Is My Model Up-to-date? Detecting CoViD-19 Variants by Machine Learning*

Oguzhan Avci[1,†], Giuseppe Pozzi[1,**,†]

[1]DEIB, Politecnico di Milano, P.za L. da Vinci 32, I-20133, Milano, Italy

Abstract

Machine learning extracts models from huge quantities of data. Models trained and validated over past data can be deployed in making forecasts as well as in classifying new incoming data. The real world which generates data may change over time, making the deployed model an obsolete one. To preserve the quality of the currently deployed model, continuous machine learning is required. Our approach retrospectively evaluates in an online fashion the behaviour of the currently deployed model. A drift detector detects any performance slump, and, in case, can replace the previous model with an up-to-date one.

The approach experiments on a dataset of 8642 hematochemical examinations from hospitalized patients gathered over 6 months: the outcome of the model predicts the RT-PCR test result about CoViD-19. The method reached an area under the curve (AUC) of 0.794 , 6% better than offline and 5% better than standard online-binary classification techniques.

Keywords

Machine learning, adaptive modelling, concept drift, model update, CoViD-19

1. Introduction

Artificial Intelligence (AI) initially aimed at simulating and replicating the human way of thinking.AI evolved very rapidly, and nowadays aims to “reason about huge quantities of data” [1], rising the concept of Machine Learning (ML). Reasoning refers to the capacity of extracting knowledge; learning refers to the capacity of acquiring new knowledge from facts, automatically deriving a thesaurus of knowledge which makes the system capable of autonomous behavior in the real world.

ML processes huge quantities of stored data to derive a model: the model enables us to make forecasts about values for future data. Forecasts are the more precise and the more accurate ones the more the model adheres to the real world. However, the real world may experience changes, thus drifting away from the original behavior, i.e., the one over which the model was initially defined. As a consequence, the performance of the model slumps. Two urgent needs arise: a) detect the performance slump of the model; b) decide when new training on more recent data is needed, to re-couple the model with the real world.

The current paper aims at identifying a criterium to detect the need for re-coupling the model with the real world. As an application scenario, the paper considers data from blood examinations from [2]: data are then processed to build a predictive model, which classifies if the patient is affected by CoViD-19 or not.

The paper is structured as follows: Section 2 describes the state of the art in ML; Section 3 focuses on the approach we propose here; Section 4 describes some details about the prototype implementation; Section 5 reports about the on-the-field deployment of the model; finally, Section 6 draws some concluding remarks.

2. Related Work

This section describes some background issues and the state of the art on modelling and machine learning.

2.1. Background

Given a huge quantity of data (the *ground truth*), ML enable us to build up a model which best fits the real world data come from. The bigger the data are, the more precise the fitting. The ground truth is split into two parts: one part is used to train the model; the other part is used to validate the model, i.e. to assess the quality of the identified model in terms of its performances in predicting or classifying newly acquired data (*operating data* – not those used to train the model).

A properly trained model can behave well in the real scenario for some time; however, the performances degrade over time [3], and the model is no longer adequate. The performance slump has to be properly detected and managed: several reasons may trigger such a situation, including changes that occurred in the real world *after* the model has been initially trained.

Models must then be retrained (i.e., trained again), either automatically or manually, to face changes in op-

Published in the Workshop Proceedings of the EDBT/ICDT 2023 Joint Conference (March 28-March 31, 2023), Ioannina, Greece

*Corresponding author.

†These authors contributed equally.

oguzhan.avci@mail.polimi.it (O. Avci); giuseppe.pozzi@polimi.it (G. Pozzi)

https://www.deib.polimi.it/pozzi (G. Pozzi)

– (O. Avci); 0000-0002-2828-862X (G. Pozzi)

erating data. Manual retraining is functional but expensive and time-consuming. The current enterprise approach offers MLOPs (Machine Learning OPerations) as a potential solution for automated retraining. However, minimal effort was devoted to developing adaptive machine-learning frameworks that can continuously update models. Operating data must be integrated into models quickly to enable reliable predictions. Online machine learning allows the ongoing recalibration of ML models in fast-changing circumstances.

Deploying a two-fold system could benefit. One part of the system, once suitably trained, takes care of reading new incoming data and of making suitable forecasts or suitable classifications over the most recent data. The second part of the system guarantees *experiment tracking* and *automated deployment*: this part of the system automatically warns in case of a performance slump, retrains the model, and deploys the most recent model.

2.2. Related Work

Major relevant literature refers to auto-adaptive machine learning and concept drift.

