. Mathematically to calculate the R^2 it is necessary to calculate the residual sum of squares (SS_{res}) and the total sum of squares (SS_{tot}).

$$SS_{res} = \sum_{t=1}^{T} (\hat{y}_t - \bar{y})^2 \tag{3.13}$$

$$SS_{tot} = \sum_{t=1}^{T} (y_t - \bar{y})^2 \tag{3.14}$$

where y_t is the real value for the observation t, $\hat{y}_t$ is the predicted value and $\bar{y}$ its the mean of the y value.

After those calculus, the R^2 is calculated:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{3.15}$$

where if the R^2 value is 1, it means that the model has achieved the best possible performance. A value of 0 indicates that the predicted values are no better than the mean. If the value is below 0, it means that the model's predictions are worse than if it were simply using the mean.

These two metrics are used in different ways: the first one involves calculating their values every day after the 48 real data points have been collected, i.e, to give us an idea of how the model is performing on a day-to-day basis. The other approach is to calculate their values at the end of all predictions.

The duration of the experiments is also an analyzed metric. It is important to note that each experiment has associated variability; each time an experiment is executed, it may take more or less time than the previous iteration of the same experiment. An optimal approach to analyze this metric would be to run each experiment multiple times and calculate the average duration. However, in this study, the time taken by each experiment is analyzed based on the most recent execution. The magnitude of time is in minutes.

3.6.1 Benchmarks

To compare the results of our approach, two benchmarks are used. The first benchmark is a basic approach where there is no retraining phase implemented. This serves as a baseline for comparison. The second benchmark involves retraining the model every 15 days without any specific criteria. This approach is commonly used to address the decay in predictions and provide insight into how the proposed retraining approach's compares to a more traditional method.

CHAPTER 4

Results and Discussion

This chapter is divided into two parts. The first part presents the results of the artificial datasets, while the second part discusses the outcomes of the real-world datasets.

4.1 Results from artificial datasets

This section aims to evaluate the effectiveness of the drift detection methods used for artificial datasets. The main objective is to identify the most suitable drift detectors and determine the optimal approach for drift detection, considering both delayed and immediate retraining. Additionally, the analysis includes the evaluation of RMSE and R^2 to study the actual impact of retraining.

For this section, only two drift detection approaches were employed since the data does not exhibit weekly seasonality. These approaches consist of the traditional method and the error-rate method.

Initially, the parameters were optimized in an attempt to obtain satisfactory results. However, despite multiple attempts, the error-rate method failed to yield satisfactory outcomes. In most cases, the method either failed to detect drifts or inaccurately identified their locations.

It is important to note that the results for sudden increase drifts and sudden decrease drifts were practically identical, making it unnecessary to present both sets of results.

Furthermore, each type of drift is analyzed to assess the adaptation capabilities of the incremental learning algorithm.

4.1.1 Preparing the XGBoost model

Afterward, the experiments were analyzed with retraining using 30, 50, or 100 days of data. Although a comprehensive analysis was not conducted in this study, retraining with 30 days appears to have a more positive impact on the overall model performance. Additionally, considering its lower computational cost, 30 days was selected as the number of days used for retraining.

4.1.2 Sudden Drift

For the dataset with a sudden drift, you can observe the performance of the drift detectors in Figures 4.1 and 4.2. The red trace indicates the moment when the drift is detected.

Both the PH and the KSWIN detectors have correctly identified the drift at the same location. In the case of ADWIN, although it could be argued that the first trace is detected accurately, albeit with a delay, it has also triggered a false alarm.

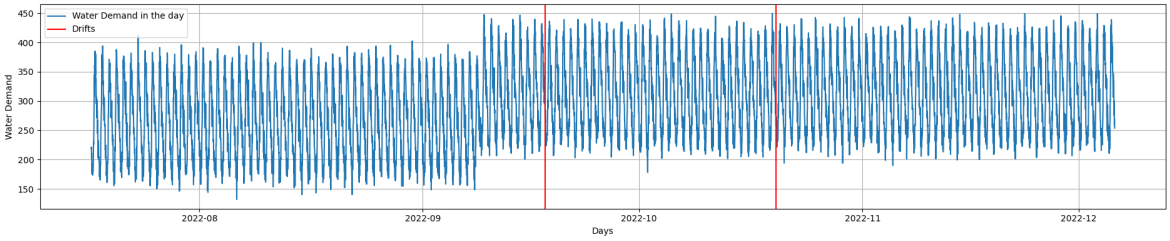

Figure 4.1: Drifts detected by ADWIN on a dataset with sudden drift

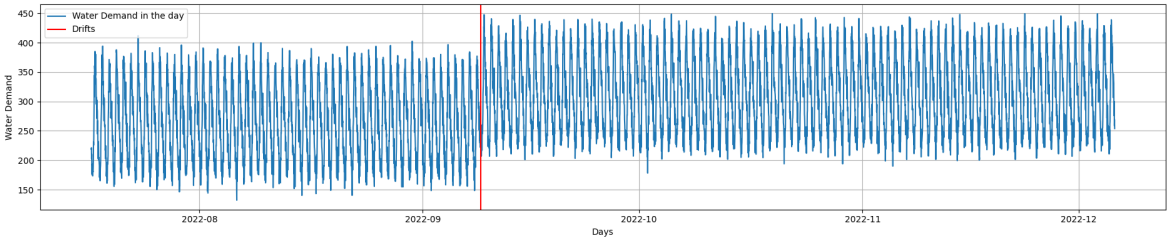

Figure 4.2: Drifts detected by PH and KSWIN on a dataset with sudden drift

Although ADWIN had poor performance in detecting drifts, it is still interesting to observe the impact of this case on the overall performance of the model. Figures 4.3 and 4.4 illustrate the influence of retraining on the model's performance, regardless of the process. The utilization of the ADWIN approach significantly improved the overall model performance. However, delaying the retraining did not enhance the model's performance, which is understandable since the drift is already detected with a delay.

Regarding Figure 4.4, it is evident that delaying the retraining greatly enhances the performance. When considering the number of days delayed, a delay of 14 days appears to further improve the model's performance. It is noticeable that retraining immediately after the drift does not have a significant influence on the overall performance; the influence is only observed in the initial days after the drift has been detected.

In Table 4.2, the impact of each experiment is presented. The best R^2 and RMSE scores are achieved by KSWIN and PH with a 14-day delay, resulting in a 23.03% improvement in terms of the R^2 metric and a 26.27% improvement in terms of RMSE. It is also notable that all experiments yield better results compared to the no retraining experiment. It is worth noting that there was minimal variation observed in the time metric across all the experiments.

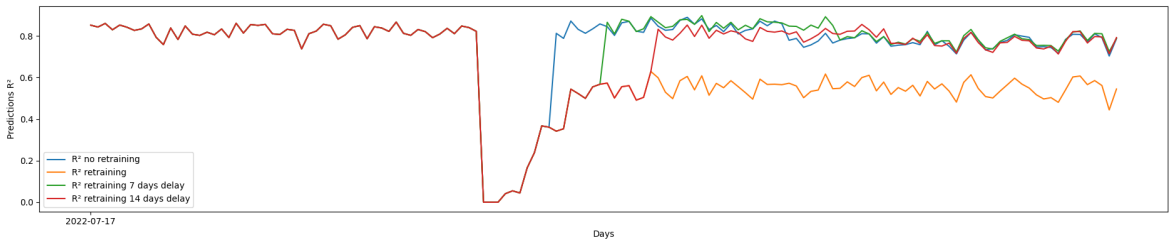

Figure 4.3: Comparison between the daily R^2 of the experiments evolving ADWIN and the no retrain benchmark in the sudden drift data

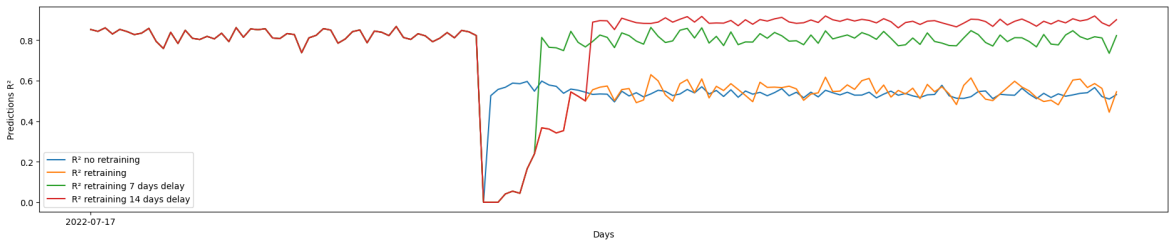

Figure 4.4: Comparison between the daily R^2 of the experiments evolving PH/KSWIN and the no retrain benchmark in the sudden drift data

	No Retrain	ADWIN	ADWIN 7	ADWIN 14	PH&KSWIN	PH&KSWIN 7	PH&KSWIN 14
R^2	0.6646	0.7882	0.7800	0.7591	0.6853	0.7948	**0.8177**
RMSE	40.3651	32.0798	32.6921	34.2096	39.0996	31.5721	**29.7613**
Time	0.14	0.41	0.33	0.36	0.25	0.29	0.3

Table 4.1: Overall performance by the drift detectors on a data set with sudden drift

Sudden Drift - incremental learning

Both incremental model experiments, as shown in Table 4.2, demonstrated improvements in both the R^2 and RMSE metrics compared to the best retraining experiment and even more so when compared to the benchmark. This type of modeling, specifically in the case of sudden drifts, exhibits rapid and effective responsiveness. However, the only issue encountered with this experiment was the significantly increased time metric, taking considerably longer than the retraining experiences (almost 14 times more then the best retraining experience).

	No Retrain	Best Retrain	Incremental Daily	Incremental Weekly
R^2	0.6646	0.8177	**0.9478**	0.9467
RMSE	40.3651	29.7613	**15.9267**	16.1051
Time	0.14	0.30	4.49	2.17

Table 4.2: Overall performance by both incremental algorithms, the best retraining experience and the no retrain benchmark in the sudden drift data

In Figures 4.5 and 4.6, you can observe the substantial difference in the daily R^2 between the daily incremental model and the batch incremental model, compared to the benchmark.

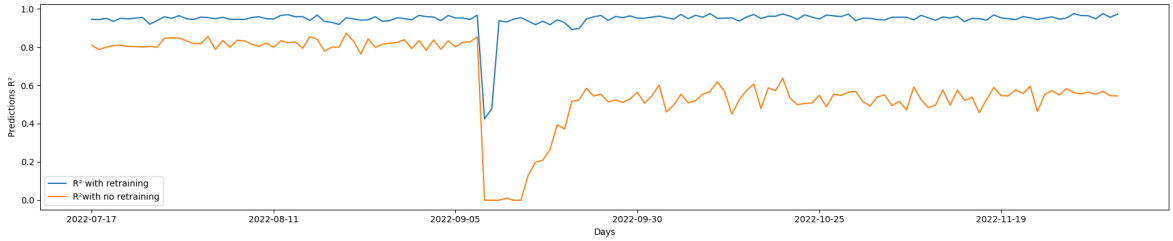

Figure 4.5: Comparison between the daily R^2 of the experience with daily incremental model and the no retrain benchmark in the sudden drift data

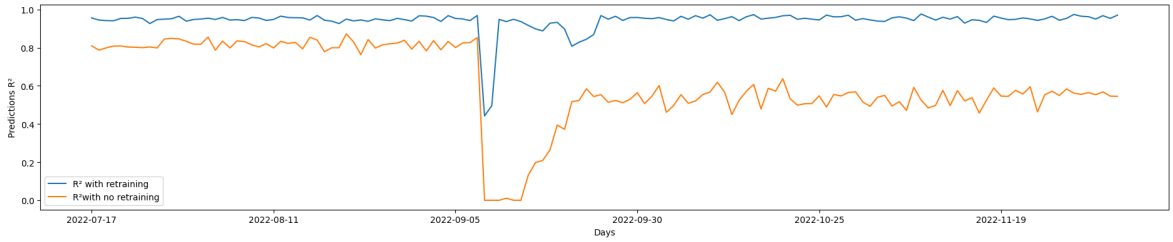

Figure 4.6: Comparison between the daily R^2 of the experience with bath incremental model and the no retrain benchmark in the sudden drift data

4.1.3 Recurrent Drift

In this type of drift, it is evident that ADWIN is encountering the same issue as the previous drift. In Figure 4.7, it can be assumed that ADWIN is also detecting drifts with a significant delay and is also prone to false alarms. On the other hand, the other two drift detectors, as shown in Figure 4.8, have accurately detected the drifts at the appropriate locations.

