big data and cognitive computing

MDPI

Article

Reusing ML Models in Dynamic Data Environments: Data Similarity-Based Approach for Efficient MLOps

Eduardo Peixoto [1,†], Diogo Torres [1,†], Davide Carneiro [1,2,*,†], Bruno Silva [3,†] and Ruben Marques [3,†]

[1] Escola Superior de Tecnologia e Gestão, Instituto Politécnico do Porto, 4610-156 Felgueiras, Portugal; efrp@estg.ipp.pt (E.P.); dpto@estg.ipp.pt (D.T.)
[2] INESC TEC, R. Dr. Roberto Frias, 4200-465 Porto, Portugal
[3] Muvu Technologies, 1050-052 Lisboa, Portugal; bruno.silva@muvu.tech (B.S.); ruben.marques@muvu.tech (R.M.)
* Correspondence: davide.r.carneiro@inesctec.pt
† These authors contributed equally to this work.

Abstract: The rapid integration of Machine Learning (ML) in organizational practices has driven demand for substantial computational resources, incurring both high economic costs and environmental impact, particularly from energy consumption. This challenge is amplified in dynamic data environments, where ML models must be frequently retrained to adapt to evolving data patterns. To address this, more sustainable Machine Learning Operations (MLOps) pipelines are needed for reducing environmental impacts while maintaining model accuracy. In this paper, we propose a model reuse approach based on data similarity metrics, which allows organizations to leverage previously trained models where applicable. We introduce a tailored set of meta-features to characterize data windows, enabling efficient similarity assessment between historical and new data. The effectiveness of the proposed method is validated across multiple ML tasks using the cosine and Bray–Curtis distance functions, which evaluate both model reuse rates and the performance of reused models relative to newly trained alternatives. The results indicate that the proposed approach can reduce the frequency of model retraining by up to 70% to 90% while maintaining or even improving predictive performance, contributing to more resource-efficient and sustainable MLOps practices.

Keywords: MLOps; frugal AI; machine learning; meta-learning

check for updates

Academic Editor: Domenico Ursino

Received: 10 January 2025
Revised: 31 January 2025
Accepted: 7 February 2025
Published: 19 February 2025

Citation: Peixoto, E.; Torres, D.; Carneiro, D.; Silva, B.; Marques, R. Reusing ML Models in Dynamic Data Environments: Data Similarity-Based Approach for Efficient MLOps. *Big Data Cogn. Comput.* **2025**, *9*, 47. https://doi.org/10.3390/bdcc9020047

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In recent years, organizations across industries have increasingly adopted Machine Learning (ML) and Artificial Intelligence (AI) to optimize their operations, improve decision-making, and automate tasks [1]. This growing integration into business processes led to a dramatic rise in computational demands, driven not only by the scale of data involved but also by the increasing sophistication of the models being deployed. As ML techniques become a central pillar of modern operations, organizations are either investing in large-scale infrastructure to handle the significant computational requirements necessary for model training and deployment or relying on cloud infrastructure and services [2].

Contributing to this increase in resource demand is the fact that training ML models has become simpler and more accessible than ever before. Advances in open-source libraries, cloud computing, and powerful pre-trained models have lowered the entry barrier to ML, making it easier for organizations to quickly implement and refine ML solutions [3]. However, this accessibility comes with a trade-off: as more models are

developed and deployed, the computational footprint of ML projects continues to grow. The demand for computational resources is particularly steep and continuous in industries where data patterns are dynamic and where models must be frequently retrained to adapt to new conditions.

The environmental implications of this trend are substantial. Training state-of-the-art ML models, especially deep neural networks, requires significant energy, leading to an increase in carbon emissions. Some estimates indicate that training a single large language model can emit as much carbon dioxide as five cars being driven over their lifetimes [2]. This environmental cost is further magnified when multiple models are trained for different scenarios or updated frequently in dynamic data environments. Relying heavily on data labeling and continuous model retraining creates an unsustainable cycle of resource consumption, which, if unchecked, can have long-term negative consequences on both economic and environmental fronts [4].

Current practices in ML often default to "throwing resources at the problem", in which new models are continually trained as data change. While effective in the short term, this approach is neither resource-efficient nor sustainable, given the rising awareness of environmental impacts and the growing demand for responsible AI practices. A more sustainable Machine Learning Operations (MLOps) pipeline is essential: one that minimizes energy use and computational costs while also reducing dependency on human-intensive tasks like data labeling and managing the whole ML cycle. Reducing the need to develop new models in response to data shifts is one of the possible ways toward more efficient and sustainable ML operations.

In this paper, we introduce a method for enhancing MLOps efficiency by reusing previously trained ML models based on data similarity measures. The key idea is based on the evaluation of the similarity between new and historical datasets and their corresponding models to determine when existing models can be effectively reused. By doing so, this strategy not only conserves computational resources by reducing redundant model training but also minimizes the dependency on expensive and time-consuming data labeling efforts. Since reusing an existing model eliminates the need for obtaining new labeled data, this approach is particularly valuable in industrial settings, where labeling costs often pose a significant challenge to the adoption of AI technologies.

This study is motivated by challenges observed in the manufacturing sector, particularly within the plastics industry, where quality inspection models are typically tailored to specific product types or versions. Factors such as machine wear, environmental variations, and operator fatigue contribute to shifts in production data over time, leading to model degradation. These dynamic conditions underscore the necessity for a framework that facilitates model reuse while adapting to evolving data distributions. To address these issues, we present a structured methodology for reusing ML models, offering a resource-efficient MLOps approach that aligns with the principles of sustainable AI practices [5].

2. Background

2.1. Machine Learning Operations

MLOps refers to the practice of deploying, managing, and maintaining machine learning models in production environments. It integrates DevOps principles with machine learning to streamline the model life-cycle from development to deployment, including continuous integration and delivery (CI/CD) pipelines, automated testing, version control, monitoring, and governance [6].

MLOps aims to enhance the efficiency and reliability of machine learning workflows by ensuring consistent and reproducible results. Key approaches to achieving effective MLOps include infrastructure automation, robust data management, model versioning, and

real-time monitoring to promptly detect and resolve issues. These practices are essential for scaling machine learning solutions and maintaining their performance in real-world applications [7].

Despite its potential, many industrial machine learning projects fail to progress from proof of concept to production due to challenges in automation and operationalization [8]. The paradigm of MLOps addresses these issues by providing best practices, sets of concepts, and a development culture to automate and operate ML systems effectively [8].

The standardization and automation of the machine learning pipeline is also an opportunity to introduce sustainable practices and approaches such as those proposed in this paper, which, if seamlessly integrated into the MLOps pipeline, will become a part of the organizational culture regarding machine learning and maximize the benefits regarding sustainability.

2.2. Frugal AI

The Frugal AI vision focuses on creating AI solutions that are resource-efficient, cost-effective, and accessible. The goal is to achieve high performance while minimizing the use of data, computational power, time, or human resources. This approach is essential in industries where data are scarce, costly, or rapidly changing, making traditional AI methods less viable. Frugal AI techniques include computational optimizations such as model pruning, quantization, distillation, and leveraging pre-trained models through transfer learning [9]. Continuous learning methods such as active learning and rotation-based techniques also play a role [10].

On the data side, Frugal AI uses data reduction and generation strategies, including sampling methods, PCA, data augmentation, and synthetic data creation. These methods ensure efficient use of resources while maintaining or improving AI model performance. Additionally, Frugal AI aligns with Green AI principles by optimizing algorithms, improving hardware efficiency, and adopting sustainable data management practices to reduce environmental impacts [11].

2.3. Concept Drift

In the context of a dataset, concept drift refers to changes in the statistical properties of the target variable, which machine learning models aim to predict over time. These changes can degrade model performance as the underlying data distribution evolves, and they can occur at different speeds, severity, and distributions [12]. Concept drift poses a significant challenge in maintaining accurate and reliable models, particularly in dynamic environments. There are several types of concept drift, including sudden drift (abrupt changes in data distribution), gradual drift (slow and incremental changes over time), incremental drift (continuous but gradual changes), or recurrent drift (cyclical changes where old patterns reappear).

When addressing concept drift, organizations traditionally rely on strategies such as frequent model retraining, adaptive algorithms, and monitoring systems to promptly detect and respond to changes. The impact of concept drift on model performance has been notably observed in clinical applications such as sepsis prediction, where data drift can significantly affect the accuracy of machine learning models [13].

Due to its nature, concept drift might have a significant impact on resource consumption in machine learning, due to the need to frequently train new models, which often also imply new labeled data. Approaches such as the one proposed in this paper may minimize this impact—more so in cases of recurrent drift—as repeating patterns in the data increase the likeliness of successfully retrieving a past machine learning model.