Auto-adaptive machine learning [4] is a reference architecture where ML models can be deployed and do not need to be manually maintained. While several products are in this direction, the motivation of the major ones of these systems is smoothing the path to production. The focus is on the supervision of the model once deployment has ensued. Fast deployment without constant vigilance would worsen technical debt [5], while the problem of optimal continual learning is NP-hard [6].

A first try is from Widmer et al. [7] with the FLORA framework. Major features are: holding only one window of currently trusted samples and beliefs; keeping concept descriptions and re-using them when a previous context re-appears; managing both functions by a heuristic that constantly monitors the system's behaviour. The main problem is that the window adjustment heuristic is conditional on parameters. Although the parameter settings chosen earlier yielded somewhat robust behaviour in most artificial environments, this is not sufficient.

The approach by Gomes at al. [8] is similar to the one above. It uses an online learning system that controls available context information to enhance an existing drift detection technique in circumstances where *previously observed* concepts recur. It constructs a context-concepts history used to detect and adjust to drift.

Another similar approach is also proposed by Nascimento et al. [9]. They suggest a control module that configures ML models, monitors the context modifications, and holds a history of the trained ML-based models that deliver the best result for one of the operational contexts. When the context changes, the rate of the outcomes achieved by the ML model currently in use may drop. Therefore, if context transformations are known, the controller will retreat to an earlier composition instead of retraining itself. Otherwise, it will train new ML models. Consequentially, if a context is not taken into consideration explicitly, substantial bias can be introduced.

A completely new technique is proposed by Bach et al. [10], somehow similar to the one later on used in the current paper. The novel vision is to use the interaction between two learners, a stable online learner with a reactive one, and their contrasts in accuracy to manage concept drift. The stable learner beats the reactive learner when acquiring a new concept, but the reactive learner surpasses the stable learner in the time behind which the concept changes.

Concept *drift* in ML refers to changes in the problem's relationships between input and output data gradually [11]. With more focus on the issues relevant to the current paper, the detection of concept drifts is to be performed in the data streams over time. While concept drift detection can be performed by some statistical process control as DDM [12], or temporal window [13] distribution as ADWIN [14], the remediation of concept drift can be collected by the taxonomy presented by Gama at al. [15]: retrain, i.e., use more recent data; ensembles, i.e., keep a collection of learners and merge their decisions to take a general conclusion.

Baier presents a switching adaptation strategy [16] for dealing with concept drift in real-world datasets. The initial model is updated incrementally with the most recent samples for a specific time to adjust to the most current concepts. However, following a particular time span, updates will not be sufficient to adapt the model to the most recent data shifts since the concept changed, which means that the current model is obsolete. Thus, this needs the retraining of a new model.

In the current paper, we follow the approach of [16], where instead of retraining a new model when the actual model becomes outdated, we replace the current model with the one that is updated recurrently in the background with more recent data.

3. Proposed Approach

One essential part of concept drift handling is not just the detection of drifts but even how to re-adjust the underlying ML model. The model might be commonly adapted whenever new labelled data are obtained. *Incremental practices* update the existing model regarding the most recent data samples [16]. Relevant features of the underlying model are continuously adapted, e.g., the weights of a neural network. Adaptions can either be founded on triggers, such as precise concept drift detectors, or can be taken out regularly on an evolving story. Triggered transformation methods are defined as knowledgeable or operational strategies, whereas growing strategies are viewed as random or inactive. Triggered transformation methods use a single prediction model based on a drift detection algorithm (e.g., ADWIN [14]). If a drift warning is signalled, the ML model is adjusted [17, 18].

Our approach is based on a detector strategy, where the system contains a primary neural network for output prediction and a secondary neural network for the model update in the background (Figure 1). Thus, we first identify a model by ML techniques, then we monitor the up-to-dateness of the model over time.

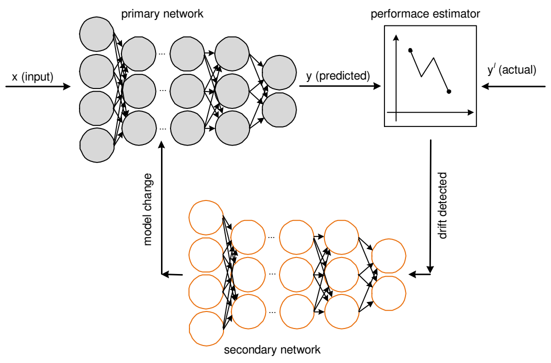

Figure 1: Online Deep Learning workflow. The first model (primary network) is used to deliver actual predictions (y) from input data (x); the second model (secondary network) is updated in the background. If the performace estimator detects a model degradation, the model is replaced.