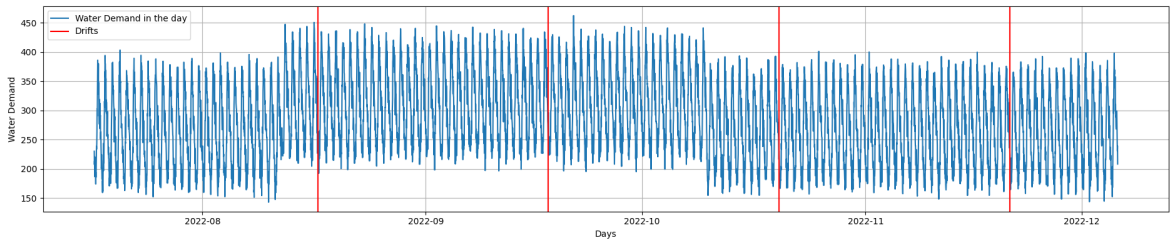

Figure 4.7: Drifts detected by ADWIN on a data set with recurrent drift

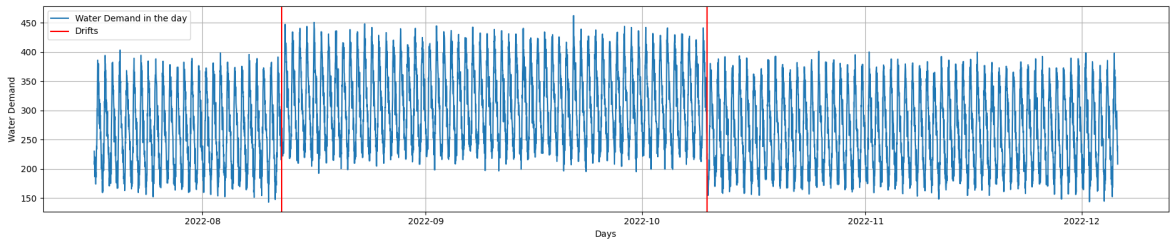

Figure 4.8: Drifts detected by PH and KSWIN on a data set with recurrent drift

The performance on this type of drift can be observed in Figures 4.9 and 4.10. Notably, in the final concept of the model, the no retraining experiment achieved good results as the

data reverted to the same pattern as in the beginning. However, it should be noted that the other models exhibited better overall performance.

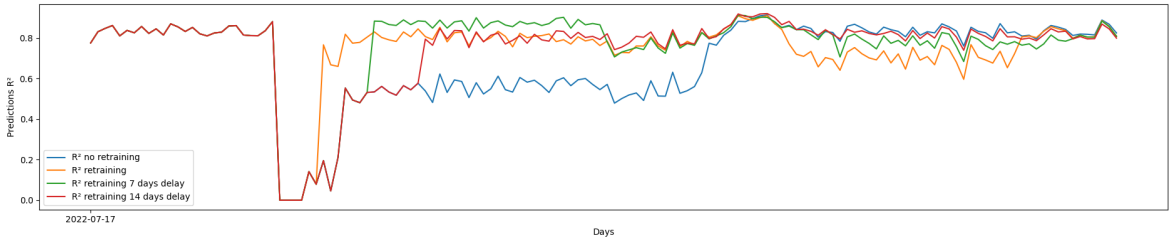

Figure 4.9: Comparison between the daily R^2 of the experiments evolving ADWIN and the no retrain benchmark in the recurrent drift data

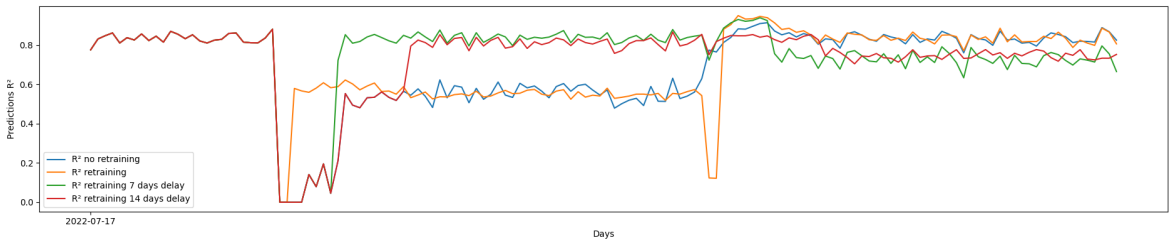

Figure 4.10: Comparison between the daily R^2 of the experiments evolving PH/KSWIN and the no retrain benchmark in the recurring drift data

The overall results (Table 4.3) indicate that the best result was achieved by ADWIN with a 7-day delay. However, taking into account the accuracy of drift detection, the best experiment is considered to be the one conducted with Page Hincley and KSWIN detectors, also with a 7-day delay. In this case, the overall improvement compared to no retraining is 8.67% in terms of R^2 and 11.94% in terms of RMSE. The time measure was not significantly different between the experiments.

	No Retrain	ADWIN	ADWIN 7	ADWIN 14	PH&KSWIN	PH&KSWIN 7	PH&KSWIN 14
R^2	0.7227	0.7811	**0.7916**	0.7783	0.7431	0.7854	0.7637
RMSE	36.5892	32.5111	**31.7221**	32.7204	35.2201	32.1878	33.7799
Time	0.18	0.44	0.42	0.45	0.31	0.33	0.35

Table 4.3: Overall performance by the drift detectors on a data set with recurrent drift

Recurring Drift - incremental learning

The daily incremental model and the batch incremental model have performed well in this type of drift, similar to the sudden drift dataset. However, the only drawback is the increased time required, as can be seen in Table 4.4. In this metric, the batch incremental model achieves half the time of the daily incremental model (reduces the time by 50.7%).

	No Retrain	Best Retrain	Incremental Daily	Incremental Weekly
R^2	0.7227	0.7916	**0.9411**	0.9328
RMSE	36.5892	31.7221	**16.9421**	18.1014
Time	0.18	0.45	3.37	1.71

Table 4.4: Overall performance by both incremental algorithms, the best retraining experience and the no retrain benchmark in the recurring drift data

In Figures 4.11 and 4.12, significant differences can be observed in the daily R^2 between the daily incremental model experience and the batch incremental model experience.

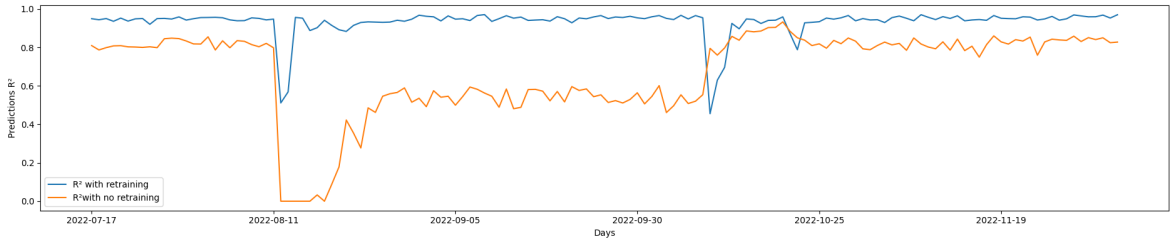

Figure 4.11: Comparison between the daily R^2 of the experience with bath incremental model and the no retrain benchmark in the recurring drift data

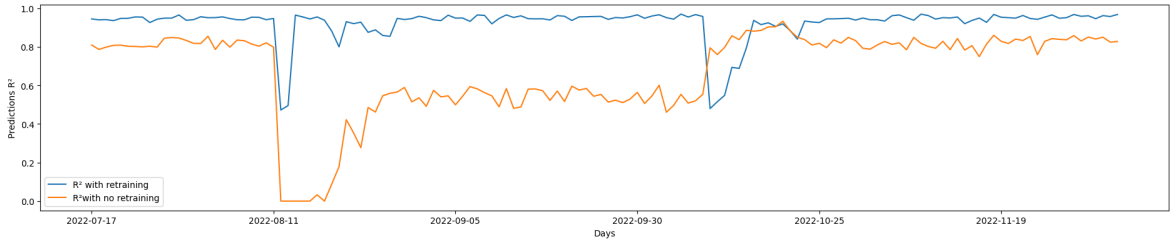

Figure 4.12: Comparison between the daily R^2 of the experience with bath incremental model and the no retrain benchmark in the reccuring drift data

4.1.4 Incremental Drift

This type of drift is the most challenging to analyze in terms of whether the drifts are accurately detected or not. As depicted in Figures 4.13, 4.14, and 4.15, all the drift detectors have detected drifts at different locations. ADWIN appears to perform better in detecting this type of drift compared to the others previously discussed. PH, on the other hand, detected the highest number of drifts, indicating that they were likely well detected. Similarly, the KSWIN drift detector identified only one drift, occurring in the middle of the increment.

The performance of the different experiments (Figures 4.16, 4.17, and 4.18) shows a significant improvement compared to no retraining. The main difference appears to be in the experiments with PH, as the drifts are detected earlier, allowing the algorithms to adapt sooner as well.

