A Robust Drift Detection Algorithm with High Accuracy and Low False Positives Rate

Maxime Fuccellaro[1,2], Laurent Simon[1] and Akka Zemmari[1]

[1]*University of Bordeaux, CNRS, Bordeaux INP, LaBRI, UMR 5800, 351, cours de la Libération F-33405 Talence*

[2]*Mangrove, 87 Avenue des Aygalades, 13015 Marseille*

Abstract

The number of decision-making processes that rely on machine learning models to operate has been increasing in recent years. Safety of those systems is compromised when models deviate from their expected behavior. One root cause is a shift in the underlying data distribution, known as concept drift. A direct consequence of concept drift is a rapid drop in model's predictive power. Accurate detection of drift is essential as false alarms lead to unnecessary down time and undermine confidence in the drift detection model. This paper introduces Real-Drift Detector (RDD), a drift detector that is not triggered by virtual drift. RDD does not need use class labels during the inference phase to operate. Our detector outperformed the state of the art in an extensive benchmark on a large panel of well-known datasets used in drift detection.

Keywords

Concept Drift, Real Drift, Virtual Drift, Unsupervised, Hypothesis Testing

1. Introduction

More and more online systems rely, at least partly, on a form of machine learning model to operate. The widespread integration of Artificial Intelligence based model has its roots in the constant progress made in the field, which enables models to solve increasingly complex tasks well suited to real world applications. The democratisation of Machine Learning (ML) models allows non-experts to use it to automate repetitive tasks and is simplified by the easy access and processing of large quantities of data required to train predictive models. The emergence of cloud computing has also been accelerating the industrial use of ML models in production.

However, ML models can be crippled by a wide range of problems that raise serious questions regarding their impact on the safety of systems and their consequence on society. Some models inadvertently induce bias in their predictions [1] such as black box models that are therefore excluded from applications where explanability is a critical feature such as loan applications. However, ML models are often poorly adapted to detect out of distribution samples [2] that are not classified correctly. Models can then see a drop of performance while in the inference phase due to a change in the underlying data distribution [3]. A shift of distribution is referred as *Concept Drift* (CD) and its detection will be the focus of this paper.

Machine learning models are built under the hypothesis that data seen during the training phase share the same distribution as unseen future data. Concept drift appears when the underlying distribution of a data source changes over time. If the static distribution hypothesis is violated, historic data cannot be used to predict the future and predictive models see their performances drop. Concept Drift can impact every ML domain including video analysis [4], [5].

In contrast to anomaly detection, where the goal is to isolate few out of distribution samples, concept drift will cause a large part of the data to deviate from past distributions. One way to categorize drifts is with its impact on a model's performance. *Virtual drift* is used to describe a distribution change that does not impact a model while *real drift* does. Let X be a set a variables used to predict the target class vector y. We distinguish three root causes of concept drift: it may come from the change in the class distribution $\mathbb{P}(X \mid y)$, the feature space $\mathbb{P}(X)$ or the class priors $\mathbb{P}(y)$. Where the change in distribution is rapid, the drift is said to be abrupt. Drift is incremental when the distribution shifts slowly over time. Recurrent drift is defined as a distribution that oscillate in between two or more concepts. Drift detection differs from outliers detection [6] as the goal is to identify and take actions to deal with a global distribution change and not to remove out of distribution samples.

We present RDD: Real-Drift Detector, a unsupervised drift detection method based on the supervised partition of feature space aiming to detect local distribution changes that impacts models performances. RDD works

Proceedings of the Workshop on Artificial Intelligence Safety 2023

maxime.fuccellaro@u-bordeaux.fr (M. Fuccellaro);
laurent.simon@u-bordeaux.fr (L. Simon);
akka.zemmari@u-bordeaux.fr (A. Zemmari)
https://www.labri.fr/perso/lsimon/ (L. Simon);
https://www.labri.fr/perso/zemmari/ (A. Zemmari)
0000-0003-0544-5503 (L. Simon); 0000-0002-9776-0449
(A. Zemmari)

CEUR Workshop Proceedings (CEUR-WS.org)

in any number of dimensions. Our detector does not need labels during inference and outperforms the state of the art in a thorough experiment. In Section 2, we present related work and position our paper. The RDD algorithm is described in Section 3. In Section 4, the experimental protocol is presented and the results are discussed. Section 5 concludes this paper.

2. Related Work

Over the last few years, concept drift has become a major field of research in the machine learning community. Focus was mostly aimed at models dealing with drift as well as drift detectors. Recent advances in detecting drift when true class labels are available led algorithms to achieve almost perfect detection. Other (more realistic) contexts still show room for improvement.