3.1. Model Identification

We use x_t to represent an array having the feature of a dataset instance *i*. The primary neural network (Figure 1) accepts x_t as input, builds up a first model, receives a new incoming value x, and delivers a predicted result y.

After the actual outcome y' is observed, it is used to update the model of the secondary neural network (Figure 1). A performance estimator (Figure 1) follows the predicted (y) and ground truth (y') results.

The primary neural network $f(x_i; \lambda_t)$ takes instance measurements as inputs to predict outcomes. We use λ_t to represent the weights and biases of the primary neural network, initially trained using the training dataset.

The secondary neural network $f(x_i; \lambda_m)$ is being updated behind the scenes by an online optimizer. λ_m is initialized to the same parameters as the initial parameters λ_t. Afterward, at the evaluation of new incoming samples, λ_m is constantly updated, using new incoming data within the latest time window, as the time window occasionally slides forward (Figure 2).

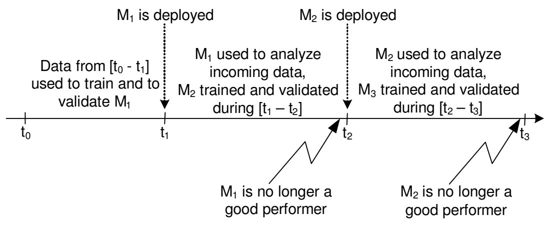
M₁ is deployed
M₂ is deployed
Data from [t₀ - t₁] used to train and to validate M₁
M₁ used to analyze incoming data, M₂ trained and validated during [t₁ – t₂]
M₂ used to analyze incoming data, M₃ trained and validated during [t₂ – t₃]
t₀
t₁
t₂
t₃
M₁ is no longer a good performer
M₂ is no longer a good performer

Figure 2: The original dataset is used at time t_1 to train and validate M_1, a first version of the model. At time t_2 the model M_1 is obsolete and needs to be replaced: incoming data from the time span $t_1 \ldots t_2$ are used to continuously retrain the new model M_2, which is deployed at time t_2 replacing M_1.

Real-world datasets usually do not include details regarding the exact starting time and ending time of the drift, since they are usually affected by hidden aspects that cannot be calculated. Thus, comparing the general predictive accuracy of other drift-handling techniques on real-world datasets is a prevalent practice. Metrics such as accuracy or Area Under the Curve (AUC) are used for classification problems, and the Mean Absolute Error (MAE) is used for regression ones.

3.2. Model Up-to-dateness Monitoring

The performance estimator (Figure 1) aims at detecting the performance slump, so that, if a performance degradation is detected, the currently deployed neural network (initially, it is the primary network of Figure 1) is replaced by an updated neural network (the secondary network of Figure 1). Thus, whenever a revision of the model is evaluated as necessary, the optimizer (e.g., Adam or Stochastic Gradient Descent – SGD) revises the secondary model parameters λ_m after a time window has elapsed. The optimization is a stochastic gradient descent method and minimizes the loss function (i.e., cross-entropy in our approach)

Cross-entropy:

$$-y_i \log f(x_i; \lambda_m) + (1 - y_i) \log(1 - f(x_i; \lambda_m))$$

We use a batch task collecting the recently gathered data (x_i, y_i) of size k (number of samples in the considered time span). We perform some batch SGD updates n times: n is suitably chosen (see Section 5.1).The performance estimator signals the degradation of the operating neural network and activates a replacement.

Our proposed approach initially requires the primary network to identify a model based on data collected over a time span; the identified model is then deployed. Next, the approach compares the results (y) predicted by the deployed model (which at the beginning is the model

Algorithm 1 Online evaluation

```
1: Input: $\lambda_t, \lambda_m$, collected data $\{x_i, y_i\}$
2: $\#s_y$ stores results predicted by the network
3: $\#s_{y'}$ stores results actually observed
4: for $i$ in incoming data do
5:     $s_y.\ add(x_i, f(x_i; \lambda_t))$ # update predicted results
6:     $s_{y'}.\ add(x_i, y_i')$        # update observed results
7: end for
8: if Detector($s_y$, $s_{y'}$) then
9:     $\lambda_t \leftarrow \lambda_m$ # replace the primary neural network
10: end if
11: $\lambda_m \leftarrow SGD(\lambda_m, s_{y'})$
```

from the primary network) with the actual ones (y'), using a specific metric based on the problem we are solving. According to the metric (line 8 of Algorithm 1), if the performance of the currently deployed model is worse than the performance of that same model during the previous time span (i.e., we observe a model drift), the currently deployed model is getting worse, and it needs to be replaced. The model from the secondary network is continuously retrained with new incoming data, keeping this replacement model ready to jump in: when requested, the model from the secondary network (the one continuously retrained) replaces the previously deployed model, thus becoming the newly deployed model. In other words: the detector compares the metric within the actual time span with the metric from the previous time span. If the metric for the in-use model decreases, the in-use model is replaced by the continuously retrained model from the secondary network. Algorithm 1 resumes the implementation in each time span.