As observed in Tables 4.5, 4.6, and 4.7, delaying the drifts appears to have a lesser effect. However, for each type of drift, a delay of 7 days seems to yield better results. Among

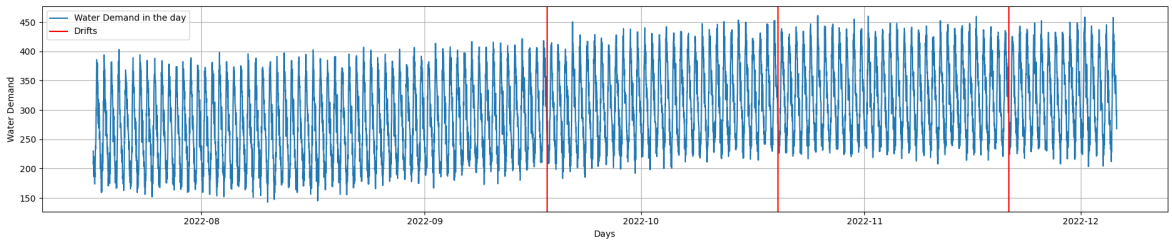

Figure 4.13: Drifts detected by ADWIN on a data set with incremental drift

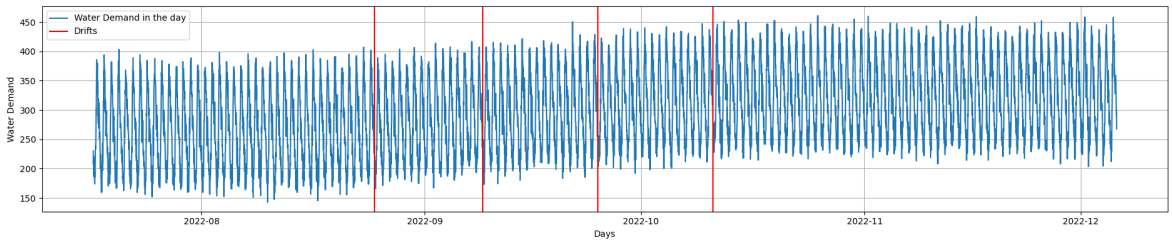

Figure 4.14: Drifts detected by PH on a data set with incremental drift

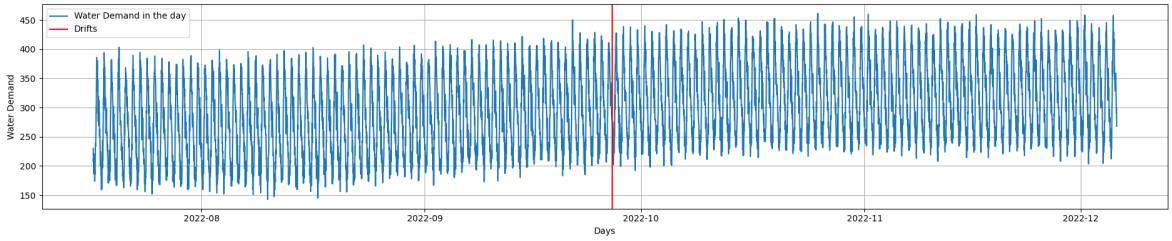

Figure 4.15: Drifts detected by KSWIN on a data set with incremental drift

the drift detection methods, PH emerges as the winner, achieving the lowest RMSE and highest R^2 values. The best performing version, PH with a 7-day delay, improves upon the no retraining model by 27.56% in terms of RMSE and 25.39% in terms of R^2

	No Retrain	ADWIN	ADWIN 7	ADWIN 14
R^2	0.6518	**0.7935**	0.7871	0.7798
RMSE	41.1419	**31.6807**	32.1720	32.7188
Time	0.15	0.6	0.54	0.44

Table 4.5: Overall performance by ADWIN on a data set with incremental drift

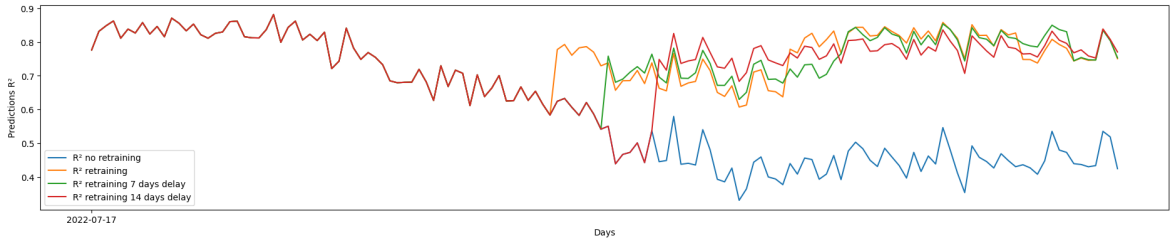

Figure 4.16: Comparison between the daily R^2 of the experiments evolving ADWIN and the no retrain benchmark in the incremental drift data

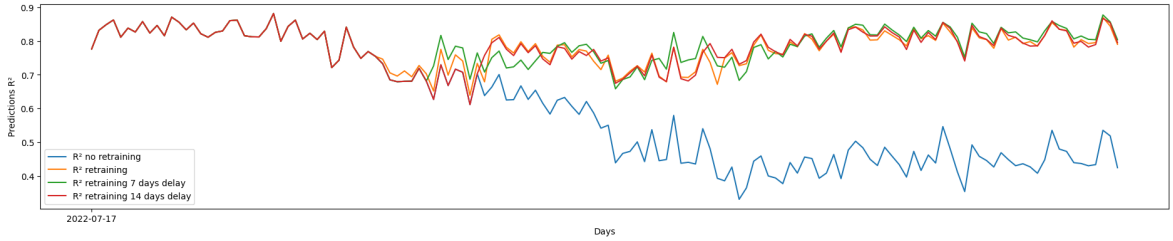

Figure 4.17: Comparison between the daily R^2 of the experiments evolving PH and the no retrain benchmark in the incremental drift data

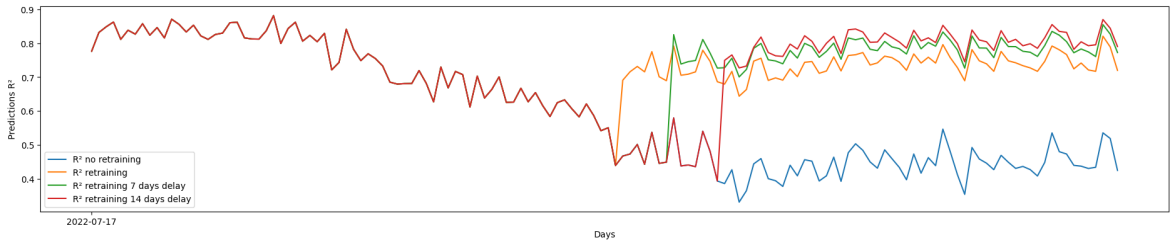

Figure 4.18: Comparison between the daily R^2 of the experiments evolving KSWIN and the no retrain benchmark in the incremental drift data

	No Retrain	PH	PH 7	PH 14
R^2	0.65178	0.81345	**0.81727**	0.81265
RMSE	41.1419	30.4674	**29.9694**	30.6634
Time	0.15	0.53	0.47	0.5

Table 4.6: Overall performance by page-hincley on a data set with incremental drift

	No Retrain	KSWIN	KSWIN 7	KSWIN 14
R^2	0.6517	0.7330	0.7735	**0.7879**
RMSE	41.1419	36.125	33.3438	**32.1959**
Time	0.15	0.53	0.47	0.5

Table 4.7: Overall performance by KSWIN in a data set with incremental drift

Incremental Drift - incremental learning

In this type of data drift, both incremental learning experiences have outperformed the benchmark and the best retrain experience. The results in Table 4.8 indicate that time is, again, the only potential issue. Regard which incremental approach is better, since the RMSE and the R^2 are close, the time factor is decisive. The batch incremental approach spends half the time that the daily incremental approach achieved, leading it to be considered the best experience.

	No Retrain	Best Retrain	Incremental Daily	Incremental Weekly
R^2	0.6517	0.81727	**0.9402**	0.9332
RMSE	41.1419	29.9694	**17.0772**	18.0440
Time	0.15	0.45	3.24	1.93

Table 4.8: Overall performance by both incremental algorithms, the best retraining experience and the no retrain benchmark in the incremental drift data

In Figures 4.19 and 4.20, significant differences can be observed in the daily R^2 performance between both incremental experiences and the benchmark.

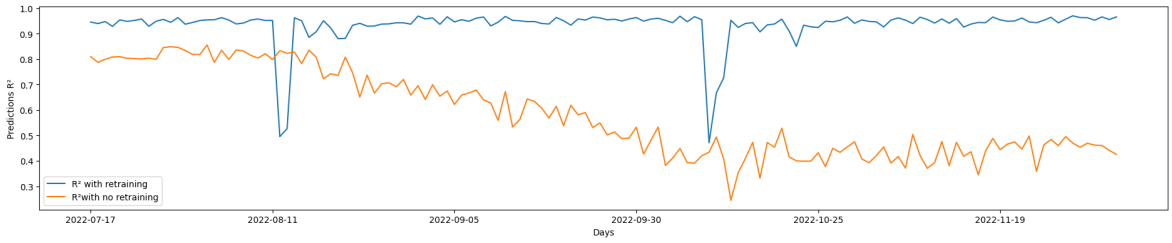

Figure 4.19: Comparison between the daily R^2 of the experience with bath incremental model and the no retrain benchmark in the incremental drift data

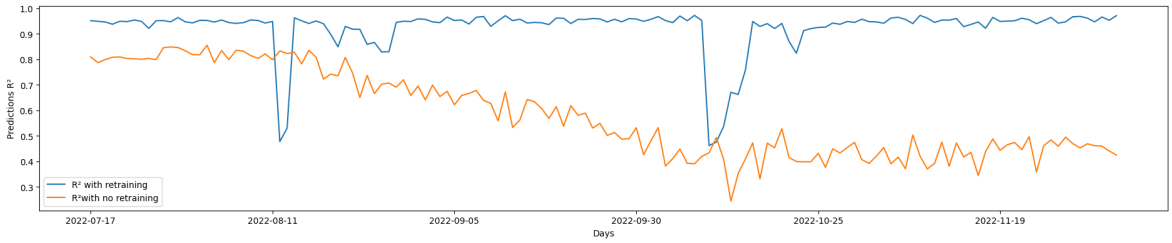

Figure 4.20: Comparison between the daily R^2 of the experience with bath incremental model and the no retrain benchmark in the incremental drift data

4.1.5 Gradual Drift

For the last drift that was tested, both KSWIN and PH detectors have successfully detected the drifts in the correct locations (Figure 4.2). However, ADWIN was unable to detect any drift effectively and only detected one drift towards the end. One possible explanation for this is that the delta parameter may need to be more sensitive. However, the chosen delta parameter was based on the best results obtained across all drifts. If a more

sensitive parameter were chosen, it could lead to a higher number of false positives in the other drift scenarios.

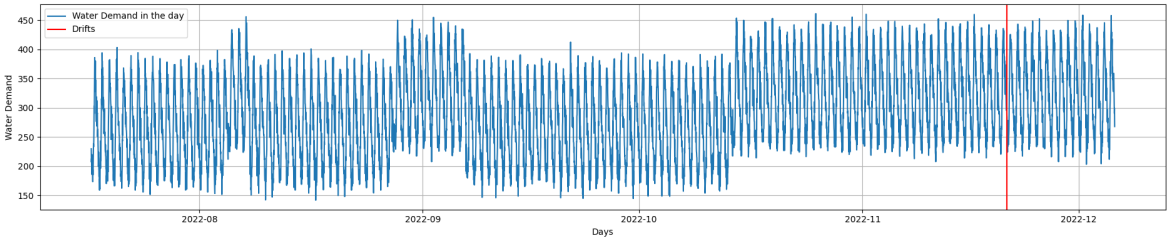

Figure 4.21: Drifts detected by ADWIN on a data set with gradual drift

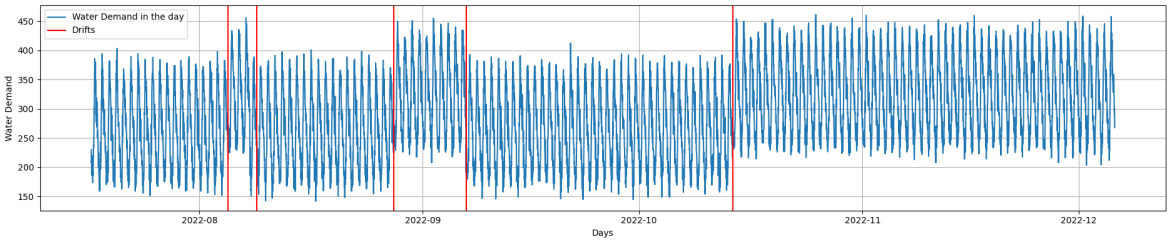

Figure 4.22: Drifts detected by page-hincley and KSWIN on a data set with gradual drift

Therefore, it is only necessary to demonstrate the performance of KSWIN and PH (Figure 4.23). The results in this case are the most challenging to analyze. However, particularly in the latter part of the data, the retraining with 7 days delay and 14 days delay showed excellent performance.