To prevent drift from affecting ML models, several mechanisms have been proposed. Assuming that recent data share the same distribution as upcoming data, one way is to continuously update a pool of models. In [7] a batch of new data is scored by a pool of models, the individual model's contribution are weighted by it's recent performances. At each new batch, a model is trained on it and added to the pool while a long term poor performing one is removed. This ensures good performances in the presence of concept drift and a fast adaptation on recurrent drifts. Methods that work on a dynamic pool of models have been thoroughly studied [8], [9].

When detecting a drift, the consensus is that an increase of the error rate of a model means the presence of concept drift. Detection methods based on that hypothesis have been given a lot of attention as this methodology is able to systematically detect real drift while consistently ignoring virtual drift. The detection methods presented in [10] and [11] work by monitoring a model's error rate. A drop of performance is interpreted as a presence of drift. Different statistical tests are used to monitor the error rate and signal drifts.

Both updating a pool of models and monitoring the error rate perfectly deal with drift. As virtual drift does not impact accuracy, it is systematically ignored. However, in order to work, both approaches need access to the true labels immediately after the inference phase. This is not realistic for real-world scenarios when true class labels are almost never available promptly and are sometimes never known. To address this issue, several ways of dealing with drift in an unsupervised manner have been studied.

To detect drift without class labels availability, window-based technique have been studied. The authors of [12] introduce ADWIN, an adaptive sliding window algorithm. It works by keeping a reference window containing past instances. The window widens when no change is detected, while its size decreases rapidly in the presence of drift. The detection-mechanism works by repeatedly splitting the window, based on the time of appearance into two smaller sets. A drift is detected when the averages of the values of two sets are statistically different. Several other window-based detectors have been presented since such as [13].

To avoid detecting drift over one-dimension sliding windows, [14] detects swift or gradual changes in data values with minimum enclosing balls. A ball is defined as a centroid and the minimum radius that enables to include all of the samples in the ball. A drift is detected when too many values are labelled as outliers, in which case the centroid is updated.

To circumvent the unavailability of true labels, the authors of [15] trained a model to distinguish past data from recent data. All timestamp-like aggregates are removed prior to training the model to prevent trivial identification. The ability of the model is assessed using the AUC metric, using a threshold of .75 in their paper. Using time to find drift, the authors of [16] include the timestamp attribute in the observations and train a model on past and recent data to predict the target variable. If the timestamp attribute is an informative feature, the target variable depends on time and the presence of a drift is assumed.

The authors of [17] present another way of detecting drift in an unsupervised manner. A first model (teacher) is trained on past labelled data, a second model (student) is trained to mimic the behavior of the teacher model. During the inference phase the authors monitor the error rate of the student model and use [12] to trigger an alarm. In [18], drift detection in an unsupervised image classification context is studied. The authors first apply a dimension reduction technique before using a two-sample test to find drift. A number of dimension reduction and two-sample tests including (MMD [19] and KS) are evaluated on an extensive study of different types of shift applied to images. In [20] the authors incorporate the target class in the dimension reduction mechanism, enabling the detector to ignore virtual drift.

The idea behind concept drift detection by statistical tests is that a distribution change will be a strong indicator of drift [14], [18]. A distribution change will enable the detection of drift, but won't discriminate between real drift and virtual drift. To the best of our knowledge, few algorithms are able to discriminate between real drift and virtual drift without access to true labels after detection. In this paper we introduce RDD a detector that does not need true class labels to operate during the inference phase. RDD works in any dimension and successfully discriminates between real and virtual drift.

3. RDD

3.1. Our detector

The idea behind our model is that a real drift changes the distribution of regions made by of a class dependant partitioning while virtual drift does not. We use a decision tree to partition the feature space into regions of homogeneous class labels.

Our intuition is that a data distribution change in a leaf between the training phase and inference phase indicates a drift. It is our assumption that a real drift is likely to change in which region the observations are assigned to. Such misplaced samples are likely to have a different distribution than that of the training observations. A distribution change leading to virtual drift is unlikely to be seen locally as it may affect less the way observations are distributed in leaves.

In order to better discriminate real drift from virtual drift, each region is attributed a weight. The weights represent the ability of a given region to correctly assign a label to an observation. A region that only contains a single class of observations will have a large weight. While a region that cannot well separate samples on their label will have a small weight. This is done to reduce the risk of sudden class imbalance to be detected as a distribution change. Our model signals a drift when enough regions flag their distribution as changing.

3.2. Mathematical Background

During the initialization step, a decision tree classifier ($\mathcal{M}$) is fit over the training data. Like most drift detection algorithms, we assume the training data is sampled from one single concept. We do not consider the training data to include past concepts that might offset the detector. For both the training (T) and inference (I) test sets, we consider the variables to follow a normal distribution. This hypothesis is required to test for homoscedasticity in the latter. After discarding all leaves that contain too few samples or that are not pure, we store, for each leaf, the training instances that belong to it. Let T_k, I_k be the training and inference data within leaf k of class c. For the test to be significant [21], we remove all leaves in the decision tree containing less than a number of observations ν. We set $\nu = 20$. We have $\forall k, \min(|T_k|, |I_k|) \geq \nu$ as well as $Y_{T_k} = \hat{Y}_{I_k} = c$.