The critical issue is now about the selection of the metric according to which a drift is detected. Usually, it is required to select a decision threshold for a deployed model to achieve some action (e.g., to predict true or false). One traditional method in ML is picking a threshold from a set of potential thresholds to get good tradeoffs on specific metrics, such as accuracy and sensitivity. Nonetheless, frequently such thresholds are manually set. Therefore, if a model updates new samples, the previously manually set threshold may be weak. Manually correcting multiple thresholds across numerous models is breakable. One mitigation approach for this issue occurs in [5], in which thresholds are learned via straightforward evaluation on holdout validation data.

In our approach, we detect drifts by measuring all the performances via AUC, which is a benchmark for binary classification. Given that: $Sensitivity = \frac{TP}{TP+FN}$ and $Specificity = \frac{TN}{FP+TN}$, the Area Under the Curve AUC is computed as: $Sensitivity - (1 - Specificity)$ Thus, according to the metrics, to detect drifts correctly AUC has to be evaluated as new data are processed.

4. Implementation Details

We now describe the implementation details of the proposed approach: tool selection and dataset selection.

4.1. Tool Selection - MLOPs

Any MLOPs relies on several steps and software modules. The plain ML workflow has a streamlined structure and it is set up by three major steps: data acquisition and preparation (DAP), to process and augment data for use by models; model development and training (MDT), to build the model based on a set of goals; and model deployment and operations (MDO), for integrating model predictions into the business and build some system-wide features. The market offers several very good tools, some of them taking care of all three steps: however, in most cases, one tool performs very well on one step (DAP, or MDT or MDO), only, and to date, no tool performs very well on all the steps. Thus, to select the proper tools, we build up an evaluation matrix of the major open-source tools.

In our approach, we enrich the plain ML workflow to fulfill the requirements of Figure 1, which includes a re-training and a re-deployment of the process model. The resulting enriched workflow is depicted by Figure 3, where the steps are suitably orchestrated.

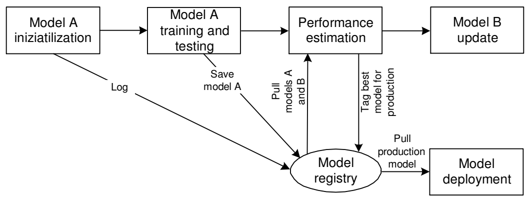

Figure 3: The enriched workflow, derived from Figure 1, where the steps are suitably orchestrated.

As a result, we choose the following tools (Figure 3):

- Airflow orchestrator: Airflow is a platform created to write, plan and monitor workflows programmatically. Airflow features scalability, dynamicity for pipeline generation, and extensibility to define own operators. Airflow orchestrates *Model A initialization*, *Model A training and testing*, *Performance estimation*, *Model B update*.
- MLflow model registry: MLflow is an open-source platform for end-to-end ML lifecycle management. It allows one to track experiments, manage and deploy models, and host MLflow templates as a REST endpoint. MLflow is the repository of all the identified models (*Model registry*).
- BentoML deployment: BentoML is a high-performance framework for serving and deploying ML models either in an offline or online fash-

ion. It supports multiple ML frameworks, such as Tensorflow, Keras, and many others. BentoML is used to deploy the model in the online production system (*Model deployment*).

4.2. Tool Selection - The Dataset

We need a dataset to train, validate, and test the model in a binary classification problem: predicting an output value according to some incoming values.

Given the current CoViD-19 pandemics and also in the light of [19, 20], we choose a dataset containing data from blood examinations from the Hospital Israelita Albert Einstein in São Paulo, Brazil [2]. The dataset has one instance per patient, with no missing data, and collects data about age, sex, hematocrit, hemoglobin, red cell count, etc. for a total of some 111 parameters. The dataset also collects one boolean output value (RT-PCR test result, 1 if CoViD-19 detected, 0 if CoViD-19 not detected). We want our model to classify the patients and, moving from the values measured in the blood sample, to predict the test outcome y, i.e., if the patient is/is NOT affected by CoViD-19. The dataset includes data acquired:

- from November 1^{st}, 2019 till February 16^{th}, 2020, when patients did not accomplish any RT-PCR test, so there is no detected case of CoViD-19 (i.e., no measured y' output value). These samples are discarded from our experiment; and
- from February 25^{th}, 2020 till August 11^{th}, 2020, when patients accomplished a RT-PCR test and come with a true/observed y' value. In our experiment, we use this part, including 8642 patients admitted to the hospital (6625 patients are not affected by CoViD-19, 2017 patients are).