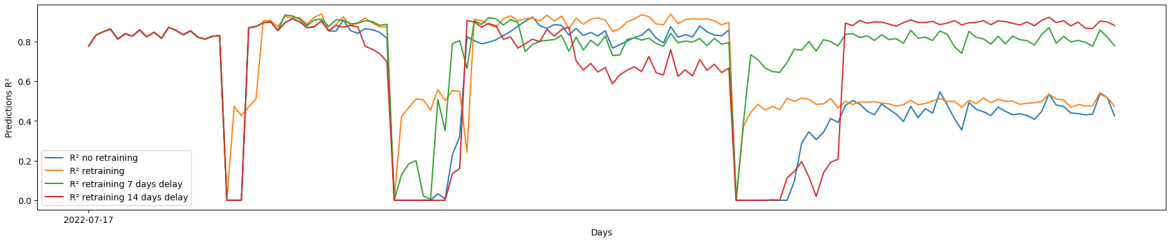

Figure 4.23: Comparison between the daily R^2 of the experiments evolving PH/KSWIN and the no retrain benchmark in the gradual drift data

As confirmed in Table 4.10, PH and KSWIN exhibit the best overall performance in both metrics. They improve the baseline model by 25.76% on the R^2 metric and by 24.43% on the RMSE. The time metric, corroborating with the previous cases, don't present significant differences.

	No Retrain	ADWIN	PH&KSWIN	PH&KSWIN 7	PH&KSWIN 14
R^2	0.62478	0.64999	0.72991	**0.7857**	0.68271
RMSE	43.93796	42.43666	37.27812	**33.2055**	40.40395
Time	0.22	0.36	0.58	0.54	0.48

Table 4.9: Overall performance by the drift detectors on a data set with gradual drift

Gradual Drift - incremental learning

For this final artificial data with gradual drift, as seen in Table 4.10, both incremental experiences have achieved better results in terms of the RMSE and R^2 metrics. However, the only drawback is once again the time required for these experiences. Being the batch incremental approach better on that metric, it makes it also the best overall option.

	No Retrain	Best Retrain	Incremental Daily	Incremental Weekly
R^2	0.62478	0.7857	**0.9043**	0.8548
RMSE	43.93796	33.2055	**22.2056**	27.3526
Time	0.22	0.54	3.34	2.00

Table 4.10: Overall performance by both incremental algorithms, the best retraining experience and the no retrain benchmark in the gradual drift data

In Figures 4.24 and 4.25, the performance advantages of the incremental models were not as significant compared to previous cases. However, the incremental models still outperformed the other approaches.

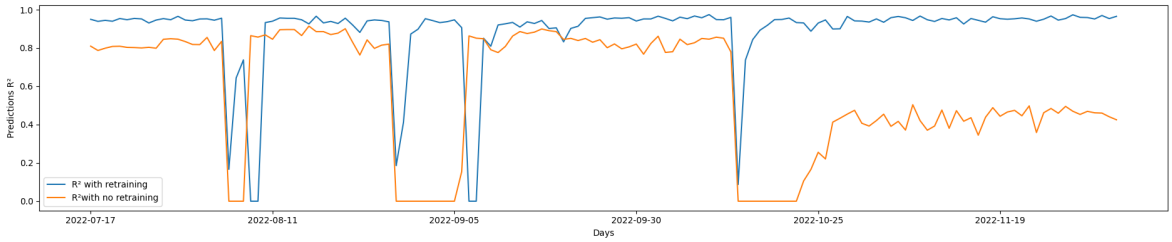

Figure 4.24: Comparison between the daily R^2 of the experience with bath incremental model and the no retrain benchmark in the gradual drift data

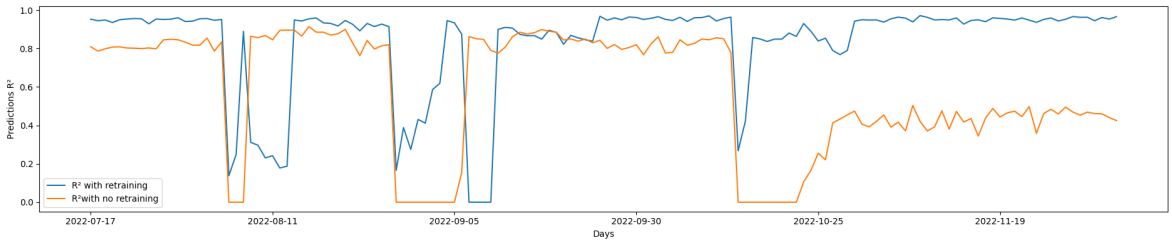

Figure 4.25: Comparison between the daily R^2 of the experience with bath incremental model and the no retrain benchmark in the gradual drift data

4.1.6 Conclusion concerning artificial data

Based on the results obtained from the artificial data, it can be observed that even though the ADWIN drift detector struggles to accurately locate drifts, its utilization still leads to performance improvements compared to no retraining.

When drifts are well detected, immediate retraining after detection does not appear to be the most effective approach. It should be noted, however, that this approach still results in improvements. On the other hand, delaying the retraining by 7 days in all experiments seems to be the optimal approach, as delaying it by 14 days does not provide significant performance benefits.

It is important to mention that while the KSWIN drift detector performs well, it occasionally detects drifts with a delay of 2 or 3 days. Consequently, each time the code is run with KSWIN for drift detection, it may detect drifts in different locations. The presented results represent the best achieved, but it is worth noting that this drift detector exhibits more variability compared to PH.

Regarding the two incremental learning experiences, both demonstrated superior performance in terms of R^2 and RMSE across all experiments. The batch incremental experience, despite being slightly inferior in each experiment, requires only half the time to complete. Therefore, it may be a better choice, at least for the artificial data.

4.2 Results from Real data sets

This section aims to utilize the best experiences obtained from the artificial datasets. In addition to these experiments, two additional experiments are conducted, taking into account the seasonality of the data in the drift detectors. The primary objective is to identify the most suitable approach for each real case data. In addition to the benchmark used in the artificial data, the benchmark that incorporates periodic retraining is also employed. Furthermore, an analysis that compares two incremental learning approaches with the best retraining experience and the no retraining benchmark is made.

4.2.1 Effect of Training Data Size on Model Performance

The study on the number of days for batch retraining began in conjunction with the initial grid search approach for the models. Retraining was also performed whenever visual drifts occurred. Unfortunately, the R^2, RMSE, and time metrics for these experiments were not recorded. However, consistent conclusions were drawn across all cases. The model's performance deteriorated as the batch size decreased, while the time metric exhibited the opposite trend. Therefore, it was determined that a trade-off point needed to be identified, and it was found that using 50 days of data yielded the best results.

4.2.2 Real case 1 - Retraining Experiences

For the real case 1, XGBoost provided the following optimal parameters:

- Learning rate: 0.01

- Max depth: 3
- Min child weight: 500
- Number of estimators: 7

The next step was to generate two benchmarks. The first benchmark involved training a model on the first 50 days of data and then making predictions and saving them. The second benchmark involved retraining the model every 15 days, resulting in a total of 28 retrainings.

As shown in Figure 4.26, the daily R^2 values between the two benchmarks are very similar, indicating that blindly retraining the model at fixed intervals does not provide significant improvements.

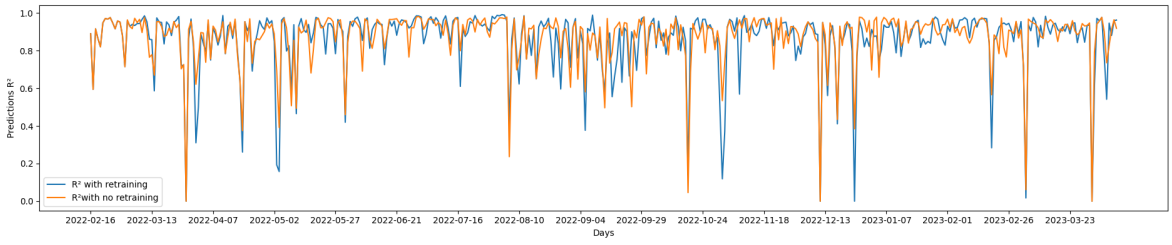

Figure 4.26: Comparison between the daily R^2 of the two benchmarks in the real case 1

The overall RMSE and R^2 values (see Table 4.11) confirm this observation and even suggest that the no retrain benchmark performed slightly better overall, although the differences may not be significant. Furthermore, the time taken by each experience supports this finding, as the no retrain experience was approximately 3.25 times faster than the other benchmark while still achieving slightly better overall R^2 and RMSE values, as mentioned earlier.

Regarding the experiences that include the drift detectors, the results are presented in the three tables below (Tables: 4.11, 4.12, 4.13).

The traditional experiences (Table 4.11) were not able to outperform any of the benchmarks. However, similar results were achieved among the experiences, which are also comparable to the differences between the two benchmarks.

The drift detector KSWIN detected 66 drifts, which seems to be an exaggerated number considering that visually the data doesn't appear to have significant changes. This resulted in a significantly longer runtime for the experience, without corresponding performance improvements.

On the other hand, the drift detector PH performed slightly better by detecting 21 drifts. Both experiences using PH achieved similar results compared to the other experiences. It can be considered a better experience than the benchmark that retrains periodically, as it required fewer retraining events while achieving similar results. It seems to be the experience where the drifts are the most well detected as the Figure 4.27 shows.

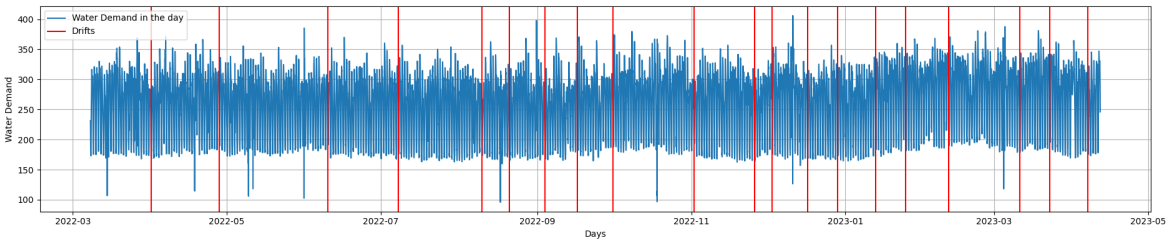

Figure 4.27: Demands of the real case 1 along with the drifts detected by the traditional PH

In the traditional approach experiences, delaying the retrain phase seems to be slightly more beneficial in the case of PH and roughly equal in the case of KSWIN.

	No Retrain	Retrain 15	PH	PH 7	KSWIN	KSWIN 7
Retrains	0	28	21	21	66	66
R^2	**0.8811**	0.8729	0.8661	0.8720	0.8631	0.8622
RMSE	**18.1551**	18.7724	19.2619	18.8334	19.4775	19.5443
Time	0.63	2.57	2.57	2.01	5.19	5.26

Table 4.11: Results of the traditional approach experiences in the real case 1

Regarding the performance of the weekend day experiences (Table 4.12), both drift detectors seem to have detected fewer false alarms compared to the traditional approach. However, these experiences were still unable to outperform the benchmarks, and the results remain similar and seemingly insignificant.