2.4. Meta-Learning

Meta-learning, or "learning to learn", aims to improve the efficiency and performance of machine learning algorithms by leveraging knowledge from prior learning tasks. It reduces the need for extensive trial-and-error by utilizing meta-data, which includes performance metrics of algorithms or dataset characteristics. Meta-features are specific characteristics of datasets that aid in this process. Examples of meta-features include simple statistical properties, advanced statistical measures, information-theoretic metrics, and model-based features. Rivolli et al. [14] categorized meta-features into several types: simple, statistical, information-theoretic, model-based, and landmarking meta-features.

Examples of meta-learning applications include hyperparameter optimization, model selection, and algorithm recommendation. In the past, we have worked extensively on meta-learning, which culminated in the development of an algorithm recommendation system that significantly reduced the time and resources spent on selecting an appropriate algorithm and performing hyperparameter tuning [15]. In this paper, we build on this past experience.

3. Methodology

The main objective of this research is to propose an approach for reusing existing machine learning models to minimize the use of computational resources, which takes advantage of the similarity between old and new data, and the respective models. This approach was developed to be used in various scenarios, but mainly in data streaming scenarios where data evolve over time, a phenomenon known as concept drift. By reusing models that were previously trained on similar data, it is possible to significantly reduce the need to (re)train new models from scratch, thus saving time and computational power. This section begins by briefly summarizing the methodology followed, which is then further detailed below.

The methodology was developed and validated using a large group of publicly available datasets. Specifically, we use 6 datasets for selecting and evaluating a small but representative set of meta-features and 24 datasets for validating the proposed approach for model reuse, based on the selected set of meta-features. Table 1 provides a general characterization of the datasets used, which were selected to encompass a wide range of domains and data properties.

These datasets were divided into fixed-size blocks to simulate data streaming over time. For each data block, a comprehensive set of meta-features is extracted using the PyMFE library [16]. These meta-features provide a detailed characterization of each data block across multiple dimensions, such as general characteristics, statistical properties, information-theoretic metrics, model-based characteristics, and reference measures.

The size of the block was not fixed as an absolute number or a strict percentage across datasets. Instead, it was chosen heuristically to ensure a reasonable number of windows per dataset while considering dataset size variability. Specifically, for larger datasets, a larger block size was used, whereas for smaller datasets, a smaller block size was chosen to avoid excessively short windows with too little information. The goals were to maintain comparability across datasets and achieve a balance where datasets of different sizes contributed a roughly comparable number of windows, thereby making the meta-feature representations more consistent across datasets.

By characterizing data blocks with meta-features, we transform them into a suitable format for measuring similarity. Meta-feature vectors serve as a compact representation of data blocks, capturing essential information about their statistical and structural properties.

After the meta-features are extracted, the distance between data blocks is calculated using these meta-feature vectors. This distance calculation encodes the similarity between

the new data block and those in the historic model pool, and it is based on that distance that a certain model is reused or not. The rest of this section describes the two distance metrics tested (Section 3.1), the process for finding a reduced set of meta-features suitable for retrieving past models (Section 3.2), and the process through which models are reused (Section 3.3).

Table 1. General characterization of the datasets used. The first six datasets were used for selecting the meta-features, and the remaining 24 for validating the proposed approach.

Dataset	Domain	Rows	Columns
airlines	Aviation and air transport	539,383	8
covType	Forest Ecology and Remote Sensing (US Forest Service)	581,012	55
elec	Energy and electricity market	45,311	7
agr-a	Synthetic, Random Agrawal Generator (credit risk assessment)	1,000,000	10
hyper_f	Synthetic, Random Hyperplane Generator	1,000,000	11
sea_a	Synthetic, Random SEA Generator	1,000,000	4
abalone	Biology Science	4177	9
bank_10k	Financial	10,000	17
calories_9k	Nutrition	9000	8
cardio_10k	Healthcare	10,000	12
cardio_dv3	Healthcare	68,783	12
creditcard_1k	Finance	1000	30
fram_heart	Healthcare	1000	16
general_data	Human Resources	1000	35
german_credit	Finance	1000	20
Housing	Real Estate	500	12
HR_Attr	Human Resources	1000	32
pima	Healthcare	768	9
SN_Ads	Marketing	400	4
wdbc	Healthcare	500	32
wine_green	Food	4500	12
wine_red	Food	1600	12
world_food	Nutrition	300	11
wpbc	Healthcare	200	34

3.1. Distance Metrics

Calculating the distance between two vectors is a key element in this work. In this section, we describe two widely used vector distance metrics, which we compared to implement the proposed approach.

3.1.1. Bray–Curtis Dissimilarity

In this work, the Bray–Curtis dissimilarity measures the proportional differences between two data blocks. It is calculated as depicted in Equation (1).

$$Bray - Curtis\ Dissimilarity = \frac{\sum |x_i - y_i|}{\sum (x_i + y_i)} \quad (1)$$

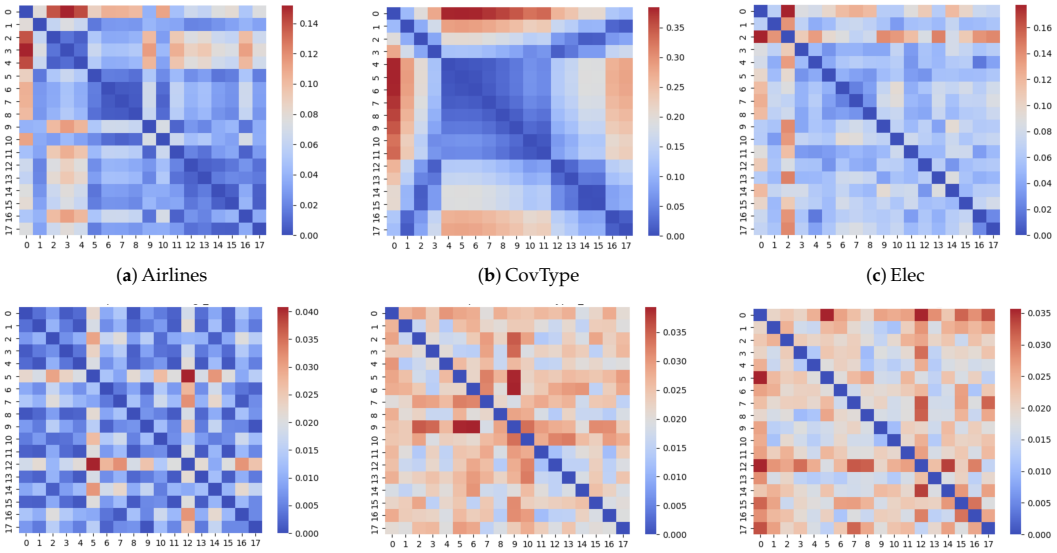
where x_i and y_i are the values of the i-th feature in two data blocks being compared.

Under the Bray–Curtis approach, the distance between two feature vectors is computed based on the direct comparison of each pair of values in the vector, as depicted in Figure 1.



Figure 1. Intuition behind the Bray–Curtis dissimilarity.

The Bray–Curtis dissimilarity thus measures the proportional differences between the data blocks based on the feature comparisons. Figure 2 depicts, through heatmaps and using the Bray–Curtis dissimilarity metric, the distance between each pair of data blocks in six datasets. The top row consists of real-world datasets, while the bottom row consists of synthetic datasets. In these images, a dark-blue color indicates a very small distance, whereas a dark red indicates a large distance. The diagonal has a zero distance as it represents the distance between each block and itself. The values of distance using the Bray–Curtis dissimilarity range between 0 (equal vectors) and 1 (completely different vectors).



(**a**) Airlines (**b**) CovType (**c**) Elec

(**d**) Agrawal-abrupt (**e**) Hyper_f (**f**) Sea_a

Figure 2. Heatmap visualization of distances between blocks of data for both real-world (upper row) and synthetic (lower row) datasets using the Bray–Curtis dissimilarity.

3.1.2. Cosine Similarity

Cosine similarity, on the other hand, measures the cosine of the angle between two vectors in a multidimensional space. It is calculated as depicted in Equation (2).

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{||\mathbf{A}||||\mathbf{B}||} \tag{2}$$

where $A \cdot B$ is the dot product of the vectors, and $||A||$ and $||B||$ are the magnitudes of the vectors.

The intuition behind the cosine similarity is depicted in Figure 3 for a bi-dimensional feature space. In this representation, each arrow represents the feature vector of one data block, both starting at the origin. The closer the vectors are to each other, the more similar they are. The same principle applies in a higher-dimensional feature space, but its visualization becomes impracticable. Cosine similarity ranges from -1 to 1, where 1 indicates identical vectors, 0 indicates orthogonality (no similarity), and -1 indicates completely opposite vectors.