5. Experimental Study

We now describe the experimental study: we apply the approach to blood examinations from [2]; we build up a predictive model which classifies if the patient is affected by CoViD-19 or not; we evaluate the performances.

5.1. Model Identification over CoViD-19 Data

We split the dataset into two parts by entry date: model training and validation data are from Feb 25^{th}, 2020 to Mar 25^{th}, 2020 (validation split is 0.8); model testing data are from Mar 26^{th}, 2020 to Aug 11^{th}, 2020. Approximately, 5 months of the evaluation samples include the evolution of the CoViD-19 virus and a concentration of RT-PCR positive test results.

The algorithm for model identification is a neural network, which uses a grid search. The data split is a K-fold cross-validation with 10-fold. The performance metric is an AUC-score as a compromise between precision and sensitivity. The performance evaluation is the best model performance on outer folds. Also, data are scaled to have a mean value of 0 and a variance of 1, so that many ML algorithms can work at their optimal implementation.

We select the hyperparameters of our neural network based on a grid search using the Keras Tuner library [21]. In particular, we use a HyperBand Tuner [22], an algorithm designed for hyperparameter optimization with early stopping for fast convergence.

As shown in Table 1, we discover some potential hyperparameters for our one-hidden-layer neural network. The model is, however, supported by a learning rate scheduler with a patience level of 15, monitoring the degradation of the validation loss.

Hyperparameter	**Value**
Learning rate	1e-4
Neurons	224
Activation function	Relu
Optimizer	Adam
Dropout	0.2

Table 1
Best hyperparameters (Keras Tuner library). The dropout layer technique limits the overfitting of the neural network.

The hyperparameter for the number of iterations n of updates on the secondary network is chosen based on some analysis concerning its size. With small values for n, a decrease in the AUC occurs. Instead, overfitting of the model occurs with greater values for n. An experimental tradeoff value of 75 achieves the best performance.

5.2. Model Evaluation over CoViD-19 Data

Once the first version model has been identified, we deploy it and run it on new incoming data, i.e., from Mar 26^{th}, 2020 to Aug 11^{th}, 2020. As the model drifts away from the ground truth, we mark that timestamp as 1^{st} drift, 2^{nd} drift etc., and update the model consequently.

Figure 4 reports drift detections over time, where drifts are gradual and not recursive or sudden [11]. The more data come in, the higher the chance that the model must be replaced by a more recent one. This is particularly true in the earlier stages of the experiment: then, drifts become less "dense". Whenever the AUC detects a drift, the new model continuously retrained by the secondary network replaces the previously deployed one.

We compare our model with some classic offline and online models: we consider the same training data and the same validation data. We consider here some current ML models for binary classification, including Logistic Regression (LR), Random Forest (RF), XGBoost [23], and other offline artificial neural networks; we also compare

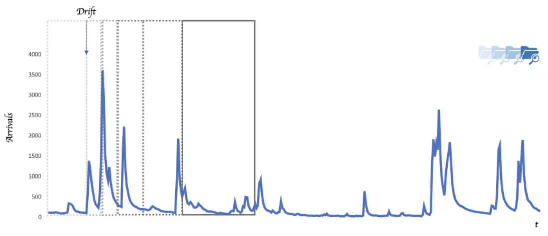

Figure 4: Progressive update of the currently deployed network. Rather than by manual selection of the time span window, whenever the AUC detects a drift, the secondary network replaces the currently deployed network.

AUC estimation				
Approach	**Online**	1^{st} **drift**	3^{rd} **drift**	**Avg**
Our approach	Yes	0.784	**0.794**	**0.806**
Online LR	Yes	**0.819**	0.738	0.794
Ad. Random Forest	Yes	0.790	0.756	0.745
Logistic Regression	No	0.768	0.715	0.754
Random Forest	No	0.734	0.700	0.741
XGBoost	No	0.798	0.745	0.782
Artif. Neural Net.	No	0.784	0.715	0.754

Table 2
Results from different models, highlighting the best AUC score in each drift section and the best average score.

our approach with some standard online learning methods that are functional, including online LR and adaptive RF [16, 24]. The base neural network of our approach comprises one hidden layer with a RELU activation function connected to a dropout layer to limit the overfitting. Figure 5 reports about our approach, one online model (online LR), and two offline models (RF and XGBoost), measuring their respective AUC.