Delaying the retraining phase in these experiences slightly deteriorated the results for PH and had identical results for the KSWIN experiences

	PH Weekend	PH 7 Weekend	KSWIN Weekend	KSWIN 7 Weekend
Retrains	2	2	45	45
R^2	**0.8650**	0.8519	0.8614	0.8649
RMSE	**19.3473**	20.2621	19.6021	19.3483
Time	3.02	2.85	4.58	4.42

Table 4.12: Results of the weekend day approach experiences in the real case 1

In the weekly approach experiments (Table 4.13), the first overall result emerged, surpassing the benchmarks. This result was achieved by the PH experience with a delay. It detected only one drift, and the differences observed are minimal, as confirmed in Figure 4.28, which displays the daily $RMSE$ of this experience compared to the no retrain benchmark.

The experiments using the evolving KSWIN also exhibited fewer false alarms compared to the other experiments. However, the error metric results still did not outperform the benchmarks.

Interesting to analyze all the results and observe that regardless of the number of retrains,

	PH weekly	PH 7 weekly	KSWIN weekly	KSWIN 7 weekly
Retrains	1	1	41	41
R^2	0.8808	**0.8824**	0.8696	0.8684
RMSE	18.1797	**18.0541**	19.0149	19.0966
Time	2.09	2.44	3.4	2.85

Table 4.13: Results of the week approach experiences in the real case 1

only one experiment was able to improve upon the results of the no retraining benchmark. However, this best-performing experiment took approximately three times longer than the benchmark. Therefore, I consider the benchmark to be the best among all the retraining experiences.

Figure 4.28 presents a box plot illustrating the daily RMSE of the three experiences that achieved the best overall RMSE. The box plot demonstrates the similar results across all experiences, further emphasizing that the deciding factor for determining the best experience should be the time required.

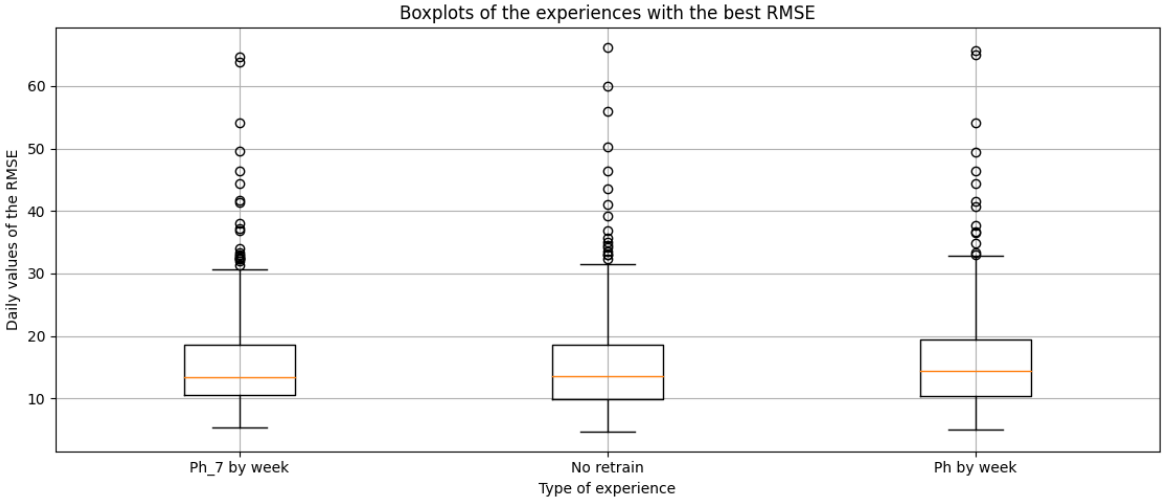

Figure 4.28: Box plot of the best results in the real case 1

The conclusions drawn from this analysis are that when a dataset exhibits minimal changes, it is preferable to stick with the same model rather than retraining it. Retraining incurs additional costs in terms of time and computational resources. However, if a company has both the time and resources available, implementing the PH drift detector, as demonstrated in the week experience, appears to be a viable option. The PH detector exhibited minimal false alarms and achieved slightly better performance compared to other approaches.

4.2.3 Real Case 1 - Incremental Learning

After conducting the retraining experiments, the incremental learning approach using ARF-Reg was tested by updating the model on a daily basis and in batches of 7 days.

Among the two incremental experiences, as shown in Table 4.14, the best results were achieved with the daily incremental approach. It achieved the lowest error metrics among all the experiences. However, since the model is updated every day, it consumes significantly more resources and takes approximately 15.55 times longer than the best retraining experience. Given the minimal differences in error metrics, this particular incremental machine learning model does not justify the additional time required.

	No Retrain	Best Retrain	Incremental Daily	Incremental Weekly
R^2	0.8811	0.8824	**0.8830**	0.8768
RMSE	18.1551	18.0541	**17.9467**	18.4176
Time	0.79	2.44	41.38	12.47

Table 4.14: Overall performance by both incremental algorithms, the best retraining experience and the no retrain benchmark in the real case 1

As you can see in Figure 4.29, the results between the benchmark and the best incremental approach are almost equal throughout the entire time period.

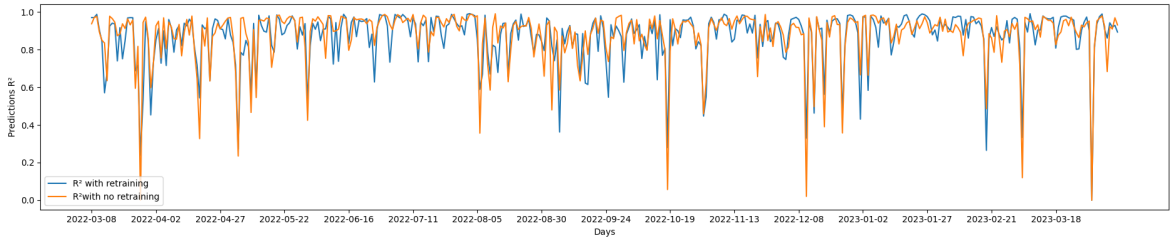

Figure 4.29: Comparison between the daily R^2 of the incremental experience and the no retrain benchmark in the real case 1

Regarding the experiences where the model is updated in batches, it can be observed that the incremental learning model significantly reduces the time it takes compared to the incremental daily experience. However, considering the trade-off between the time it takes and the results in terms of error metrics, it still does not outweigh the performance of the benchmark or the best retraining experience.

4.2.4 Real case 2 - Retraining Experiences

For real case 2, the optimal parameters obtained from XGBoost were:

- Learning rate: 0.005
- Max depth: 5
- Min child weight: 1000
- Number of estimators: 7

Learning rate: 0.005 Max depth: 5 Min child weight: 1000 Number of estimators: 7

The benchmarks were created in the same manner as in real case 1. As shown in figure 4.30, despite the fluctuations in the predictions, the model with no retraining is able to visually adapt to the sudden drift that occurs in the data. The daily R^2 comparison between the two benchmarks also reveals some differences, which appear to be more pronounced compared to real case 1. However, these differences do not seem to be significant, except for the last part of the model's performance, where the retraining benchmark shows better results.

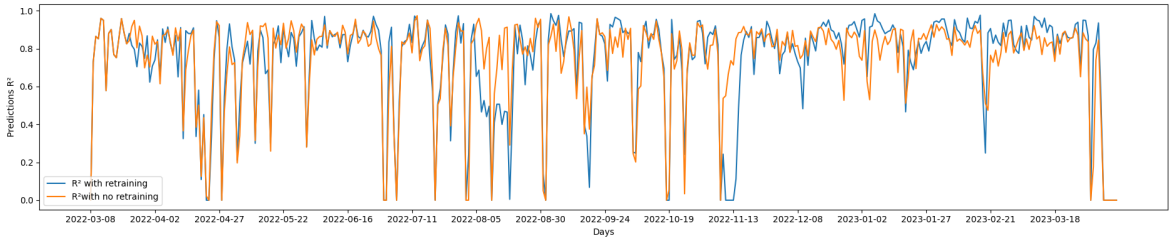

Figure 4.30: Comparison between the daily R^2 of the two benchmarks in the real case 2

Besides the improved performance of the last part by the retraining benchmark, the overall error metrics of the no retraining benchmark are better. Furthermore, it requires significantly less time compared to the retraining benchmark.

Analyzing the results of the traditional experiences, only one result outperforms the no retraining benchmark, and it was achieved by PH without any delay. However, the difference between them is negligible. From all the experiences, these one is one where its possible to see that the drifts have been well detected as its possible to see in the Figure 4.31.

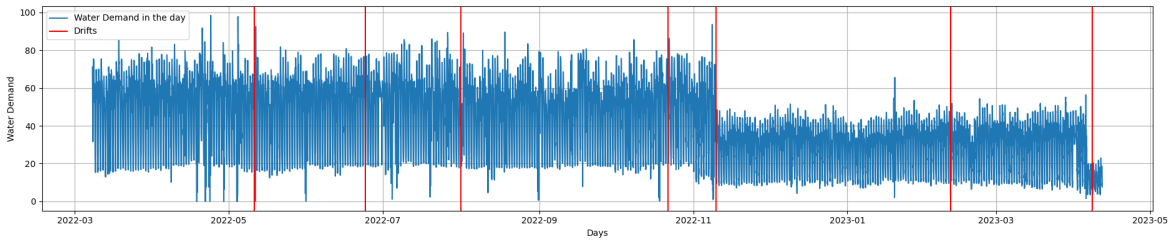

Figure 4.31: Demands from the real case 2 and the respective drifts that the traditional PH detected

Despite detecting 7 drifts, the PH experience takes more time than the no retraining benchmark, making the latter a better choice.

On the other hand, the KSWIN experiences continue to detect numerous drifts, indicating a high number of false alarms, considering that the dataset only has one evident drift.

Regarding the weekend experiences, there is only one experience that performs better than the no retraining benchmark, and it was achieved using KSWIN. However, it faces the same issue as the traditional PH approach, as it detects 33 drifts, and the improvements in

	No Retrain	Retrain 15	PH	PH 7	KSWIN	KSWIN 7
Retrains	0	28	7	7	26	26
R^2	0.7918	0.7781	**0.7935**	0.7834	0.7536	0.7916
RMSE	8.7305	9.0129	**8.6347**	8.9045	9.4972	8.7355
Time	0.62	2.88	1.16	1.04	3.41	2.86

Table 4.15: Results of real case 2 traditionall experiences

the error metrics are also insignificant.

Overall, the results of the weekend experiences are very close to each other. Interestingly, these experiences detect more drifts with both drift detectors compared to the traditional experience.

	PH Weekend	PH 7 Weekend	KSWIN Weekend	KSWIN 7 Weekend
Retrains	8	8	33	33
R^2	0.7851	0.7786	**0.7956**	0.7895
RMSE	8.8689	9.0032	**8.6501**	8.7781
Time	1.29	1.33	3.59	4.27

Table 4.16: Results of the real case 2 experiences applied according if its weekend

Regarding the weekly experiences, there is one result that outperforms the no retraining benchmark. It was achieved using the PH drift detector, and again, the difference is small. In this case, only one drift is detected. However, considering all factors, the no retrain experience would still be a better option.

	PH weekly	PH 7 weekly	KSWIN weekly	KSWIN 7 weekly
Retrains	1	1	28	28
R^2	**0.7945**	0.7940	0.7878	0.7933
RMSE	**8.6744**	8.6840	8.8131	8.6988
Time	1.28	1.28	3.05	2.96

Table 4.17: Results of the real case 2 experiences applied weekly

In Figure 4.32, it is evident that the RMSE values of the two best RMSE experiences and the no retrain benchmark are very similar.

The conclusions that can be drawn from this real case are as follows: Firstly, the XGBoost model is capable of adapting to clear drifts on its own, and the retraining performed when the sudden drifts occur has shown improvements. However, overall, the model is capable of making good predictions even without retraining.