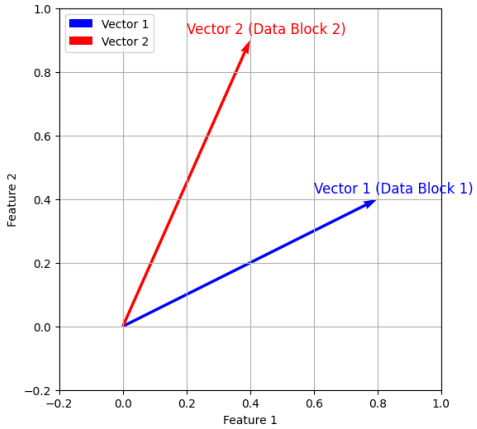
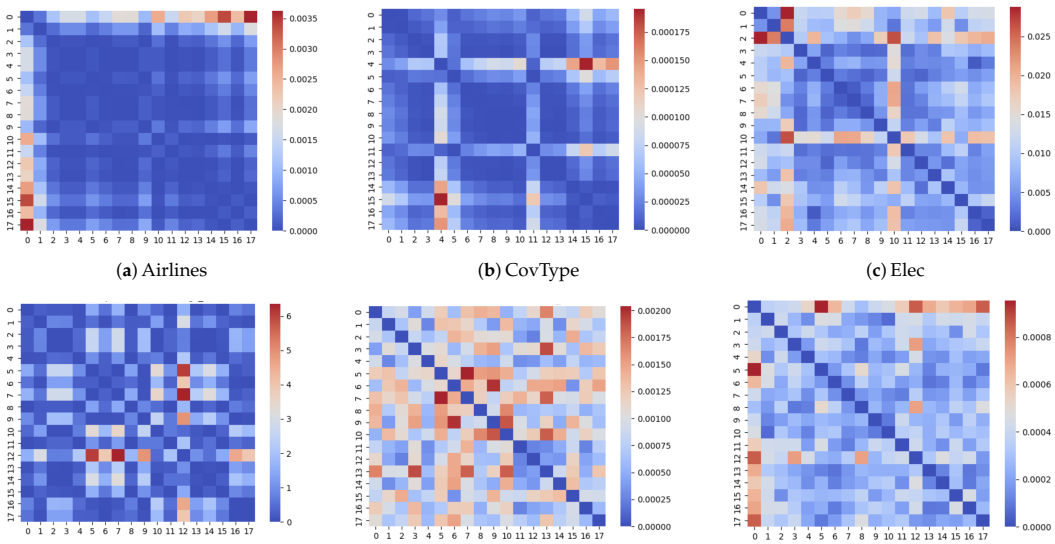


Figure 3. Intuition behind the cosine similarity.

Figure 4 visually depicts the differences between the same datasets and blocks of data used in Figure 2, but now using the cosine similarity. While the heatmaps are similar to some extent, it is also visible that the notion of distance varies according to the distance metric used. The choice might thus impact the performance of model reuse policies, and several metrics should be tested and assessed.



(**a**) Airlines (**b**) CovType (**c**) Elec

(**d**) Agrawal-abrupt (**e**) Hyper_f (**f**) Sea_a

Figure 4. Heatmap visualization of distances between blocks of data for both real-world (upper row) and synthetic (lower row) datasets using the cosine distance.

3.2. Meta-Feature Selection

Given the variety of meta-features that can be considered, one prior aspect of model reuse is the selection of the set of meta-features to use. On the one hand, some of these meta-features might not be relevant at all and introduce entropy in the process. On the other hand, we want to find the minimum set of features that allow to efficiently reuse models. This has to do with computational efficiency aspects, as the extraction of meta-features from a block of data also has a computational cost. While extracting features generally has a much smaller computational cost than training a model, a large set of meta-features might reduce the advantages of the proposed approach. Thus, the selection of an appropriate set of meta-features is a key step in this process.

The methodology employed to achieve this is depicted in Figure 5. It consists of two main phases: selection and validation.

The first phase consisted of pre-selecting a set of meta-features for posterior validation. In this process, we used several datasets (already shown in Figures 2 and 4), both real-world and synthetic, exhibiting different types of concept drift (gradual and abrupt). Real-world datasets include Airlines, CovType, and Electricity (Elec), while synthetic datasets include Agrawal-abrupt, Hyper_f, and Sea_a [17–19].

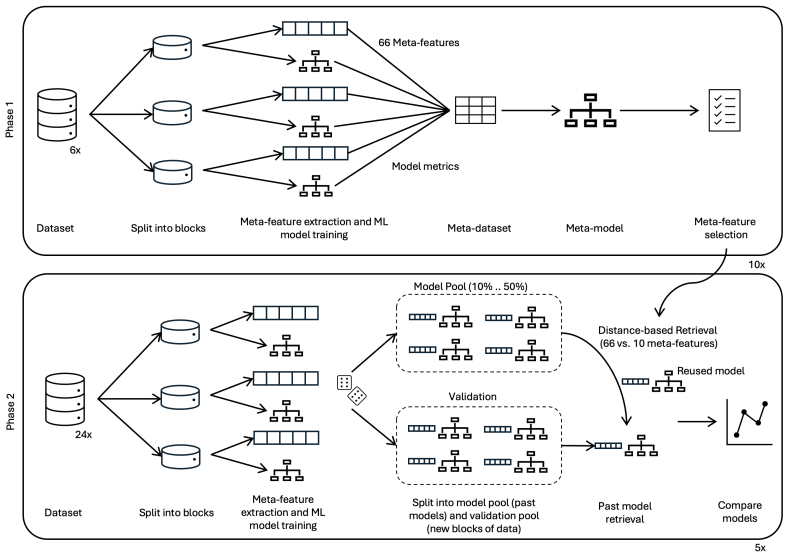


Figure 5. Methodology followed for selecting and validating a reduced set of meta-features [20].

Each dataset was segmented into an arbitrary fixed-sized set of sequential blocks based on its size. From each block, and based on our own previous work and experience [15], we extracted an initial set of 66 meta-features using the PyMFE library [16] and trained a Random Forest model with default settings using the scikit-learn library (v1.4). Performance metrics such as accuracy, precision, and recall were calculated for each model using 5-fold cross-validation, forming six meta-datasets. Each meta-dataset contained rows corresponding to the blocks of the original datasets, linking block characteristics (meta-features) to model performance.

Subsequently, Random Forest models (which we deem meta-models) were trained on each meta-dataset, with the goal of predicting accuracy based on the meta-features. The relative importance of each meta-feature in predicting model accuracy was assessed, and those with at least 1% relative importance (when compared to the most important meta-feature) were selected. This selection process was repeated across the six meta-datasets, and meta-features identified as important in at least three out of the six cases were pre-selected.

To ensure robustness and mitigate the effects of randomness, this process was repeated ten times. Each iteration resulted in a different reduced set of meta-features, ranging in number between five and ten. Finally, the ten meta-features with the highest selection frequency across all iterations were chosen. Table 2 presents these meta-features along with their frequencies.

The second phase consisted of the validation of the selected meta-features, specifically in their suitability for model reuse. In this phase, twenty-four datasets were used, which did not include the previous 6. Following a similar procedure to the first phase, each dataset was divided into fixed-sized blocks, and meta-features (both the initial set of 66 and the reduced set of 10) were extracted. One model was trained for each block of data, which was then (both the blocks of data and the models) randomly assigned to one of two groups.

The first group is the model pool, which simulates the existing past models and their corresponding blocks of training data. The second group is the validation group, which simulates future blocks of data that are arriving and the corresponding models that would

be trained on these data. The goal is to compare the would-be models with the retrieved models in order to assess the quality of the retrieval approach.

Table 2. Reduced set of 10 selected meta-features and their frequency over the 10 iterations of the process.

Meta-Feature	Description	Frequency
nre	Normalized relative entropy	10
one_nn.mean	Performance of the 1-Nearest Neighbor classifier	9
freq_class.sd	Relative frequency of each distinct class	9
linear_discr.sd	Performance of the Linear Discriminant classifier	9
elite_nn.mean	Performance of Elite Nearest Neighbor	7
naive_bayes.sd	Performance of the Naive Bayes classifier	5
linear_discr.mean	Performance of the Linear Discriminant classifier	4
best_node.mean	Performance of the best single decision tree node	4
naive_bayes.mean	Performance of the Naive Bayes classifier	2
mean.sd	Mean value of each attribute (standard deviation)	2

For each block in the validation group (representing new incoming data), the closest block from the pool was identified. We then retrieved the corresponding model from the pool and compared its predictive performance on the new block of data against the performance of its respective model, which is the model that would have been trained with that new data. We did this using both sets of meta-features (10 and 66), in order to assess both groups of meta-features in terms of number of meta-features (complexity) vs. predictive performance of the reused models.