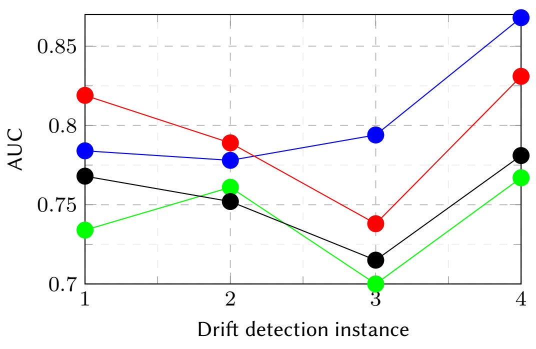

Figure 5: Comparison of the AUC after the drifts for the proposed model (blue), of OLR (red), RF (green), LR (black). The proposed approach starts poorly concerning the online learning regression method. Still, the model of the proposed approach shows more longevity and accuracy in the long term than the other methods, particularly in the third drift detection instance where the model of the proposed approach outperforms all the other models.

However, our approach and the online logistic regression model support a robust AUC over time despite oscillations. The proposed approach yields the most elevated AUC in the third frame. A comprehensive comparison of all the models used is presented in Table 2.

For the average AUC over all the time spans, our approach (0.806) performs 3% better than the best offline and the best online models (0.782). The distinctions are more relevant in the third frame, where our approach (AUC 0.794) is 6% higher than that of the best offline model (0.745) and 5% better than that of the best online model (0.756). A comprehensive view of the models indicates the primacy of online strategies.

5.3. Discussion

We now explore the shifts in patient attributes over the pandemic, explaining the need for online models. Again, in the light of explainable AI [25, 26], we examine the advantages of a model replacement mechanism and the impacts of hyperparameters of the neural network.

Figure 6 depicts the hemoglobin (Hgb) count during the 2^{nd} and 3^{rd} time spans, i.e. after the 2^{nd} and the 3^{rd} drift. During the 2^{nd} time span, positive (CoViD-19 = “yes”) and negative (CoViD-19 = “no”) outcomes are coupled to balanced Hgb count: i.e., positive and negative patients feature similar levels of Hgb counts. In the 3^{rd} time span, the Hgb counts of patients with a negative outcome differ from the Hgb counts of patients with a positive outcome: thus, during the 3^{rd} drift, the Hgb count can be a differentiating feature between positive and negative patients. Because of the deteriorating likeness of the patients during the 2^{nd} time span, the trained models are instantaneously impaired, including our initial neural network. The rationale for this shift requires further investigation:however, a first guess could probably be in the direction of variants on the side of the CoViD-19 virus, where variants lead to changes in the hematological examinations.

6. Conclusions

The application and deployment of machine learning (ML) to infer knowledge from huge amounts of data is a challenging endeavour. In this context, significantly altered data – aka concept drift – and their effects on the forecast quality are limiting a more widespread of ML.

In this paper, we focus on a novel concept of the drift-handling methodology for ML systems in a real-world environment. We build an online deep learning system for forecasting outcomes, using MLOPs practices and tools, and aim at keeping the in-use model constantly up-to-date. Typically, a vital issue of the approach is

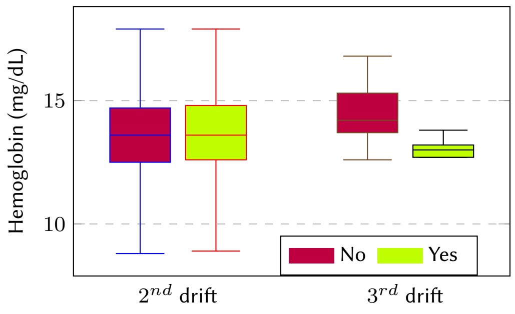

Figure 6: Relevant changes in the Hemoglobin counts and RT-PCR outcome results from the 2^{nd} to the 3^{rd} time span. During the 2^{nd} time span, Hemoglobin counts are similar both for positive and negative patients. In the 3^{rd} time span, Hemoglobin counts differ for positive and negative patients, probably due to the changing characteristics of the virus.

the choice of the time interval between two successive updates of the in-use model. Manual decisions regarding the length of the windows are uncertain and impractical. Thus, we keep a secondary network constantly updated by new incoming data: as the performance of the in-use model, measured in terms of area under the curve (AUC), are worsening, the model from the secondary networks jumps in, replacing the currently in-use model.