By construction, the leaves of a decision tree don't hold the same separation power of class labels. The intuition is that leaves containing pure class labels should be less affected by $\mathbb{P}(y \mid X)$ concept change as they are generally further away from the decision boundary. On the contrary, impure leaves are more likely to experience a distribution change due to a $\mathbb{P}(y)$ drift or to be subject to misclassifications that may impact the inference distribution. During the initialisation step, leaves that cannot well separate class labels are removed, all leaves with less than 20% impurity are dropped. During early experimental runs, we found that setting a low minimum impurity percentage yields very few leaves with enough samples to conduct the statistical test. We also found that setting a high value prevents us from confidently assigning a class to a leaf. We set the maximum impurity value at 20% as it offers a good compromise, although this value could be changed based on the data at hand. In an effort to rank the remaining leaves, we attribute to each leaf a weight which corresponds to a leaf's purity during training.

Weights somehow capture the separation power of a leaf. If a given observation is classified at a leaf with high weight, we may expect that the probability of this observation to be misclassified is low. On the other hand, a leaf with a lower weight will be more susceptible to assign the wrong label to an observation. Our goal by ranking the leaves based on their predictive power is to help our detector ignore virtual drift.

The detection step is detailed in Algorithm 1. In line 1-5 test set observations are attributed to L_{test} based on the leaf they are at. In line 6, we go over each leaf that contains test data; in line 7 we initialize the *drift features* DF variable that tracks the number of drifted features. In line 8, the number of observations at a leaf is checked. In lines 9 through 13, for all leaves containing enough test instances, we proceed to do a Levene test of variance equality on all dimensions between the inference and training set contained in a leaf. We choose here the Levene test as it is adequate when the data distribution may slightly deviate from the normal one. Of course, other tests could be used, based on the knowledge of the underlying data distribution to improve the detector's performances (when the data distribution is strictly normal, the Barlett's test should be used; the Brown–Forsythe test may also be an alternative when the data does not follow a normal distribution). We did not make any assumptions on the distribution and independence of variables, it will be the focus of future work.

In lines 14 through 18, leaves are classified as drifting if the ratio of features that fail the homoscedasticity test exceeds the user defined γ threshold. In line 21, the algorithm flags a drift if the weighted average of leaf's drift-labels exceeds the user defined β threshold.

3.3. Hyper-parameter discussion

The first hyper-parameter is α. Setting a low value for the α parameter reduces the risk to make a type I error, which, in our case, indicating drift when there is not. The α parameter was set to 0.01.

The γ parameter is the minimum ratio of features to reject the equal variance hypothesis. A low γ parameter

Algorithm 1 RDD - Inference

```
Inputs:
- $\mathcal{M}$ : Trained Decision Tree Model
- $L_{train}$ : Dictionary of leaves mapping to the training instances
- $I \in \mathbf{R}^{\mathbf{d}\times\mathbf{m}}$ : Test set with $\mathbf{d}$ features and $\mathbf{m}$ samples
- $W$ : Leaves weights
- $d$ : Number of variables in dataset
Parameters:
- $\alpha$ : Hypothesis rejection risk
- $\nu$ : Minimum number of observation in a leaf
- $\gamma$ : Minimum ratio of $\mathcal{H}_0$ rejection for a leaf to drift
- $\beta$ : Minimum ratio of leaves to drift to trigger an alarm.
Variables:
- $L_{test}$ : Dictionary of leafs mapping to the test instances in those leafs
- $DL$ : Drift status of leaves
- $DF$ : Number of features that drift within a leaf
for $i \in I$ do
    if $\mathcal{M}(i) \in L_{train}$ then
        $L_{test}[\mathcal{M}(i)] \leftarrow L_{test}[\mathcal{M}(i)] + i$
    end if
end for
for $i \in L_{test}$ do
    $DF = 0$
    if $L_{test}[i] \geq \nu$ then
        for $j \in [0, d-1]$ do
            if $H_\alpha : \sigma_{L_{train}[i][j]} \neq \sigma_{L_{test}[i][j]}$ then
                $DF = DF + 1$
            end if
        end for
        if $\frac{DF}{d} \geq \gamma$ then
            $DL[i] = 1$
        else
            $DL[i] = 0$
        end if
    end if
end for
Return $\sum_{i \in L_{test}} W_i * DL[i] \geq \beta$
```

means leaves will be considered as drifting if few features present a shift in variance (*i.e.* the detector will be sensible).