Since the size of the pool might affect the results, as larger pools will potentially hold better candidate models, the experiments were conducted with varying pool sizes (10% to 50% of the blocks) and repeated five times to account for randomness in the distribution of data and models between the pool and validation groups. The effects of both groups of meta-features on the quality of the retrieved models, which are discussed in detail in Section 4, indicates that the smaller set of 10 meta-features performs at a comparable degree to the larger set while being computationally less intensive.

3.3. Model Reuse

This section describes the proposed methodology for reusing machine learning models based on similarity between blocks of data, measured using the selected set of meta-features. In this process of model reuse, there are three key elements, which are described next.

The pool consists of group triplets $T_i =< M_i, P_i, F_i >$ of size n in which n represents the number of data blocks received in the past via streaming, and for which machine learning models have been trained. Thus, $i \in [1, n]$ and M_i, P_i, and F_i represent, respectively, the model trained with block i, the performance metrics of M_i (e.g., accuracy, precision, recall), and the vector of meta-features of block i. There is also a temporal relationship between these triplets, so that T_i comes before T_{i+1}. While the order in which data blocks are received (and models trained) is not currently relevant, we will, in the future, experiment with more complex distance functions, which might include the freshness of the data.

The second relevant element is the distance function to be used to estimate the similarity between sets of data. As previously described, each block of data is characterized by a vector of meta-features, which is intended to represent core properties of the data. As shown in Section 3.1, namely through Figures 2 and 4, the function used influences how data similarity is measured. Some aspects to bear in mind when selecting the distance function are that each dimension in the feature vector might have very different ranges of values (with some contributing proportionally more to the total distance than others) and

that one may want to attribute different weights to each dimension, depending, for example, on the relative frequency of each one during the selection process (Table 2). Testing and evaluating different distance functions is thus paramount.

Finally, there is also the need to define a maximum distance threshold. While the approach consists of selecting the closest model, or more precisely the model whose training data are closest to the current block of data, it may happen that the closest block of data is not close enough. This maximum distance threshold thus defines the point after which we choose to train a new one based on the new data instead of reusing a model. The selection of this threshold significantly impacts the performance of the approach. Setting it to a low value will lead to a very low model reuse rate but a higher confidence in the quality of the models reused. Conversely, if this threshold is very high, the number of reused models will rise significantly, but eventually, their average predictive performance will be lower. This threshold also cannot be defined in an absolute manner as it depends on several aspects, including the distance function used or the characteristics of the data stream. Instead, it should be defined empirically.

Finally, with these three elements, the problem of model reuse can be formalized as follows: the goal is to select the model M_k from the pool such that the distance between F_{new} and F_k (the meta-features vector of model M_k) is minimized. That is,

$$k = \arg\min_{i \in [1,n]} d(F_{\text{new}}, F_i)$$

where $d(\cdot, \cdot)$ represents a distance function between meta-feature vectors and k is the index of the model in the pool with the smallest distance to F_{new}.

Thus, the selected model M_k is given by the following:

$$M_k = M_{\arg\min_{i \in [1,n]} d(F_{\text{new}}, F_i)}$$

To test the assumption that similarity metrics combined with meta-features are a good proxy to predict model performance, we devised the methodology depicted in Figure 6, which was validated with 24 datasets and which is similar to the methodology followed for selecting the reduced set of meta-features.

For each dataset, the process begins with splitting it into fixed-sized blocks, to simulate windows of streaming data. The size of each block was defined arbitrarily based on the size of the dataset. For each block, a machine learning model was trained and its performance metrics were recorded. Specifically, we used Random Forest classifiers with scikit-learn v1.4 and the default hyper-parameters and evaluated model performance using 5-fold cross-validation. For each block of data, we also extracted the meta-features vector using the PyMFE package (v0.4.3). This results in the previously described group of triplets in the shape of $< M_i, P_i, F_i >$.

It is important to note that no effort was made to optimize each individual ML model, as the primary goal of this work is to validate the proposed approach rather than to find the optimum model for each specific use case. Furthermore, we also did not do any processing of the original datasets nor any feature engineering or selection that would eventually improve the observed results. This decision was taken to maintain the proposed approach general and reproducible, and not bind it to specific actions taken on the basis of each individual dataset. Nonetheless, researchers who implement this approach in specific use cases can add typical steps to this process, such as hyper-parameter optimization, feature selection, or feature engineering.

The next step involved splitting this group of triplets into two sub-groups: the pool sub-group (representing the historical data and models that existed at time t) and the validation sub-group, which represented all the data that arrived after time t and the corresponding

models. The assignment of blocks to each of these two groups was performed randomly. We tested pools of different sizes, ranging from 10% to 50% of the total data, at 10% intervals.

For each data block in the validation group, which represents newly arrived data, we retrieved the closest block from the pool. As described previously, this was performed taking into consideration the distance between the vectors of meta-features and each of the two distance metrics used, for comparison. In essence, once the closest block of data to be reused was identified, if its distance to the new block was smaller than a predefined threshold, the corresponding model from the pool was reused. Otherwise, a new model was trained on the new block. Actually, a new model is also trained when a model is reused from the pool in order to allow a performance comparison between both. However, in a real use case, models would only be trained when no model in the pool is similar enough.

To evaluate the idea of selecting past machine learning models based on the similarity of current data to the training data, we ran a parallel experiment in which models were randomly selected from the pool. That is, we aimed to assess whether selecting models from a historic pool according to data similarity and meta-features is significantly better than selecting them randomly from the same pool.

The performance of the selected models (distance-based: Bray–Curtis and cosine similarity) on the new data block was assessed using the Root Mean Squared Error (RMSE) metric. To minimize the effect of randomness, each experiment was repeated five times for each dataset and pool size, resulting in a total of 5×5 (pool sizes) $\times$ 6 (datasets) = 150 experiments. This procedure was repeated for each of the five different distance metrics, leading to a total of 750 experiments.

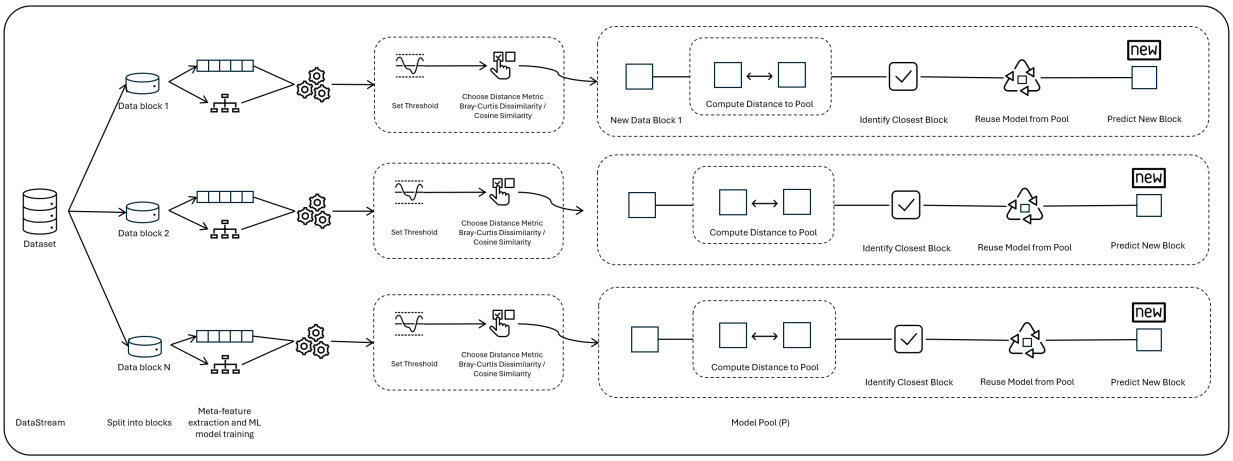


Figure 6. Process followed to validate the proposed approach.

To summarize, our methodology employs meta-feature extraction and similarity measurement to facilitate the efficient reuse of machine learning models in data streaming scenarios. By leveraging both Bray–Curtis dissimilarity or cosine similarity metrics, we aim to identify and utilize previously trained models on similar data blocks, thereby reducing the computational resources required for model (re)train.

4. Results

This section presents the outcomes of validating the reduced group of meta-features and subsequently using them to reuse past machine learning models. The section is, therefore, structured into two primary subsections: Meta-feature Selection and Model Reuse.

We consider two main metrics to assess the performance of the proposed approaches in what concerns retrieving and reusing past models: correlation of accuracies and average

distance of accuracies. The correlation of accuracies measures reliability through the degree to which the accuracies of reused models when predicting on new data align with those of the models that would be trained on the same new data. The average distance of accuracies evaluates prediction accuracy through the absolute differences in performance.