As a validation scenario, we deploy our approach on real-world clinical data collected during the CoViD-19 pandemics by the Hospital Israelita Albert Einstein in São Paulo, Brazil [2]; over a five-month-long evaluation time span, we achieve a more promising performance than standard online and offline learning techniques.

The generality of our approach contains some particular restrictions. Despite these favorable outcomes, the results are based on one dataset, only. Identifying other real-world datasets with incremental concept drift practices will add value and reliability to our approach. Due to its nature, our approach is appropriate for handling incremental concept drift. Sudden or reoccurring concept drifts will likely demand a diverse strategy, such as changing between two utterly distinct forecasting models, e.g., one model for the everyday problems and one for the harsh circumstances [16], or one forecast model for summertime and one for wintertime, respectively.

Consequently, more analysis is required to explore the correct matching of drift-handling scenarioes. Also, developing differently-sized detection windows on the forecast implementation requires additional study. Finally, the triggered adaption method executed in this work is founded on the belief that actual labels are obtained soon after the model computes a forecast, but this assumption may be not always valid or feasible.

Acknowledgments

G.P. is partially funded by the EU H2020 program: "PERISCOPE: Pan European Response to the ImpactS of CoViD-19 and future Pandemics and Epidemics" (grant n. 101016233).

References

[1] C. Combi, Editorial from the new editor-in-chief: Artificial intelligence in medicine and the forthcoming challenges, Artif. Intell. Medicine 76 (2017) 37–39. URL: https://doi.org/10.1016/j.artmed.2017.01.003. doi:10.1016/j.artmed.2017.01.003.

[2] Kaggle, CoViD-19 open research dataset challenge (CORD-19) - website, May 12th, 2022. https://www.kaggle.com/datasets/allen-institute-for-ai/CORD-19-research-challenge/discussion/139347.

[3] G. I. Webb, R. Hyde, H. Cao, H. Nguyen, F. Petitjean, Characterizing concept drift, Data Min. Knowl. Discov. 30 (2016) 964–994. URL: https://doi.org/10.1007/s10618-015-0448-4. doi:10.1007/s10618-015-0448-4.

[4] T. Diethe, T. Borchert, E. Thereska, B. Balle, N. Lawrence, Continual learning in practice, CoRR abs/1903.05202 (2019). URL: http://arxiv.org/abs/1903.05202. arXiv:1903.05202.

[5] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, J. Crespo, D. Dennison, Hidden technical debt in Machine Learning systems, in: C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, R. Garnett (Eds.), Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada, 2015, pp. 2503–2511. URL: https://proceedings.neurips.cc/paper/2015/hash/86df7dcfd896fcaf2674f757a2463eba-Abstract.html.

[6] J. Knoblauch, H. Husain, T. Diethe, Optimal continual learning has perfect memory and is NP-hard, in: Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event, volume 119 of *Proceedings of Machine Learning Research*, PMLR, 2020, pp. 5327–5337. URL: http://proceedings.mlr.press/v119/knoblauch20a.html.

[7] G. Widmer, M. Kubat, Learning in the presence of concept drift and hidden contexts, Mach. Learn. 23 (1996) 69–101. URL: https://doi.org/10.1007/BF00116900. doi:10.1007/BF00116900.

[8] J. B. Gomes, E. M. Ruiz, P. A. C. Sousa, CALDS: Context-aware learning from data streams, in: Proceedings of the First International Workshop on Novel Data Stream Pattern Mining Techniques, StreamKDD '10, Association for Computing Machinery, New York, NY, USA, 2010, p. 16–24. URL: https://doi.org/10.1145/1833280.1833283. doi:10.1145/1833280.1833283.

[9] N. M. do Nascimento, P. S. C. Alencar, C. Lucena, D. D. Cowan, A context-aware machine learning-based approach, in: I. Onut, A. Jaramillo, G. Jourdan, D. C. Petriu, W. Chen (Eds.), Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering, CASCON 2018, Markham, Ontario, Canada, October 29-31, 2018, ACM, 2018, pp. 40–47. URL: https://dl.acm.org/citation.cfm?id=3291297.

[10] S. H. Bach, M. A. Maloof, Paired learners for concept drift, in: Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy, IEEE Computer Society, 2008, pp. 23–32. URL: https://doi.org/10.1109/ICDM.2008.119. doi:10.1109/ICDM.2008.119.

[11] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, G. Zhang, Learning under concept drift: A review, IEEE Trans. Knowl. Data Eng. 31 (2019) 2346–2363. URL: https://doi.org/10.1109/TKDE.2018.2876857. doi:10.1109/TKDE.2018.2876857.