The β parameter is the minimum ratio of drifted leaves to signal a drift has taken place.

In order to set relevant β and γ values for our detector, we conducted a hyper parameter search on three datasets (airlines, poker and weather). We excluded those datasets from the experimental study to prevent bias. In Figure 1 we plot the influence of both β and γ on TP, TN and the H score. The H score is detailed in Section 4.

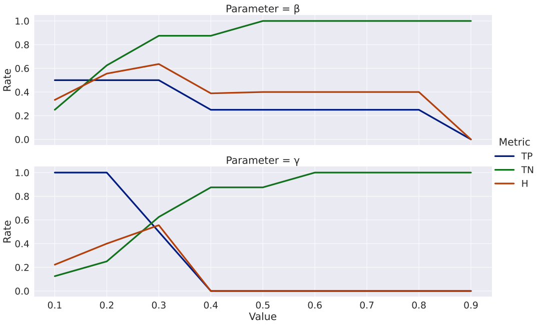

Figure 1: Influence of the β and γ parameters on the True Positive rate, True Negative rate and H score. On the β plot, $\gamma = 0.3$ and on the γ plot, $\beta = 0.3$. The graph confirms our intuition that low values tend to flag virtual drift as real drift while high values cause the detector not to detect any drift.

4. Experiment

In this experiment we assess our method's ability to detect drift while ignoring virtual drift. We extensively tested our method against a wide panel of state of the art detectors on a extensive set of both real and synthetic datasets. The benchmark used in this section are the standard ones when testing drift detectors [20], [22], [23].

4.1. Experimental Setup

The usual procedure to test algorithms suited to handle drift when true class labels are available after inference, is the test-then-train approach. A model predicts the class on a batch of samples, then, the true class is revealed and the model updates itself. The global prediction accuracy is then used to rank models.

This setup is not suited for models that do not rely on true labels availability. In most datasets used to benchmark drift handling methods, the presence of drift is only assumed or artificially introduced by sorting the observations on an attribute. To the best of our knowledge, the exact occurrence of drift is unknown for all usual datasets. The experimental setup described bellow allows us to know the exact drift occurrence and to evaluate the effect it has on a model's accuracy.

The goal of the experiment is to assess the performance of detectors on real drift and virtual drift. Two distinct perturbations are used to change the dataset: the Step Drift, where a subset of the features are shuffled and the Noise Drift, where Gaussian $\mathcal{N}(1, 1)$ noise is added to a subset of features (Gaussian noise with mean equal to 1 is used to change the mean of the distribution, not to obfuscate the signal). The idea is now to be able to artificially generate real drift and virtual drift. To create virtual drift, we add one of the two perturbations on the 25% least important features as to their predictive power of the class labels. In doing so, we hope to change the distribution of several features will not affect a predictive model's performances. To create real drift, we modify the 25% most informative features by adding one of the two perturbations. The intuition is that a change of distribution on the most important features is likely to cause a drop of performance in a predictive model. To find 25% most and least important features, we train a Random Forest Classifier over the training data. We choose this model as it is a robust, widely used model that achieves good level of performance on the datasets. For each dataset, we introduce the 2 perturbations on the 2 sets of features thus creating 4 distinct drift set.

In order to have stationary non-drifting data before adding our generated drifts, we first randomly shuffle the observations. Each dataset is then partitioned into three: a train set, a validation set and a drift set. 4 distinct copies of the drift set are independently modified with the 4 different perturbations described above. In order to assess if a drift is virtual or real, we fit a Random Forest Classifier on the train set before reporting it's accuracy on the train set, validation set and the 4 different drift sets. The drop of the model's accuracy between the different sets is used to classify drift as virtual or real. If the difference in accuracies between the validation set and the training set is lower than that of the validation set and the drift set, we consider the drift induced to be real, otherwise, it is considered a virtual drift.

Table 1 briefly describes the datasets used in the experiment. The datasets dimensions range from 11 on Hyperplane to 500 on Spam. The classification task is binary on 10 datasets and multi-class on 3. This ensures that RDD is tested in a variety of scenarios.

In table 2 we report the average accuracies of the Random Forest Classifier over the training, validation and the four different drift set. changing the most important features generates real drift while changing the least important features creates virtual drift regardless of the perturbation. There are 3 exceptions, when noise is added to the least important features, real drift is produced on the Musk dataset. When corrupting the most important features, the step perturbation produces virtual drift on the Hyperplane dataset while the noise perturbation yields virtual drift on the Waveform dataset.

Table 1
Overview of the dataset used in our experiment. All but one RW dataset are binary classification problems. 2 RW datasets contain more than 100 features. For the synthetic datasets, we limit the number of generated observations at 10 000.