4.1. Meta-Feature Selection

This section presents the results of validating our methodology with 24 datasets, distinct from those used in selecting the reduced set of meta-features. We evaluate our approach using two key indicators: (1) the correlation between the accuracy of each model in the validation group (the new model, trained on the new data) and the accuracy of the model reused from the pool on the same new block of data; (2) the Mean Absolute Difference (MAD) between the accuracy of models in the validation group and the reused models. The correlation metric assesses how well-aligned the retrieved models are with models trained on the new data, while MAD measures the performance gap between them. Table 3 summarizes the findings for the 24 datasets with a pool size of 20%.

Table 3. Performance Comparison between the two sets of meta-features and the two distance metrics used for a pool size of 20%. In bold we highlight the best values for correlation and MAD obtained in each dataset.

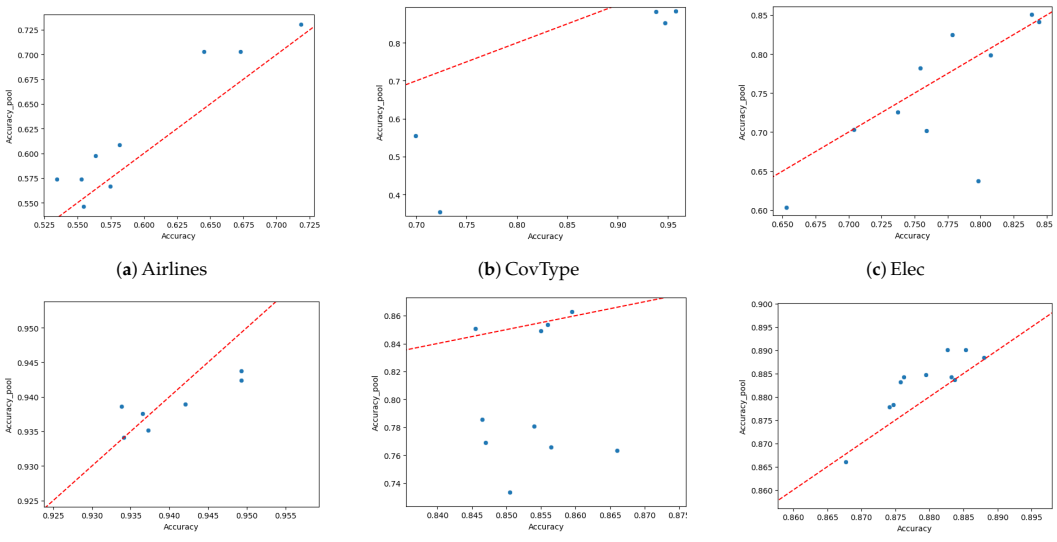
Dataset	66 Meta-Feat.		10 Meta-Feat. (BC)		10 Meta-Feat. (Cos)	
	Corr	MAD	Corr	MAD	Corr	MAD
airlines	**0.85**	**0.04**	0.80	**0.04**	0.21	0.08
elec	0.54	0.06	**0.60**	**0.01**	−0.43	0.19
general_data	0.32	0.07	**0.46**	**0.05**	0.02	0.06
german_credit	0.69	**0.05**	**0.80**	0.06	0.09	0.07
SN_Ads	0.81	0.05	**0.91**	**0.01**	0.44	0.20
agr_a	0.84	**0.01**	**0.87**	**0.01**	0.84	**0.01**
cardio_10k	**0.74**	**0.01**	0.40	0.07	0.54	0.02
pima	**0.74**	0.06	0.48	0.06	0.50	**0.05**
sea_a	0.43	**0.01**	**0.85**	0.05	0.80	**0.01**
wpbc	0.71	0.13	0.07	**0.03**	**0.92**	0.08
abalone	0.42	0.04	0.51	0.19	**0.86**	**0.02**
cardio_dv3	0.72	**0.01**	**0.73**	0.02	0.59	0.02
bank_10k	0.08	0.78	**0.85**	**0.01**	−0.36	0.15
covtype	**0.57**	0.27	−0.12	**0.08**	−0.37	0.48
wine_red	0.74	**0.06**	0.34	0.11	**0.78**	0.08
credit_1k	0.37	0.03	**0.69**	**0.01**	−0.28	0.05
wine_green	−0.31	0.08	0.02	**0.02**	**0.85**	0.05
Housing	−0.05	0.18	**0.79**	**0.01**	−0.04	0.14
fram_heart	**0.71**	0.02	0.58	0.06	0.69	**0.01**
hyper_f	0.35	**0.02**	**0.74**	**0.02**	−0.19	**0.02**
HR_Attr	0.93	**0.01**	0.86	0.02	**0.95**	**0.01**
wdbc	0.20	**0.05**	0.29	0.38	**0.35**	0.08
world_food	−0.07	**0.03**	**0.03**	0.09	−0.13	0.07
calories_9k	**0.73**	**0.01**	0.02	0.06	0.87	**0.01**

Overall, for a 20% pool size, using the selected 10 meta-features preserved a strong correlation compared to the full set of 66 meta-features across various datasets, independently of the distance function used. In some cases, such as Social_Network_Ads, bank_10000, and hyper_f, the reduced set even outperformed the full set. However, there were instances, such as cardio_vascular_10000 and pima_indians_diabetes, where using only 10 meta-features resulted in poorer correlation and/or increased error. These results suggest that a case-by-case selection of meta-features may be necessary for different use cases, as outlined

in our methodology. For a 20% pool size, the reduction to 10 meta-features outperformed the full set of 66 in terms of correlation for 62.5% of the datasets and achieved a smaller MAD in 37.5% of the datasets.

It is also useful to visually depict the results of the proposed approach. Figure 7 shows the accuracy of the specifically trained models against the retrieved model, using the reduced set of meta-features and the Bray–Curtis distance, for six datasets. It can be seen how the retrieved models are generally well aligned with the actual models that would be trained on the new block of data. Figure 8 is similar, but the retrieval was performed using cosine similarity rather than the Bray–Curtis distance. It can be seen that there are datasets for which it is generally harder to reuse models (e.g., CovType) and also that Bray–Curtis seems to be a better distance metric to reuse models than cosine similarity, as corroborated by Table 4.

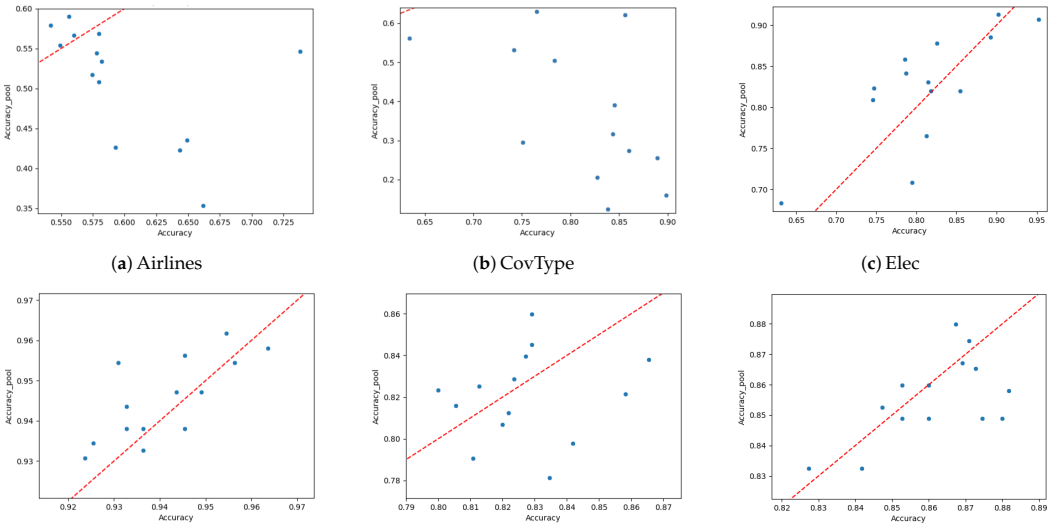
Table 4 summarizes the outcomes for varying pool percentages from 10% to 50% using different meta-feature sets. On average, the identified group 10 meta-features achieves better correlation between validation and reused model accuracies in 2 out of 5 pool sizes compared to using all 66 features. It also yields a lower average MSE in 4 out of 5 cases. The analysis of datasets on an individual basis shows that the group of 10 features outperforms the full set in terms of correlation for 54–63% of the problems, depending on the pool size. A similar 38–46% advantage is observed for the average MSE. This confirms a reduced meta-feature set can effectively characterize datasets for the purpose of model retrieval while lowering computational overhead. When comparing both distance metrics investigated, it is clear from the table that across all twenty-four datasets and five pool sizes tested, Bray–Curtis generally holds better results than cosine similarity.