[12] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, Intelligent Data Analysis 8 (2004) 286–295. doi:10.1007/978-3-540-28645-5_29.

[13] C. Combi, B. Oliboni, G. Pozzi, A. Sabaini, E. Zimányi, Enabling instant- and interval-based semantics in multidimensional data models: the t+multidim model, Inf. Sci. 518 (2020) 413–435. URL: https://doi.org/10.1016/j.ins.2019.12.074. doi:10.1016/j.ins.2019.12.074.

[14] A. Bifet, R. Gavaldà, Learning from time-changing data with adaptive windowing, in: Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA, SIAM, 2007, pp. 443–448. URL: https://doi.org/10.1137/1.9781611972771.42. doi:10.1137/1.9781611972771.42.

[15] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. 46 (2014) 44:1–44:37. URL: https://doi.org/10.1145/2523813. doi:10.1145/2523813.

[16] L. Baier, Concept Drift Handling in Information Systems: Preserving the Validity of Deployed Machine Learning Models, Ph.D. thesis, Karlsruhe Institute of Technology, Germany, 2021. URL: https://nbn-resolving.org/urn:nbn:de:101:1-2021091504594834614171.

[17] M. Pechenizkiy, I. Zliobaite, Handling concept drift in medical applications: Importance, challenges and solutions, in: T. S. Dillon, D. L. Rubin, W. M. Gallagher, A. S. Sidhu, A. Tsymbal (Eds.), IEEE 23rd International Symposium on Computer-Based Medical Systems (CBMS 2010), Perth, Australia, October 12-15, 2010, IEEE Computer Society, 2010, p. 5. URL: https://doi.org/10.1109/CBMS.2010.6042653. doi:10.1109/CBMS.2010.6042653.

[18] I. Zliobaite, Combining similarity in time and space for training set formation under concept drift, Intell. Data Anal. 15 (2011) 589–611. URL: https://doi.org/10.3233/IDA-2011-0484. doi:10.3233/IDA-2011-0484.

[19] C. Combi, G. Pozzi, Clinical information systems and artificial intelligence: Recent research trends, Yearbook of Medical Informatics 28 (2019) 083–094. URL: https://www.thieme-connect.de/products/ejournals/abstract/10.1055/s-0039-1677915. doi:10.1055/s-0039-1677915.

[20] C. Combi, G. Pozzi, Health informatics: Clinical information systems and artificial intelligence to support medicine in the CoViD-19 pandemic, in: 9th IEEE International Conference on Healthcare Informatics, ICHI 2021, Victoria, BC, Canada, August 9-12, 2021, IEEE, Los Alamitos, CA, USA, 2021, pp. 480–488. URL: https://doi.org/10.1109/ICHI52183.2021.00083. doi:10.1109/ICHI52183.2021.00083.

[21] T. O'Malley, E. Bursztein, J. Long, F. Chollet, H. Jin, L. Invernizzi, et al., Keras Tuner, https://github.com/keras-team/keras-tuner, 2019.

[22] L. Li, K. G. Jamieson, G. DeSalvo, A. Rostamizadeh, A. Talwalkar, Hyperband: A novel bandit-based approach to hyperparameter optimization, J. Mach. Learn. Res. 18 (2017) 185:1–185:52. URL: http://jmlr.org/papers/v18/16-558.html.

[23] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: B. Krishnapuram, M. Shah, A. J. Smola, C. C. Aggarwal, D. Shen, R. Rastogi (Eds.), Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, ACM, 2016, pp. 785–794. URL: https://doi.org/10.1145/2939672.2939785. doi:10.1145/2939672.2939785.

[24] H. M. Gomes, A. Bifet, J. Read, J. P. Barddal, F. Enembreck, B. Pfahringer, G. Holmes, T. Abdessalem, Adaptive random forests for evolving data stream classification, Mach. Learn. 106 (2017) 1469–1495. URL: https://doi.org/10.1007/s10994-017-5642-8. doi:10.1007/s10994-017-5642-8.

[25] C. C. Yang, Explainable artificial intelligence for predictive modeling in healthcare, J. Heal. Informatics Res. 6 (2022) 228–239. URL: https://doi.org/10.1007/s41666-022-00114-1. doi:10.1007/s41666-022-00114-1.

[26] C. Combi, B. Amico, R. Bellazzi, A. Holzinger, J. H. Moore, M. Zitnik, J. H. Holmes, A manifesto on explainability for artificial intelligence in medicine, Artif. Intell. Medicine 133 (2022) 102423. URL: https://doi.org/10.1016/j.artmed.2022.102423.