Dataset	Dimensions	Classification
Adult	(48842, 66)	Binary
Bank	(45211, 49)	Binary
Cov	(110393, 51)	Multi-class (7)
Digits08	(1499, 17)	Binary
Digits17	(1557, 17)	Binary
Elec	(45312, 15)	Binary
Musk	(6598, 167)	Binary
Phishing	(11055, 47)	Binary
Spam	(6213, 500)	Binary
Wine	(6497, 13)	Binary
Hyperplane	(10000, 11)	Binary
LED	(10000, 26)	Multi-class (10)
Waveform	(10000, 41)	Multi-class (3)

Table 2
Accuracy of a Random Forest Classifier over the training, validation and drift set. Adding noise to least informative features (LN) leads to virtual drift on all datasets. Adding step perturbation to the least informative features (LS) also leads to virtual drift except for the musk dataset. When those perturbation are made on the most informative features (MN and MS), it leads to real drift across all real datasets. We highlight in **bold** perturbations that lead to real drift.

	Train	Val.	LN	LS	MN	MS
Adult	1.	.85	.85	.85	**.28**	**.59**
Bank	1.	.94	.92	.94	**.52**	**.52**
Cov	.99	.85	.84	.84	**.49**	**.46**
D08	1.	1.	.99	.99	**.77**	**.69**
D17	1.	.99	1.	1.	**.54**	**.80**
Elec	1.	.89	.87	.87	**.57**	**.62**
Musk	1.	.98	**.94**	.97	**.51**	**.56**
Phis.	.99	.97	.96	.96	**.69**	**.47**
Spam	1.	.98	.98	.98	**.58**	**.59**
Wine	1.	1.	1.	1.	**.63**	**.44**
Hyp.	1.	.87	.87	.85	**.71**	.87
LED	1.	1.	1.	1.	**.58**	**.31**
Wav.	1.	.85	.85	.85	.72	**.41**

In an effort to aggregate both the True Positives and False Negatives into one metric, we will make use of the metric 1 defined in [20]. Since we conduct our experiment on a batch mode, we removed the impact of the detection delay defined as the number of drift samples processed before signaling a drift. The Drift Accuracy ($\widehat{DA}$) is a binary value that assess the correctness of the detection, it's equal to 1 when a virtual drift is ignored or when a real drift is detected.

$$H = 2 * \frac{\widehat{DA} * TN}{\widehat{DA} + TN} \quad (1)$$

We evaluate RDD with $\alpha = 0.01, \beta = 0.3, \gamma = 0.3$ against:

- ADWIN [12] with $\delta = 0.7$
- Discriminative Drift Detector (D3) [15]

Table 3

Least Important Step Drift: The detection results on virtual drift when the least important features are shuffled. The lower the detection ratio, the better the detectors are. Our model RDD comes second being slightly outperformed by TSDD. ST comes third with relatively few wrong detections in comparison to the other detectors that wrongfully detect virtual drift.

	Adult	Bank	Cov	D08	D17	Elec	Musk	Phish.	Spam	Wine	Hyp.	LED	Wav.
RDD	**0.0**	**0.0**	**0.0**	0.1	0.1	**0.0**	**0.0**	**0.0**	**0.0**	0.1	**0.0**	**0.0**	**0.0**
ADWIN	1.0	1.0	1.0	1.0	1.0	**0.0**	1.0	1.0	1.0	1.0	**0.0**	0.1	0.1
D3	**0.0**	0.5	**0.0**	0.9	1.0	**0.0**	1.0	1.0	1.0	1.0	**0.0**	**0.0**	**0.0**
KS	0.9	1.0	**0.0**	1.0	1.0	**0.0**	1.0	1.0	1.0	1.0	**0.0**	1.0	0.1
MMD	1.0	1.0	0.3	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.2	0.2	0.6
ST	**0.0**	0.4	**0.0**	**0.0**	**0.0**	**0.0**	0.6	**0.0**	**0.0**	**0.0**	**0.0**	0.2	**0.0**
TSDD	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.1	**0.0**

- Kolmogorov-Smirnov (KS) distribution test detector, we used the implementation of [24].
- Maximum Mean Discrepancy (MMD) [19], we used the implementation of [24].
- Student-Teacher (ST) [17]
- Task Sensitive Drift Detector (TSDD) [20]

All detectors were used with their default parameter values unless specified otherwise. For the sake of readability, we only highlight the best results in Table 7.

4.2. Virtual Drift

In Table 3 we present the detections made on virtual drift induced by a Step corruption of the least important features. On the 7 detectors evaluated, 3 are able to consistently ignore this type of virtual drift: TSDD, ST and RDD. TSDD comes first with no detections on real-world (RW) datasets and almost none on synthetic data. RDD makes no False Positives (FP) on 7 RW datasets and all synthetic datasets. On 3 real-word datasets the FP is very low (0.1). The Student Teacher detector produces no FP on 8 RW datasets and on 2 synthetic ones, however on 2 RW datasets the FP rate is high as it is around .5. ADWIN along with the statistical test-based KS and MMD fail to ignore virtual drift on all but 1 RW dataset. D3 does slightly better ignoring virtual drift on 3 RW datasets. On synthetic data, relatively few FP are made by those 4 detectors.