(**a**) Airlines (**b**) CovType (**c**) Elec

(**d**) Agrawal-abrupt (**e**) Hyper_f (**f**) Sea_a

Figure 7. Comparison of the accuracy of a specifically trained model vs. the accuracy of a reused model (using the Bray–Curtis distance) for three real (first row) and synthetic (second row) datasets [21].



(**a**) Airlines (**b**) CovType (**c**) Elec

(**d**) Agrawal-abrupt (**e**) Hyper_f (**f**) Sea_a

Figure 8. Comparison of the accuracy of a specifically trained model vs. the accuracy of a reused model (using the cosine distance) for three real (first row) and synthetic (second row) datasets.

Table 4. Summary of the results for different pool sizes when comparing 66 vs. 10 meta-features. In bold we highlight the best values for correlation and MAD obtained for each pool size.

Pool Size	66 Meta-Feat.			10 Meta-Feat. (BC)			10 Meta-Feat. (Cos)		
	Corr	MAD	SD	Corr	MAD	SD	Corr	MAD	SD
10%	0.45	0.07	0.04	0.37	0.07	0.05	**0.47**	0.08	0.06
20%	0.50	0.09	0.05	**0.52**	**0.06**	0.04	0.35	0.08	0.07
30%	**0.53**	0.06	0.06	0.42	**0.06**	0.05	0.43	0.10	0.09
40%	0.45	0.06	0.07	**0.47**	**0.06**	0.06	0.36	0.11	0.08
50%	**0.58**	0.09	0.08	0.51	**0.06**	0.07	0.40	0.12	0.09

4.2. Model Reuse

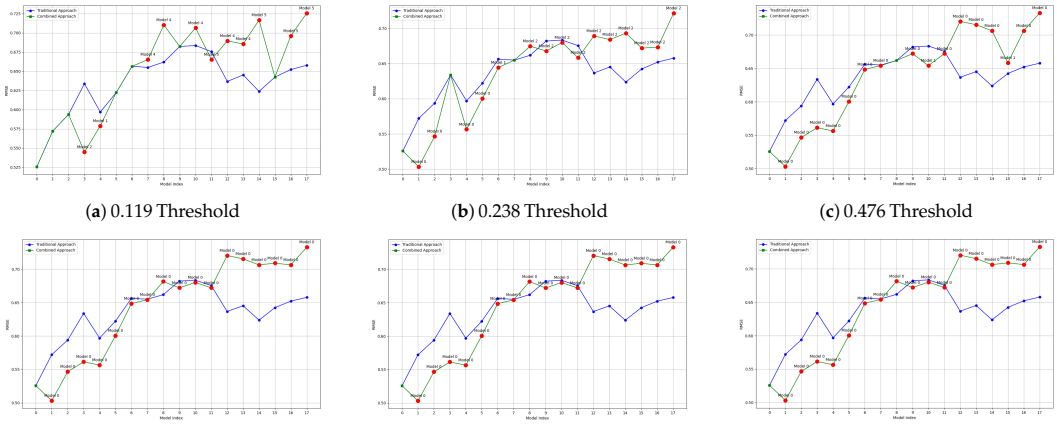
While the previous section analyzed the performance of both distance metrics investigated in reusing models, this section adopts a more realistic approach to the problem by looking at it as a time-series one. That is, for validating the reduced set of meta-features, we selected data and models at random to attribute them to both pools (historic and validation), we now take the streaming data in the order it was produced, as would have happened in reality. At a given point in time t, we thus assume that every model trained until $t-1$ is available to be reused. In each new timestamp t in which a new window of data is available, we decide whether to reuse a model or to train a new one. The criteria for doing so has to do with the maximum distance threshold and the distance between each past block of data and the current one. The only exception to this is the first iteration, in which there are no past models, so a new model is always trained.

In this case, we look at two main metrics. First, we want to calculate the percentage of models that were effectively reused, as this provides insights into the computational resources that are saved when not training models. Second, we look at the quality of the retrieved models when predicting on the new data, when compared with the models that would actually be trained on the new data. This means that, for validation purposes, we train one model for each block of data. However, in a real use case, there would be no model being trained when an old one is retrieved, which would result in the saving of computational resources, time, and eventually human resources as well if data labeling was necessary.

The effect of the proposed methodology on model reuse can be intuitively perceived in Figures 9–12. In these figures, each line represents the performance of models over time for a given machine learning problem. The blue line can be seen as the baseline, and it shows the RMSE over time when specific models are trained for each new block of data. The green line, on the other hand, shows the RMSE over time of the proposed approach, in which some models are reused. These lines overlap when there is no good candidate model to be reused (according to the threshold defined). When a model is reused, its RMSE measured on the new data is displayed, as well as its id. This is shown as a label above the red dot. That is, a label for Model n indicates that, at that moment in the stream, model n was reused rather than a new model being trained. Model indexes start at 0.

These figures also highlight the effect of using different thresholds: the higher the threshold, the more models are reused, as expected. However, the quality of the models may be lower. While the results are more precisely detailed in the tables below, it can be seen that the proposed approach results in models that are generally aligned with the traditional approach, in which one model is trained for each new block of data.

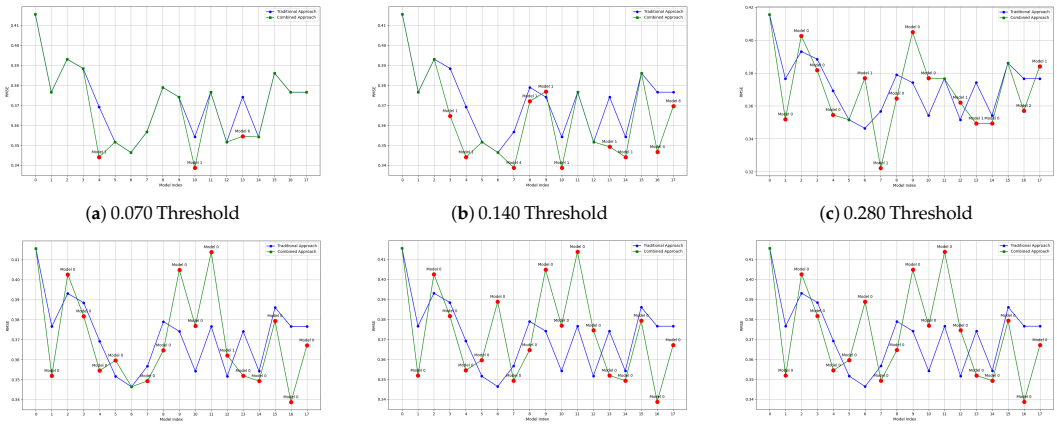
More specifically, Figures 9 and 10 show the performance of the proposed approach for two datasets (Airlines and Sea) for varying threshold levels, defined empirically, and using the cosine distance to retrieve models.



(**a**) 0.119 Threshold (**b**) 0.238 Threshold (**c**) 0.476 Threshold

(**d**) 0.714 Threshold (**e**) 0.952 Threshold (**f**) 1.190 Threshold

Figure 9. Plot of average RMSE values per threshold and model index using the Cosine method (airlines dataset [22]).

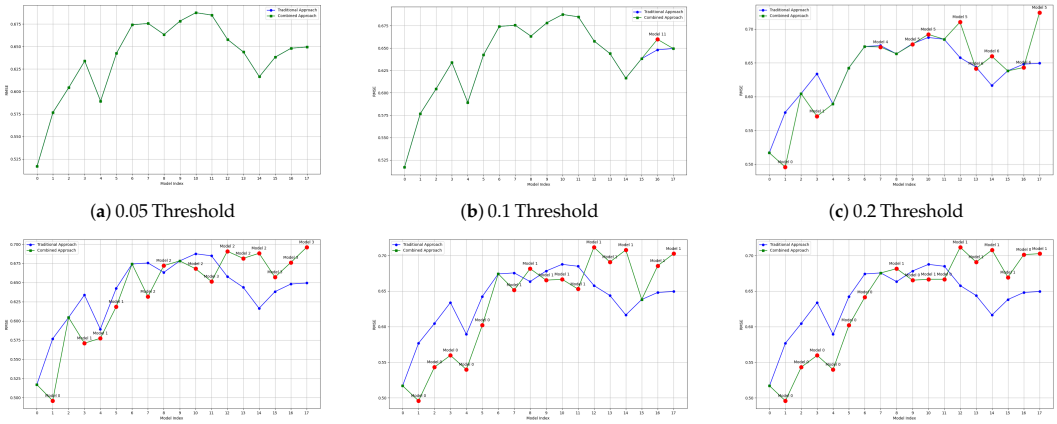


(**a**) 0.070 Threshold (**b**) 0.140 Threshold (**c**) 0.280 Threshold

(**d**) 0.420 Threshold (**e**) 0.560 Threshold (**f**) 0.700 Threshold

Figure 10. Plot of average RMSE values per threshold and model index using Cosine method (sea dataset [22]).