Secondly, the use of seasonality experiments did not demonstrate significant advantages compared to the traditional PH approach. Additionally, delaying the retraining in most

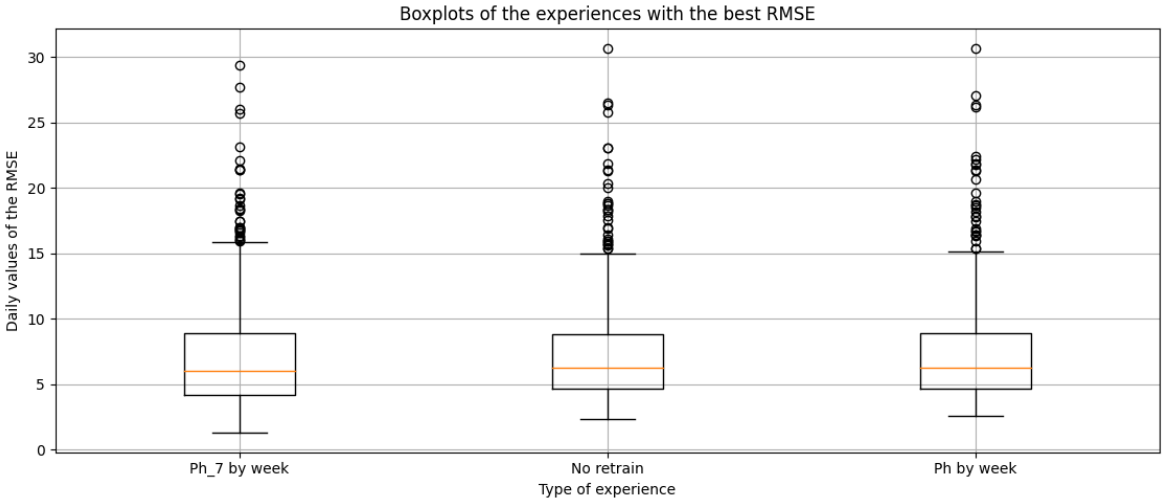

Figure 4.32: Box-plot of the 2 best RMSE and the no retrain benchmarks in the real case 2

experiments did not yield favorable results.

4.2.5 Real case 2 - incremental learning

With the analyzed retraining experiences, it is now time to examine the behavior of the incremental learning experiences. Table 4.18 shows the first significant differences in the error metrics. The daily incremental model improved the R^2 by 6.8% and the RMSE by 9.8%. However, this improvement came at the cost of significantly more time. On the other hand, the batch incremental learning model also demonstrated a substantial improvement compared to the no retrain benchmark, while taking only half the time of the daily incremental model.

	No Retrain	Best Retrain	Incremental Daily	Incremental Weekly
R^2	0.7918	0.7956	**0.8457**	0.8322
RMSE	8.7305	8.6501	**7.5190**	7.8371
Time	0.62	3.59	12.96	6.36

Table 4.18: Overall performance by both incremental algorithms, the best retraining experience and the no retrain benchmark in the real case 2

In Figure 4.33, it is clearly visible and confirmed that the daily incremental model has a higher R^2 compared to the no retraining benchmark.

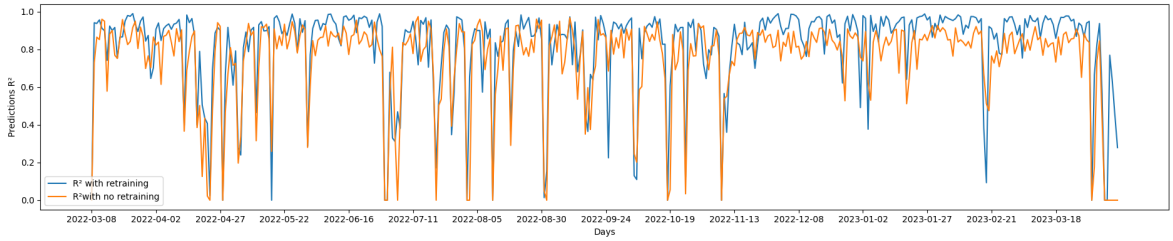

Figure 4.33: Comparison between the daily R^2 of the daily incremental model and the no retrain benchmark in the real case 2

The box plot of the daily RMSE for the best experiences and the no retrain benchmarks confirms the superior performance of the incremental model.

To conclude the analysis of the real case 2, it is clear that the incremental models are capable of achieving better results, even when compared to a retrain experience with 33 retrainings, as seen in the KSWIN weekend experience. These models are also able to outperform the benchmarks. However, a decision must be made considering the significant difference in the duration of the experiences. A trade-off needs to be considered, whether the computational cost is more important than achieving better performance.

If the incremental approach is not chosen, among the retraining experiences, the PH week experiences seem to be the better option, but the no retrain benchmark could also be considered.

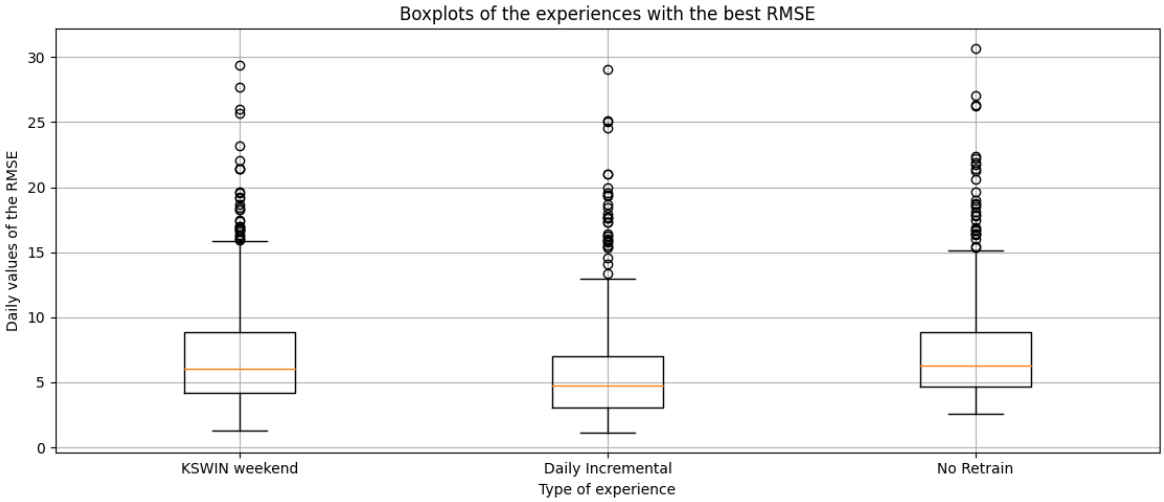

Figure 4.34: Comparison between the daily RMSE of the best retrain experience, the best incremental model and the no retrain benchmark in the real case 2

4.2.6 Real case 3 - Retraining Experiences

Using these parameters, the same process as the previous real-world cases was followed. Comparing the two benchmarks (Figure 4.35), it is evident that constant retraining has a clear advantage. The periodically retraining benchmark was able to improve the no retraining approach by approximately 18% in terms of R^2 and 11.4% in terms of RMSE, as shown in table 4.19. However, achieving these improvements required a significant amount of time.

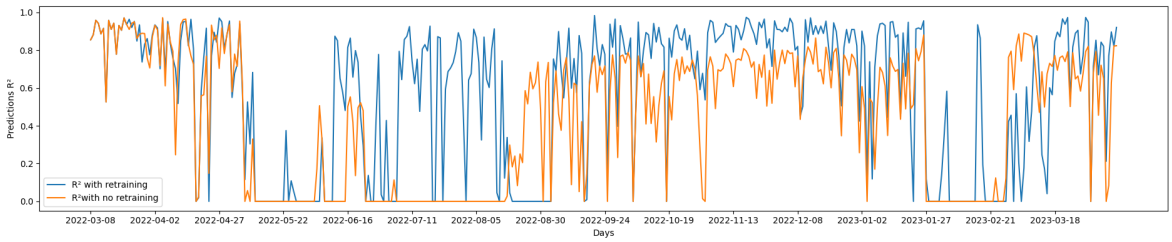

Figure 4.35: Comparison between the performance of the two benchmarks in the real case 3

Regarding the traditional approach experiences, all of them (except PH) were able to achieve better error metrics than the no retraining benchmark. Notably, the KSWIN experiment performed significantly better than the periodic retraining benchmark. It improved the R^2 by 9.1% and the RMSE by 8.5% compared to the benchmark. However, similar to the periodically retrained benchmark, it required a substantial amount of time to achieve these improvements.

On these experiences, an interesting observation emerged: the PH detector, despite having the worst overall result, was able to detect drifts where they clearly existed, as seen in Figure 4.37. In cases where drifts are visible, delaying the retraining phase appears to be a viable option.

	No Retrain	Retrain 15	PH	PH 7	KSWIN	KSWIN 7
Retrains	0	28	9	9	47	47
R^2	0.54297	0.6409	0.4286	0.6258	**0.6995**	0.6635
RMSE	9.8139	8.6994	10.9736	8.8808	**7.9584**	8.4216
Time	0.73	12.78	4.54	5.91	18.61	13.2

Table 4.19: Results of real case 3 traditional experiences

Additionally, several conclusions can be drawn from these results. First, detecting data drift accurately does not guarantee good overall performance. The best result was actually achieved by the KSWIN detector with a delay, despite detecting some false alarms. This suggests that other factors, such as the timing and frequency of retraining, play a significant role in performance.

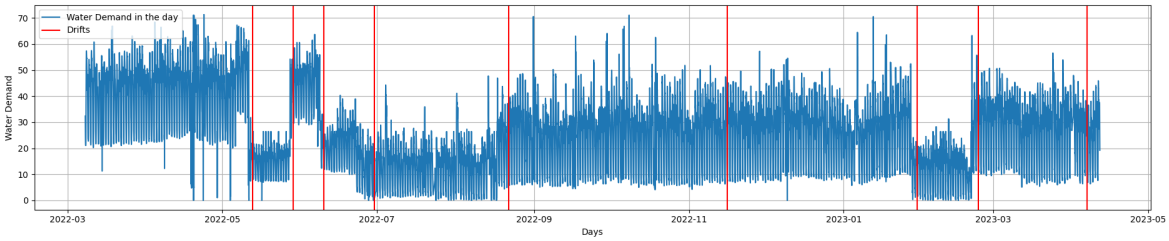

Figure 4.36: Drifts detected by traditional PH in the real case 3

From the weekend day experiences (Table 4.20), it can be observed that the KSWIN detector detected the same number of drifts as in the traditional approach. Additionally, it was the best-performing experience among the weekend day experiments. Although it was not as superior as the other experiences, it still outperformed both benchmarks. While it did consume a significant amount of time, it took less time than the periodically retraining benchmark.

	PH Weekend	PH 7 Weekend	KSWIN Weekend	KSWIN 7 Weekend
Retrains	13	13	47	47
R^2	0.6047	0.5033	**0.69141**	0.46251
RMSE	9.1267	10.2308	**8.0642**	10.6428
Time	3.85	0.95	11.51	0.91

Table 4.20: Results of the real case 3 experiences applied according if its weekend

Regarding the week experiences (Table 4.21), all of the experiences were able to outperform the no retraining benchmark. The KSWIN with delay showed the best performance, and it consumed relatively less time compared to the other top-performing experiences. However, the overall metric results did not surpass the other two best experiences. The differences in performance among them were minimal in this case.