Table 4 exhibits detection rate when adding noise to the least important features. On the Musk dataset, this type of perturbation produces real drift and therefore, detections are considered as True Positive (TP). ADWIN along with D3, KS and MMD which where not specifically built to handle virtual drift, systematically wrongfully detect drift across all real and synthetic datasets. RDD flags detects virtual drift on the Digits 08 dataset 4 out of 10 runs. Virtual drift is otherwise ignored by RDD. ST and TSDD fail to ignore the virtual drift on 2 RW datasets.

4.3. Real Drift

In Table 5 we observe real drift induced by a Step drift on the most informative features. The Hyperplane exhibits virtual drift with this corruption and low values should be regarded as TN. ADWIN, D3, KS and MMD, which all exhibit poor performance on virtual drift, now achieve almost perfect detection on RW datasets. However, D3 and ADWIN fail to detect real drift on synthetic data. Our method systematically detects real drift on 4 RW datasets and achieve good levels of detection on 3 others. Drift is detected on 50% of the runs on the phishing dataset. The ST model achieves 4 perfect detections on RW datasets and good level of detection on 3 others. The TSDD detector yields poor performance detecting only 2 drifts out of all RW datasets. On the 2 synthetic datasets with real drift, ADWIN, D3 and RDD fail to detect the drift, while TSDD detects 1. KS, MMD and ST detectors succeed in their detection.

Noise detection on the most important features results are shown in Table 6. ADWIN, D3, KS and MMD achieve perfect detection across both real and synthetic datasets. RDD detection results exceed that of ST and TSDD with 7 perfect detections on RW datasets. TSDD and ST are tied with both 5 accurate detections on RW datasets. Only our detector and TSDD ignore virtual drift on the Waveform dataset. ST and TSDD outperform RDD with one perfect detection on the virtual datasets.

4.4. Overall performances

In Table 7, we produce the combined true positive and true negative results by (1). This table showcases the overall performance achieved by each detector on each dataset. Due to the fact that on the Musk, Hyperplane and the Waveform datasets, drift induction either generates more virtual or real drift, the TN score will have a varying impact.

Table 7 allows us to assess the ability of a model to ignore real drift while detecting real drift. RDD yields the best H scores on 7 RW datasets and tying the first place

Table 4
Least Important Noise Drift: The detection results on virtual drift when noise is added to the least important features. The lower the detection ratio, the better the detectors are. Our model RDD comes first with almost no false detections. ST and TSDD take second and third place with 2 false detections while the other detectors consistently detect virtual drift.

	Adult	Bank	Cov	D08	D17	Elec	*Musk*	Phish.	Spam	Wine	Hyp.	LED	Wav.
RDD	**0.0**	**0.0**	**0.0**	0.4	0.1	**0.0**	*0.0*	**0.0**	**0.0**	0.2	**0.0**	**0.0**	**0.0**
ADWIN	1.0	1.0	1.0	1.0	1.0	1.0	***1.0***	1.0	1.0	1.0	1.0	1.0	1.0
D3	1.0	1.0	1.0	1.0	1.0	1.0	***1.0***	1.0	1.0	1.0	1.0	1.0	1.0
KS	1.0	1.0	1.0	1.0	1.0	1.0	***1.0***	1.0	1.0	1.0	1.0	1.0	1.0
MMD	1.0	1.0	1.0	1.0	1.0	1.0	***1.0***	1.0	1.0	1.0	1.0	1.0	1.0
ST	**0.0**	1.0	0.2	**0.0**	0.1	**0.0**	*0.9*	**0.0**	0.7	**0.0**	**0.0**	**0.0**	**0.0**
TSDD	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	*0.6*	**0.0**	0.7	1.0	**0.0**	0.1	**0.0**

Table 5
Most Important Step Drift: The detection results on real drift when the most important features are shuffled. The highest the detection ratio, the better the models are. MMD takes first place, followed by KS, ADWIN and D3 closely followed by our detector RDD and ST. TSDD outputs false negatives in all but 3 datasets.