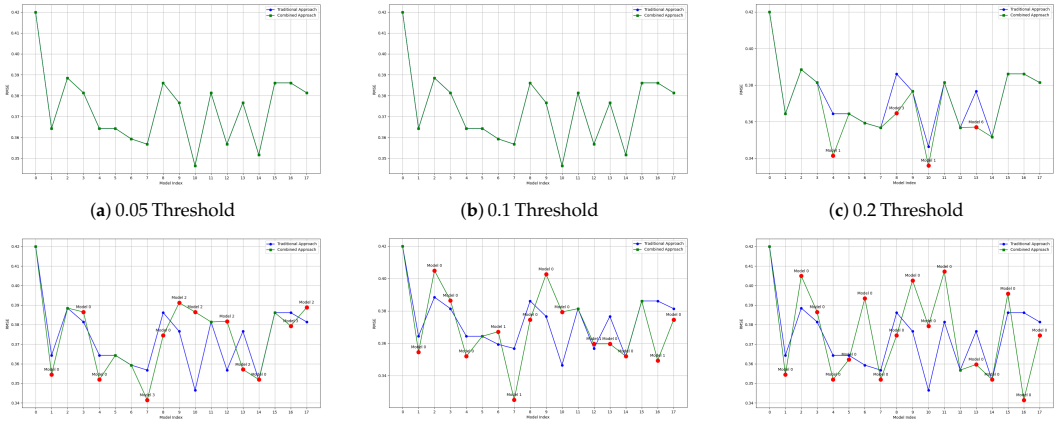
Figures 11 and 12 show the results for the same datasets, but using the Bray–Curtis distance to determine which models to reuse. As previously mentioned, it can be seen how more models are reused when a higher threshold is used. It can also be seen, independently of the distance metric used, that the models reused are, in some cases, better than the actual models and, in other cases, worse. More detail is provided in the tables below.



(**a**) 0.05 Threshold (**b**) 0.1 Threshold (**c**) 0.2 Threshold

(**d**) 0.3 Threshold (**e**) 0.4 Threshold (**f**) 0.5 Threshold

Figure 11. Plot of average RMSE values per threshold and model index using the Bray–Curtis method (airlines dataset [22]).



(**a**) 0.05 Threshold (**b**) 0.1 Threshold (**c**) 0.2 Threshold

(**d**) 0.3 Threshold (**e**) 0.4 Threshold (**f**) 0.5 Threshold

Figure 12. Plot of average RMSE values per threshold and model index using the Bray–Curtis method (sea dataset [22]).

Tables 5–8 summarize the results of this evaluation for the two datasets that were also depicted in Figures 9–12. The goals of this evaluation are twofold. On the one hand, we want to analyze the quality of the retrieved models against the quality of the actual models that would have been trained in each step if a traditional approach were used. On the other hand, we also want to assess how different thresholds impact the amount of models reused, that is, the amount of models that were not trained, and how they impact the average quality of the retrieved models. Thus, in each table, we show the following: the average and standard deviation of the RMSE of the models trained using a traditional approach (in which one new model is trained for each window of data); the average and standard deviation of the RMSE of the proposed approach at different thresholds; the average difference between the RMSE values; and the percentage of models reused.

Tables 5 and 6 summarize the results for different thresholds for the airlines dataset, using the cosine and Bray–Curtis distance metrics, respectively. Tables 7 and 8 do the same for the sea dataset.

Results show that out of 24 cases tested (considering both distance metrics), the proposed approach outperforms the traditional one in 4 (≈16%), has the same performance in 19 (≈80%), and performs worse in 1 (≈4%). Thus, it can be stated that in terms of predictive performance, both approaches are equivalent.

However, when the rate of reused models is considered, the distance metric used has an influence. When the cosine distance is used to retrieve models, the rates of reused models range between 16.67% and 94.44%, with an average across all thresholds of 78.7%. In comparison, when the Bray–Curtis distance metric is used, models are only reused in 47.2% of the cases, ranging between 0% and 88.89%, depending on the threshold. Thus, it can be concluded that the cosine distance allows for superior reuse of models while maintaining their predictive performance.

Table 5. Average RMSE values per threshold using Cosine method (airlines dataset [22]). Bold values indicate cases in which the proposed approach outperformed the traditional one.

Threshold	Traditional		Proposed		Avg. Difference	Models Reused (%)
	Avg	Sd	Avg	Sd		
0.12	0.64	0.04	**0.63**	0.06	0.01	61.11
0.24			**0.63**	0.06	0.01	83.33
0.48			0.64	0.07	0.00	88.89
0.71			0.64	0.08	0.00	94.44
0.95			0.64	0.08	0.00	94.44
1.19			0.64	0.08	0.00	94.44

Table 6. Average RMSE values per threshold using the Bray–Curtis method (airlines dataset [22]) Bold values indicate cases in which the proposed approach outperformed the traditional one.

Threshold	Traditional		Proposed		Avg. Difference	Models Reused (%)
	Avg	Sd	Avg	Sd		
0.05	0.64	0.04	0.64	0.06	0.00	0.00
0.10			0.64	0.07	0.00	5.55
0.20			0.64	0.07	0.00	55.55
0.30			0.64	0.07	0.00	77.78
0.40			**0.63**	0.07	0.01	83.33
0.50			0.64	0.07	0.00	88.89

Table 7. Average RMSE values per threshold using Cosine method (sea dataset [22]). Bold values indicate cases in which the proposed approach outperformed the traditional one.

Threshold	Traditional		Proposed		Avg. Difference	Models Reused (%)
	Avg	Sd	Avg	Sd		
0.07	0.37	0.02	0.37	0.02	0.00	16.67
0.14			**0.36**	0.02	0.01	55.55
0.28			0.37	0.02	0.00	77.78
0.42			0.37	0.02	0.00	88.89
0.56			0.37	0.02	0.00	94.44
0.70			0.37	0.02	0.00	94.44

Table 8. Average RMSE values per threshold using the Bray–Curtis method (sea dataset [22]). Bold values indicate cases in which the proposed approach outperformed the traditional one.

Threshold	Traditional		Proposed		Avg. Difference	Models Reused (%)
	Avg	Sd	Avg	Sd		
0.05	0.37	0.02	**0.37**	0.02	0.00	0.00
0.10			0.37	0.02	0.00	0.00
0.20			0.37	0.02	0.00	22.22
0.30			0.37	0.02	0.00	66.67
0.40			0.37	0.02	0.00	77.78
0.50			0.38	0.02	0.00	88.89

These results show that the proposed approach for reusing models in data streaming scenarios is promising, as it almost always matches or outperforms traditional approaches while significantly reducing the number of models trained. This validates the hypothesis that data similarity metrics can be a good predictor of model performance and a computationally inexpensive way of selecting viable past models for new problems. Nonetheless, as has been shown, it must be kept in mind that the results depend largely on the configuration used, on the distance metric and thresholds, and especially on a balance between the number of reused models and their computational performance. The ideal values for achieving this balance must be, at first, searched through empirical processes, such as the one described here, and depend also on the constraints of the organization (e.g., the available computational resources or budget).

5. Conclusions, Limitations, and Future Work

In this paper, we addressed the increasingly relevant need for sustainable and resource-efficient MLOps practices in light of the growing computational and environmental demands associated with machine learning. By proposing a novel approach that leverages data similarity metrics to enable the reuse of previously trained models on the fly, we take a step toward minimizing the need for training new models and, consequently, minimizing the need for labeling data by organizations, which often represents an adoption barrier or a cutoff in model quality.

With the reuse of models, machine learning pipelines also become more autonomous and less dependent on human activities and expertise as each model being reused is a model that did not need to be trained and whose data did not need to be labeled.

All of this relies on using meta-features that accurately describe the properties of data and their relationship with the downstream model quality. As such, the process of meta-feature extraction also has a computational cost that cannot be discarded. Since one of the goals of this work is to improve sustainability indicators, we also set out to identify a reduced set of meta-features that could accurately achieve this. We have shown that a reduction from the original 66 to a set of 10 meta-features maintained or slightly improved the performance of the retrieved models, which showed that this reduced set of meta-features was ideal.

Lastly, another relevant factor that was studied is the function used to compute the distance between meta-feature vectors, i.e., between windows of data. We have compared the performance of the cosine and Bray–Curtis distance and concluded that both are equivalent in terms of the average quality of the models retrieved, but the cosine leads to higher reuse rates.