	PH weekly	PH 7 weekly	KSWIN weekly	KSWIN 7 weekly
Retrains	8	8	42	42
R^2	0.5942	0.6565	0.6499	**0.6889**
RMSE	8.5082	8.5082	8.5998	**8.0972**
Time	2.35	2.44	9.04	8.79

Table 4.21: Results of the real case 3 experiences applied weekly

In Figure 4.37, its possible to observe the differences between the no retrain RMSE, the best PH experience, and the best experience achieved by the traditional KSWIN approach. The best PH experience is analyzed here because although the KSWIN experiences are generally better, they have a significant number of detected drifts, which can make them computationally expensive.

As seen in the figure, the daily RMSE achieved by the PH experience is not significantly different from the KSWIN experience. Therefore, the PH approach may serve as a good alternative if retraining too frequently is deemed too computationally expensive.

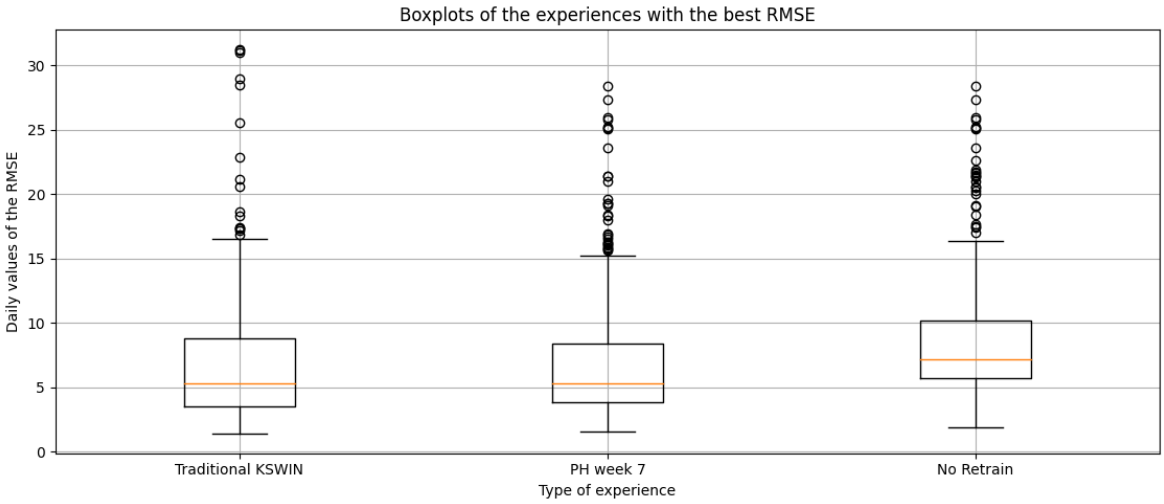

Figure 4.37: Box-plot of 3 different experiences in the real case 3

The conclusions drawn from these retraining experiences in real case 3 demonstrate that relying solely on the no retraining benchmark is not viable for data that undergoes constant changes. However, the choice of which retraining method to use may vary depending on the availability of computational resources.

In terms of optimal performance, all the different retraining experiences prove to be better options in the real-world scenario. Choosing an experience like the traditional KSWIN approach seems to yield the best results. On the other hand, if small differences in the results are not of great importance, the PH week experience appears to be a good option as it significantly reduces the frequency of retraining.

4.2.7 Real Case 3 - incremental learning

With the analysis of the retraining experiences completed, it is now time to examine how the incremental model behaves on this dataset that undergoes frequent changes.

Among the two incremental experiences, as shown in Table 4.22, only the daily incremental model is able to outperform the other results. Interestingly, in this particular dataset, the retraining phases have taken more time than in previous cases due to their increased complexity. This factor contributes to the daily incremental model achieving better results across all parameters compared to the best retraining experience and even most of the other experiences, including the periodically retraining benchmark.

	No Retrain	Best Retrain	Incremental Daily	Incremental Weekly
R^2	0.54297	0.6995	**0.7589**	0.6353
RMSE	9.8139	7.9584	**7.1277**	8.7666
Time	0.73	18.61	15.67	7.34

Table 4.22: Overall performance by both incremental algorithms, the best retraining experience and the no retrain benchmark in the real case 3

In Figure 4.38, it is evident the significant differences that the daily incremental model is able to achieve when compared to the non-retrained model.

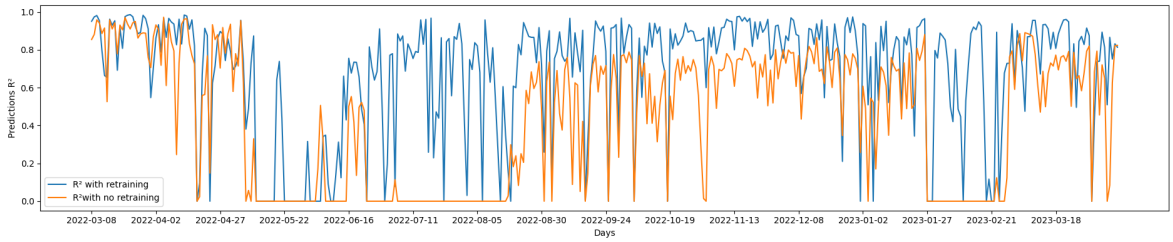

Figure 4.38: Comparison between the daily R^2 of the daily incremental model and the no retrain benchmark in the real case 3

To highlight these differences, Figure 4.39 presents a box plot comparing the best retraining experience, the daily incremental model, and the non-retrain benchmark in the real case 3.

On this type of dataset, the daily incremental learning approach seems to be the best option overall, considering both time and error metric results. It achieves significantly better performance compared to experiences with a large number of retrainings.

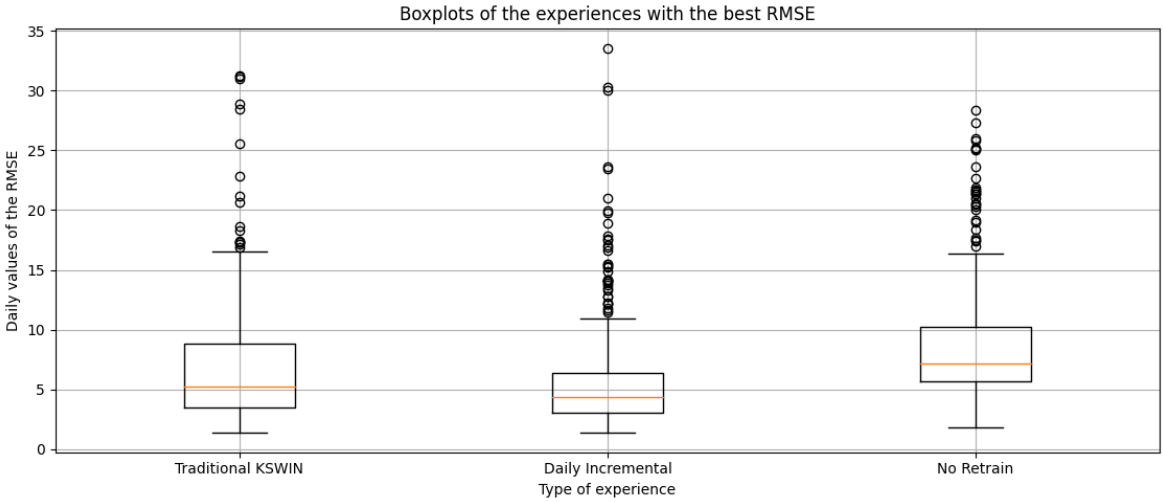

Figure 4.39: Box-plot of the best retraining experience, the daily incremental experience and the no retrain benchmark in the real case 3

4.3 Conclusions concerning real world datasets

From the analysis of the three real cases, several conclusions can be drawn. Firstly, as expected from the literature, detecting the right drift does not necessarily imply an overall performance improvement. In the first two cases, regardless of the experiments conducted, the results did not vary significantly. It is possible to enhance the results, but only by a small margin.

The true impact of drift detection becomes evident in real case 3, where the detection of drifts, regardless of their accuracy level, highlights the necessity of incorporating some form of drift detection. Furthermore, the tangible benefits of incremental modeling become apparent when the data undergo constant changes, indicating a clear advantage for this modeling approach. It is important to note that the specific model used in this study was not optimized, leaving significant room for further improvement.

Regarding the utilization of seasonality in drift detectors, it seems to offer some compensation in specific cases. However, given that traditional approaches generally detect drifts effectively, the incorporation of drift detectors in this manner may not be necessary. Nevertheless, it is important to recognize that this novel approach has potential for refinement.

The final observation from these real cases is that the number of false alarms produced by KSWIN may be excessive, with all experiments involving PH outperforming the drift detection component. Thus, two scenarios arise: if a company is willing to retrain the model more frequently, utilizing KSWIN would be beneficial; if not, the traditional pH approach may be a suitable alternative.

It is important to conduct a study to examine the trade-off between the accuracy of all the metrics before deciding which approach is better suited for a given scenario.

CHAPTER 5

Conclusion

This work aimed to find an automated method for retraining machine learning models. Several experiments were conducted using three well-known drift detectors: ADWIN, KSWIN and PH. Additional experiments were specifically carried out to test a new approach for working with these drift detectors, considering the seasonality of the data. Furthermore, an initial attempt was made at an incremental model to explore an alternative to the retraining process.

After conducting experiments involving retraining the model when drifts are detected, PH showed superior performance in drift detection, detecting fewer false drifts and outperforming both KSWIN and ADWIN. ADWIN, on the other hand, failed to detect drifts properly even in artificial data. KSWIN had the issue of generating a significant number of false drifts in real-world data, resulting in high computational effort.

Performance metrics, such as R^2 and RMSE, were analyzed. In the case of artificial data, all experiments involving drift detection outperformed the no retrain benchmark. Even ADWIN, despite its false alarms, showed improvement over the no retrain benchmark, demonstrating the importance of integrating a drift detector, even if imperfect, for achieving better predictions. Below are the best R^2 and RMSE performances for each artificial dataset:

- Sudden Drift: The best retraining experience was achieved by PH and KSWIN with a 14-day delay, resulting in a 23.04% improvement in the benchmark R^2 and a 26.27% reduction in RMSE. The incremental daily approach was able to further improve upon the best retraining experience, with a 15.9% increase in R^2 and a 46.4% decrease in RMSE.

- Recurrent Drift: The best retraining experience was achieved by ADWIN with a 7-day delay, resulting in a 9.53% improvement in the benchmark R^2 and a 13.30% reduction in RMSE. The incremental daily approach also improved the best retraining experience, with an 18.89% increase in R^2 and a 46.59% decrease in RMSE.

- Incremental Drift: The best retraining experience was achieved by PH with a 7-day delay, resulting in a 25.39% improvement in the benchmark R^2 and a 27.56% reduction in RMSE. The incremental daily approach improved upon the best retraining experience, with a 14.10% increase in R^2 and a 43.01% decrease in RMSE.

- Gradual Drift: The best retraining experience was achieved by PH and KSWIN with a 7-day delay, resulting in a 25.76% improvement in the benchmark R^2 and a 24.42% reduction in RMSE. The incremental daily approach improved the best retraining experience, with a 15.1% increase in R^2 and a 33.3% decrease in RMSE.

Regarding the results for the real cases 1 and 2, it was observed that both the model without retraining and the periodic retraining approach performed well when the data was stable. The differences between the benchmarks and most of the retraining experiences were not significant. In these cases, any of the PH experiences, with or without delay, can be considered as a good option due to their low false alarm rates. Since even stable data can have drifts in the future, having a reliable drift detector is important. The use of KSWIN did not yield significant improvements as it detected too many false drifts, and considering the performance of the benchmarks, this approach is not justified.