	Adult	Bank	Cov	D08	D17	Elec	Musk	Phish.	Spam	Wine	*Hyp.*	LED	Wav.
RDD	0.9	0.8	0.0	0.9	**1.0**	**1.0**	**1.0**	0.5	0.2	**1.0**	***0.0***	0.0	0.0
ADWIN	**1.0**	**1.0**	**1.0**	0.8	**1.0**	0.0	**1.0**	**1.0**	**1.0**	**1.0**	***0.0***	0.2	0.2
D3	**1.0**	**1.0**	0.0	0.8	**1.0**	0.0	**1.0**	**1.0**	**1.0**	**1.0**	***0.0***	0.0	0.0
KS	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	*0.2*	**1.0**	0.9
MMD	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	*0.2*	**1.0**	**1.0**
ST	0.7	0.6	**1.0**	0.1	0.6	0.3	**1.0**	**1.0**	**1.0**	0.0	***0.0***	0.9	0.8
TSDD	0.0	0.0	0.0	0.6	0.0	0.0	0.0	0.0	0.7	0.2	***0.0***	**1.0**	0.0

on 2 synthetic ones. TSDD takes second place with the highest score on 1 RW dataset but coming first or tying first place on all synthetic datasets. ST comes third with the highest H scores on 2 RW datasets and tying first place on 1 synthetic dataset. On RW datasets, we see that ADWIN, D3, KS and MMD have overall low scores due to their misclassification of virtual drift despite having detected all real drifts. On synthetic datasets, their score is better having not made too many misclassification when a step drift was induced on the least informative features.

5. Conclusion

In this paper we introduced RDD, a drift detector that does not need ground truth labels during the inference phase. We extensively challenged our algorithm against a number of state of the art drift detectors and over a large panel of both real and synthetic datasets. We experimentally proved that our method outperforms current drift detection methods. We showed our detector's ability to detect real drift and to ignore virtual drift. As false alarms are the main reason why drift detectors are not

Table 6
Most Important Noise Drift: The detection results on real drift when noise is added to the most important features. The highest the detection ratio, the better the models are. ADWIN, D3, KS and MMD achieve perfect detection across all datasets exhibiting real drift. Our model RDD achieves perfect detection on 7 datasets outperforming TSDD and ST.

	Adult	Bank	Cov	D08	D17	Elec	Musk	Phish.	Spam	Wine	Hyp.	LED	*Wav.*
RDD	**1.0**	0.9	0.0	**1.0**	**1.0**	0.5	**1.0**	**1.0**	**1.0**	**1.0**	0.0	0.5	***0.0***
ADWIN	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	*1.0*
D3	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	*1.0*
KS	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	*1.0*
MMD	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	**1.0**	*1.0*
ST	0.0	**1.0**	**1.0**	0.1	0.0	0.8	0.3	**1.0**	**1.0**	0.0	0.0	**1.0**	*0.9*
TSDD	**1.0**	**1.0**	0.0	0.1	0.4	0.0	0.8	0.0	0.6	**1.0**	0.0	**1.0**	***0.0***

Table 7
Harmonic Mean: The results of table 2 through 5 aggregated in one metric. The higher the metric the best the detector is at both ignoring virtual drift and detecting real drift. We see that our model RDD comes first with the highest score on 7 out of 10 RW datasets and on 2 of the 3 virtual datasets. TSDD comes second and ST comes third, the two models have lower scores than RDD because of some real drift ignored. The ADWIN, D3, KS and MMD detectors don't yield high score because of their inability to ignore virtual drift.

	Adult	Bank	Cov	D08	D17	Elec	Musk	Phish.	Spam	Wine	Hyp.	LED	Wav.
RDD	**0.99**	**0.96**	0.67	0.80	**0.92**	**0.93**	**0.86**	0.93	**0.89**	**0.89**	**0.86**	0.77	**0.86**
ADWIN	0.00	0.00	0.00	0.00	0.00	0.50	0.00	0.00	0.00	0.00	0.71	0.48	0.29
D3	0.60	0.36	0.50	0.09	0.00	0.50	0.00	0.00	0.00	0.00	0.71	0.50	0.29
KS	0.09	0.00	0.60	0.00	0.00	0.60	0.00	0.00	0.00	0.00	0.65	0.00	0.36
MMD	0.00	0.00	0.46	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.59	0.51	0.19
ST	0.81	0.39	**0.92**	0.71	0.75	0.87	0.50	**1.00**	0.73	0.67	**0.86**	0.91	0.71
TSDD	0.86	0.86	0.67	**0.81**	0.75	0.67	0.75	0.67	0.65	0.52	**0.86**	**0.92**	**0.86**

widely used in production. We demonstrated the usability of our detector for real world applications. We tuned the hyper-parameters on 3 datasets not used in the experimental study. We show that they are valid in a wide range of real-world scenarios and that few effort should be made when using the models in production. We also demonstrated the ability of RDD to work in any dimension, having the best detection accuracy on both datasets that had over 100 features.

Future work will consist of further modeling the partition space. Research will also deal on how a drift detector can be initialized in recurrent concept drift scenarios, when no stationary dataset can be used to initialize a detector.