We tested and validated this work on a comprehensive set of streaming and batch datasets, some of them with concept drift, both natural and synthetic. In the vast majority of cases, the proposed approach matched (≈16%) or outperformed (≈80%) a traditional

approach in which a new model is trained for each window of data. Nevertheless, it must also be stated that achieving better results than traditional approaches was never the main goal. The main goal was to achieve equal or better results while minimizing the number of machine learning models trained. And we consider that goal achieved since, in most of the cases, the number of models trained can be significantly reduced, often by 70% to 90%, while maintaining predictive performance. These results thus show that data distance metrics can indeed be used as a proxy to reuse previously trained models.

Despite the fact that this approach was extensively validated, there are still a few limitations. First and foremost, it is clear that this approach must be tailored to each specific use case, especially in what concerns the definition of the threshold to retrieve past models. This threshold is paramount as it defines the percentage of models that are reused, as well as their expected quality, since less similar data blocks will, in principle, hold less suited models. Nonetheless, the empiric process described here, in which several thresholds are objectively evaluated, can be followed in other use cases to identify the best threshold to use.

We also decided empirically on the size of each data block when simulating data streaming and deciding when to run the process of retrieving a past model or training a new one. We acknowledge that the choice of block size could influence both the computed meta-features and the retrieval performance. A very small block size might introduce high variance in meta-features, making them noisier, while a very large block size might smooth out meaningful variations within the dataset. Although we did not conduct a full sensitivity analysis, we expect that moderate changes in block size would not drastically affect retrieval performance as long as the chosen window captures sufficient statistical information. That said, we highlight that exploring the impact of block size selection more formally for each use case, namely through a sensitivity analysis, would likely further improve these results.

Another limitation not yet addressed is that of the size of the pool. The solution described in this paper assumes that the pool can grow forever. This is not realistic as there are resource limitations for the amount of models that can be stored. Moreover, as the model pool grows, so too does the computational complexity of choosing a model. Thus, in future work, we will implement this process for pools of limited size, in which the models of the pool are managed according to their age or their inter-similarity.

Moreover, we are also working on migrating from a pool to an ensemble of models. That is, instead of maintaining a pool of models but only using the most suitable one at each moment, we will use all the models in the pool in the form of an ensemble. This significantly improves the predictive quality of the models but may increase computational complexity. In this version, the size of the pool/ensemble becomes even more important, and it will be up to the organization to decide between one approach or the other, depending on resource and cost constraints.

The final limitation that is currently being addressed is that of deciding when to run the process of selecting a new model. In this work, we trigger the process at regular intervals for each window of data. However, there are cases in which the statistical patterns of the data being generated did not change significantly, where there is no need to change the model. We are currently incorporating outlier and concept drift detection so that the process is only triggered when significant changes to the data are detected, thus further optimizing resource consumption.

All in all, we believe that the proposed approach is a promising one for minimizing resource consumption and human intervention in modern MLOps, which makes them more efficient and sustainable.

Author Contributions: Conceptualization, D.C. and B.S.; methodology, D.C.; software, E.P. and D.T.; validation, D.C. and R.M.; data curation, E.P. and D.T.; writing—original draft preparation, D.C. and R.M.; writing—review and editing, B.S. and R.M.; visualization, E.P. and D.T.; supervision, D.C.; funding acquisition, D.C. and B.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been supported by the European Union under the Next Generation EU, through a grant of the Portuguese Republic's Recovery and Resilience Plan (PRR) Partnership Agreement, within the scope of the project PRODUTECH R3—"Agenda Mobilizadora da Fileira das Tecnologias de Produção para a Reindustrialização", Total project investment: 166,988,013.71 Euros; Total Grant: 97,111,730.27 Euros.

Data Availability Statement: No data was generated during this work.

Conflicts of Interest: Authors Bruno Silva and Ruben Marques were employed by Muvu Technologies. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Jordan, M.I.; Mitchell, T.M. Machine learning: Trends, perspectives, and prospects. *Science* **2015**, *349*, 255–260. [CrossRef] [PubMed]
2. Strubell, E.; Ganesh, A.; McCallum, A. Energy and policy considerations for modern deep learning research. In Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI-20), New York, NY, USA, 7–12 February 2020; Volume 34, pp. 13693–13696.
3. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
4. Henderson, P.; Hu, J.; Romoff, J.; Brunskill, E.; Jurafsky, D.; Pineau, J. Towards the systematic reporting of the energy and carbon footprints of machine learning. *J. Mach. Learn. Res.* **2020**, *21*, 1–43.
5. Govindan, K. How artificial intelligence drives sustainable frugal innovation: A multitheoretical perspective. *IEEE Trans. Eng. Manag.* **2022**, *71*, 638–655. [CrossRef]
6. Gupta, P.; Bagchi, A. MLOps: Machine Learning Operations. In *Essentials of Python for Artificial Intelligence and Machine Learning*; Springer Nature: Cham, Switzerland, 2024; pp. 489–518. [CrossRef]
7. Ruf, P.; Madan, M.; Reich, C.; Ould-Abdeslam, D. Demystifying mlops and presenting a recipe for the selection of open-source tools. *Appl. Sci.* **2021**, *11*, 8861. [CrossRef]
8. Kreuzberger, D.; Kühl, N.; Hirschl, S. Machine Learning Operations (MLOps): Overview, Definition, and Architecture. *IEEE Access* **2023**, *11*, 31866–31879. [CrossRef]
9. Kim, J.; Chang, S.; Kwak, N. PQK: Model compression via pruning, quantization, and knowledge distillation. *arXiv* **2021**, arXiv:2106.14681.
10. Liu, B. Lifelong machine learning: A paradigm for continuous learning. *Front. Comput. Sci.* **2017**, *11*, 359–361. [CrossRef]
11. Bol'on-Canedo, V.; Mor'an-Fern'andez, L.; Cancela, B.; Alonso-Betanzos, A. A review of green artificial intelligence: Towards a more sustainable future. *Neurocomputing* **2024**, *599*, 128096. [CrossRef]
12. Suárez-Cetrulo, A.L.; Quintana, D.; Cervantes, A. A survey on machine learning for recurring concept drifting data streams. *Expert Syst. Appl.* **2023**, *213*, 118934. [CrossRef]
13. Rahmani, K.; Thapa, R.; Tsou, P.; Casie Chetty, S.; Barnes, G.; Lam, C.; Foon Tso, C. Assessing the effects of data drift on the performance of machine learning models used in clinical sepsis prediction. *Int. J. Med. Inform.* **2023**, *173*, 104930. [CrossRef]
14. Rivolli, A.; Garcia, L.P.; Soares, C.; Vanschoren, J.; de Carvalho, A.C. Characterizing classification datasets: A study of meta-features for meta-learning. *arXiv* **2018**, arXiv:1808.10406.
15. Palumbo, G.; Carneiro, D.; Guimares, M.; Alves, V.; Novais, P. Algorithm recommendation and performance prediction using meta-learning. *Int. J. Neural Syst.* **2023**, *33*, 2350011. [CrossRef] [PubMed]
16. Alcobaça, E.; Siqueira, F.; Rivolli, A.; Garcia, L.P.F.; Oliva, J.T.; de Carvalho, A.C.P.L.F. MFE: Towards reproducible meta-feature extraction. *J. Mach. Learn. Res.* **2020**, *21*, 1–5.
17. Gomes, H.M.; Bifet, A.; Read, J.; Barddal, J.P.; Enembreck, F.; Pfharinger, B.; Holmes, G.; Abdessalem, T. Adaptive random forests for evolving data stream classification. *Mach. Learn.* **2017**, *106*, 1469–1495. [CrossRef]
18. Blackard, J.A.; Dean, D.J. Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Comput. Electron. Agric.* **2000**, *24*, 131–151. [CrossRef]
19. Harries, M.; Wales, N.S. *Splice-2 Comparative Evaluation: Electricity Pricing*; The University of New South Wales: Sydney, Australia, 1999.

20. Peixoto, E.; Torres, D.; Carneiro, D.; Silva, B.; Novais, P. Efficient MLOps: Meta-learning meets Frugal AI. In Proceedings of the 2nd European Symposium on Artificial Intelligence in Manufacturing—ESAIM, Athens, Greece, 16 October 2024.
21. Peixoto, E.; Carneiro, D.; Torres, D.; Silva, B.; Novais, P. Reusing past Machine Learning models based on data similarity metrics. In Proceedings of the 15th International Symposium on Ambient Intelligence, Salamanca, Spain, 26–28 June 2024.
22. Losing, V. Drift Datasets. Available online: https://github.com/vlosing/driftDatasets/ (accessed on 9 January 2025).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.