The results for real case 3 were different. This case involved data with numerous drifts, and both KSWIN and PH proved to be effective approaches. PH successfully detected all the visual drifts but did not achieve performances as high as KSWIN, which, in the traditional approach without delay, improved upon the no retrain benchmark by 28.83% in R^2 and 18.91% in RMSE. Despite its false alarms, the increased number of retrainings in KSWIN seemed to compensate in the end. However, since more retrainings require more time, it is essential to strike a trade-off between time and prediction metrics. If stability is preferred, the PH weekly approach, which detected only 8 drifts, reduced the time required compared to the best KSWIN experience by a factor of 7. It also improved upon the no retrain model by 20.91% in R^2 and 13.30% in RMSE.

The initial approach with the incremental model ARF-Reg showed great promise. The experiments involving daily incremental and batch incremental models outperformed the retraining experiences in almost every case. The only case where this was not observed was with the batch incremental approach and data exhibiting significant changes. Although these experiences took longer compared to others, except in real case 3 where the data variability seemed to fit better with an incremental model approach.

In the first case, the time spent compared to the benchmark and the retrain experience did not prioritize the use of an incremental learning approach. For the second case, the batch incremental approach appeared to be the best option. Since it takes half the time of the daily incremental approach, the differences in error metrics were not significant, and the batch incremental approach improved upon the no retrain benchmark by 5.1% in R^2 and 10.23% in RMSE. In the third case, the time spent on the incremental experience was at the same level as the no retrain experiences. The gains of the daily incremental approach, compared to the best retrain experience, seemed to justify its use. It improved upon the best retrain

experience by 8.5% in R^2 and 10.43% in RMSE.

In summary, almost all options explored in this study proved to be effective in handling the drifts present in the data, with the daily incremental model being the best approach for real case 3. The batch incremental approach, due to taking half the time of the daily incremental approach, emerged as the optimal choice for real cases 2. And for the real case 1 the traditional PH with delay is the best option.

5.1 Future Work

In future work, further exploration of incremental modeling should be the primary focus as it has the potential to outperform traditional model retraining. Testing a drift detector in combination with the incremental model would be beneficial, where the incremental model assigns more weight to the newly observed data when a drift occurs.

Additionally, it would be valuable to analyze the performance of the CPU and GPU throughout the process to optimize the available resources and reduce the training/incremental learning time.

References

[1] A. N. Angelakis, H. S. Vuorinen, C. Nikolaidis, *et al.*, "Water quality and life expectancy: Parallel courses in time hellenic union of municipal enterprises for water supply and sewerage," 2021. DOI: 10.3390/w13060752. [Online]. Available: https://doi.org/10.3390/w13060752.

[2] M. A. Rosen, A. Niknam, H. K. Zare, H. Hosseininasab, A. Mostafaeipour, and M. Herrera, "A critical review of short-term water demand forecasting tools-what method should i use?," 2022. DOI: 10.3390/su14095412. [Online]. Available: https://doi.org/10.3390/su14095412.

[3] A. A. W. Mumbi, F. A. Li, J. A. P. Bavumiragira, and F. B. F. Fangninou, "Forecasting water consumption on transboundary water resources for water resource management using the feed-forward neural network: A case study of the nile river in egypt and kenya," DOI: 10.1071/MF21118. [Online]. Available: https://doi.org/10.1071/MF21118.

[4] X.-J. Wang, J.-Y. Zhang, S. Shamsuddin, *et al.*, "Potential impact of climate change on future water demand in yulin city, northwest china," DOI: 10.1007/s11027-013-9476-9.

[5] J. D. Rinaudo, "Long-term water demand forecasting," *Global Issues in Water Policy*, vol. 15, pp. 239–268, 2015, ISSN: 22110658. DOI: 10.1007/978-94-017-9801-3_11/COVER. [Online]. Available: https://link.springer.com/chapter/10.1007/978-94-017-9801-3_11.

[6] C. J. Hutton and Z. Kapelan, "A probabilistic methodology for quantifying, diagnosing and reducing model structural and predictive errors in short term water demand forecasting," *Environmental Modelling& Software*, vol. 66, pp. 87–97, Apr. 2015, ISSN: 1364-8152. DOI: 10.1016/J.ENVSOFT.2014.12.021.

[7] L. Chen, H. Yan, J. Yan, *et al.*, "Short-term water demand forecast based on automatic feature extraction by one-dimensional convolution," *Journal of Hydrology*, vol. 606, Mar. 2022, ISSN: 00221694. DOI: 10.1016/J.JHYDROL.2022.127440.

[8] Z. Pu, J. Yan, L. Chen, *et al.*, "A hybrid wavelet-cnn-lstm deep learning model for short-term urban water demand forecasting a r t i c l e i n f o-term water demand forecasting long-short term memory neural network convolutional neural network wavelet multi-resolution analysis data-driven models," 2022. DOI: 10.1007/s11783-023-1622-3. [Online]. Available: https://doi.org/10.1007/s11783-023-1622-3.

[9] T. Salloom, O. Kaynak, and W. He, "A novel deep neural network architecture for real-time water demand forecasting," *Journal of Hydrology*, vol. 599, Aug. 2021, ISSN: 00221694. DOI: 10.1016/J.JHYDROL.2021.126353.

[10] S. Wu, H. Han, B. Hou, and K. Diao, "Hybrid model for short-termwater demand forecasting based on error correction using chaotic time series," *Water (Switzerland)*, vol. 12, 6 Jun. 2020, ISSN: 20734441. DOI: 10.3390/W12061683.

[11] E. Pacchin, F. Gagliardi, S. Alvisi, and M. Franchini, "A comparison of short-term water demand forecasting models," *Water Resources Management*, vol. 33, pp. 1481–1497, 4 Mar. 2019, ISSN: 15731650. DOI: 10.1007/S11269-019-02213-Y.

[12] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys (CSUR)*, vol. 46, 4 Mar. 2014, ISSN: 15577341. DOI: 10.1145/2523813. [Online]. Available: https://dl.acm.org/doi/10.1145/2523813.

[13] F. Bayram, B. S. Ahmed, and A. Kassler, "From concept drift to model degradation: An overview on performance-aware drift detectors," *Knowledge-Based Systems*, vol. 245, Jun. 2022, ISSN: 09507051. DOI: 10.1016/J.KNOSYS.2022.108632.

[14] P. M. Gonçalves, S. G. D. C. Santos, R. S. Barros, and D. C. Vieira, "A comparative study on concept drift detectors," *Expert Systems with Applications*, vol. 41, pp. 8144–8156, 18 Dec. 2014, ISSN: 09574174. DOI: 10.1016/J.ESWA.2014.07.019.

[15] L. Kidane, P. Townend, T. Metsch, and E. Elmroth, "When and how to retrain machine learning-based cloud management systems," *Proceedings - 2022 IEEE 36th International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2022*, pp. 688–698, 2022. DOI: 10.1109/IPDPSW55747.2022.00120.

[16] L. Baier, N. Kühl, G. Satzger, M. Hofmann, and M. Mohr, "Handling concept drifts in regression problems – the error intersection approach," *WI2020 Zentrale Tracks*, pp. 210–224, Mar. 2020. DOI: 10.30844/WI_2020_C1-BAIER. [Online]. Available: https://library.gito.de/2021/07/wi2020-zentrale-tracks-16/.

[17] R. Cavalcante, L. Minku, and A. Oliveira, "Fedd: Feature extraction for explicit concept drift detection in time series," English, pp. 740–747, Nov. 2016. DOI: 10.1109/IJCNN.2016.7727274.

[18] T. Cabello-López, M. Cañizares-Juan, M. Carranza-García, J. Garcia-Gutiérrez, and J. C. Riquelme, "Concept drift detection to improve time series forecasting of wind energy generation," vol. 13469 LNAI, 2022. DOI: 10.1007/978-3-031-15471-3_12.

[19] L. Baier, J. Reimold, and N. Kühl, "Handling concept drift for predictions in business process mining," vol. 1, pp. 76–83, 2020. DOI: 10.1109/CBI49978.2020.00016.

[20] F. Neri, *Domain specific concept drift detectors for predicting financial time series*, 2021. arXiv: 2103.14079 [q-fin.ST].

[21] D. Alberg, M. Last, and A. Kandel, "Knowledge discovery in data streams with regression tree methods," *WIREs Data Mining and Knowledge Discovery*, vol. 2, no. 1, pp. 69–78, Oct. 2011.

[22] T. Chen, "All versus one: An empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software," *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, vol. 2019-May, pp. 157–168, May 2019, ISSN: 21567891. DOI: 10.1109/SEAMS.2019.00029.

[23] T. Chen and C. Guestrin, "Xgboost: A scalable tree boosting system," *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 13-17-August-2016, pp. 785–794, Aug. 2016. DOI: 10.1145/2939672.2939785.

[24] J. Mendes-Moreira, C. Soares, A. M. Jorge, and J. F. D. Sousa, "Ensemble approaches for regression," *ACM Computing Surveys (CSUR)*, vol. 45, 1 Dec. 2012, ISSN: 03600300. DOI: 10.1145/2379776.2379786. [Online]. Available: https://dl.acm.org/doi/10.1145/2379776.2379786.

[25] "Xgboost documentation — xgboost 1.7.5 documentation," [Online]. Available: https://xgboost.readthedocs.io/en/stable/.

[26] *What Is Concept Drift and How to Detect It – Motius*, https://motius.de/insights/what-is-concept-drift-how-to-detect-it/, Sep. 2022.

[27] "Scikit-multiflow," [Online]. Available: https://scikit-multiflow.github.io/.

[28] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," *Proceedings of the 7th SIAM International Conference on Data Mining*, pp. 443–448, 2007. DOI: 10.1137/1.9781611972771.42. [Online]. Available: http://www.lsi.upc.edu/~abifet,.

[29] A. Bifet, R. Gavaldà, G. Holmes, and B. Pfahringer, *Machine Learning for Data Streams with Practical Examples in MOA*. MIT Press, 2018, https://moa.cms.waikato.ac.nz/book/.

[30] N. I. Fisher and P. K. Sen, "Hoeffding's Independence Test," *Springer Series in Statistics*, pp. 628–629, 1994.

[31] E. S. Page, "Continuos inspection schemes," *Biometrika*, vol. 41, pp. 100–115, 1-2 Jun. 1954, ISSN: 0006-3444. DOI: 10.1093/BIOMET/41.1-2.100. [Online]. Available: https://academic.oup.com/biomet/article/41/1-2/100/456627.

[32] "Machine learning for data streams," [Online]. Available: https://www.cms.waikato.ac.nz/~abifet/book/chapter_5.html.

[33] C. Raab, M. Heusinger, and F.-M. Schleif, "Reactive soft prototype computing for concept drift streams," *Neurocomputing*, vol. 416, pp. 340–351, Jul. 2020. DOI: 10.1016/j.neucom.2019.11.111. [Online]. Available: http://arxiv.org/abs/2007.05432%20http://dx.doi.org/10.1016/j.neucom.2019.11.111.

[34] "Kolmogorov smirnov test: When and where to use it," [Online]. Available: https://arize.com/blog-course/kolmogorov-smirnov-test/.

[35] *Help Online - X-Function - kstest*, https://www.originlab.com/doc/X-Function/ref/kstest.

[36] H. M. Gomes, J. P. Barddal, L. E. B. Ferreira, and A. Bifet, "Adaptive random forests for data stream regression," in *The European Symposium on Artificial Neural Networks*, 2018.

[37] H. M. Gomes, A. Bifet, J. Read, *et al.*, *Adaptive random forests for evolving data stream classification - Machine Learning*, https://link.springer.com/article/10.1007/s10994-017-5642-8, Jun. 2017.