References

[1] A.-K. Reuel, M. Koren, A. Corso, M. J. Kochenderfer, Using adaptive stress testing to identify paths to ethical dilemmas in autonomous systems., in: SafeAI@ AAAI, 2022.

[2] H. Huang, Z. Li, L. Wang, S. Chen, B. Dong, X. Zhou, Feature space singularity for out-of-distribution detection, arXiv preprint arXiv:2011.14654 (2020).

[3] M. G. Kelly, D. J. Hand, N. M. Adams, The impact of changing populations on classifier performance, in: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, 1999, pp. 367–371.

[4] I. A. Nikolov, M. P. Philipsen, J. Liu, J. V. Dueholm, A. S. Johansen, K. Nasrollahi, T. B. Moeslund, Seasons in drift: A long-term thermal imaging dataset for studying concept drift, in: Thirty-fifth Conference on Neural Information Processing Systems, 2021.

[5] A. Suprem, J. Arulraj, C. Pu, J. Ferreira, Odin: Automated drift detection and recovery in video analytics, arXiv preprint arXiv:2009.05440 (2020).

[6] R. Kamoi, K. Kobayashi, Out-of-distribution detection with likelihoods assigned by deep generative models using multimodal prior distributions., in: SafeAI@ AAAI, 2020, pp. 113–116.

[7] R. Elwell, R. Polikar, Incremental learning of concept drift in nonstationary environments, IEEE Transactions on Neural Networks 22 (2011) 1517–1531.

[8] D. Brzezinski, J. Stefanowski, Reacting to different types of concept drift: The accuracy updated ensemble algorithm, IEEE Transactions on Neural Networks and Learning Systems 25 (2013) 81–94.

[9] J. Z. Kolter, M. A. Maloof, Dynamic weighted majority: An ensemble method for drifting concepts, The Journal of Machine Learning Research 8 (2007) 2755–2790.

[10] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, R. Morales-Bueno, Early drift detection method, in: Fourth international workshop on knowledge discovery from data streams, volume 6, 2006, pp. 77–86.

[11] D. R. de Lima Cabral, R. S. M. de Barros, Concept drift detection based on fisher's exact test, Information Sciences 442 (2018) 220–234.

[12] A. Bifet, R. Gavalda, Learning from time-changing data with adaptive windowing, in: Proceedings of the 2007 SIAM international conference on data mining, SIAM, 2007, pp. 443–448.

[13] C. Raab, M. Heusinger, F.-M. Schleif, Reactive soft prototype computing for concept drift streams, Neurocomputing 416 (2020) 340–351.

[14] M. Heusinger, F.-M. Schleif, Reactive concept drift detection using coresets over sliding windows, in: 2020 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2020, pp. 1350–1355.

[15] Ö. Gözüaçık, A. Büyükçakır, H. Bonab, F. Can, Unsupervised concept drift detection with a discriminative classifier, in: Proceedings of the 28th ACM

international conference on information and knowledge management, 2019, pp. 2365–2368.

[16] M. Black, R. Hickey, Learning classification rules for telecom customer call data under concept drift, Soft Computing 8 (2003) 102–108.

[17] V. Cerqueira, H. M. Gomes, A. Bifet, Unsupervised concept drift detection using a student–teacher approach, in: International Conference on Discovery Science, Springer, 2020, pp. 190–204.

[18] S. Rabanser, S. Günnemann, Z. Lipton, Failing loudly: An empirical study of methods for detecting dataset shift, Advances in Neural Information Processing Systems 32 (2019).

[19] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, A. Smola, A kernel two-sample test, The Journal of Machine Learning Research 13 (2012) 723–773.

[20] A. Castellani, S. Schmitt, B. Hammer, Task-sensitive concept drift detector with constraint embedding, in: 2021 IEEE Symposium Series on Computational Intelligence (SSCI), IEEE, 2021, pp. 01–08.

[21] C. C. Serdar, M. Cihan, D. Yücel, M. A. Serdar, Sample size, power and effect size revisited: simplified and practical approaches in pre-clinical, clinical and laboratory studies, Biochemia medica 31 (2021) 27–53.

[22] T. S. Sethi, M. Kantardzic, On the reliable detection of concept drift from streaming unlabeled data, Expert Systems with Applications 82 (2017) 77–99.

[23] T. S. Sethi, M. Kantardzic, E. Arabmakki, Monitoring classification blindspots to detect drifts from unlabeled data, in: 2016 IEEE 17th International Conference on Information Reuse and Integration (IRI), IEEE, 2016, pp. 142–151.

[24] A. Van Looveren, J. Klaise, G. Vacanti, O. Cobb, A. Scillitoe, R. Samoilescu, A. Athorne, Alibi detect: Algorithms for outlier, adversarial and drift detection, 2019. URL: https://github.com/SeldonIO/alibi-detect.