Partially Supervised Classification for Early Concept Drift Detection

Maxime Fuccellaro
Univ. Bordeaux, CNRS
Bordeaux INP, LaBRI, UMR 5800
Talence, France
maxime.fuccellaro@u-bordeaux.fr

Laurent Simon
Univ. Bordeaux, CNRS
Bordeaux INP, LaBRI, UMR 5800
Talence, France
lsimon@labri.fr

Akka Zemmari
Univ. Bordeaux, CNRS
Bordeaux INP, LaBRI, UMR 5800
Talence, France
akka.zemmari@u-bordeaux.fr

***Abstract*—As more and more data is generated and stored, and as longer data streams become available, concept drift detection is becoming crucial for most real world applications. We introduce Partially Supervised Drift Detection, PSDD, a drift detection method based on Decision Trees that does not suppose any knowledge of true class labels during inference. Our approach works in any number of dimensions and is able to distinguish real from virtual drift. We successfully evaluated our method with well established datasets in the drift detection field.**

***Index Terms*—Concept Drift, Drift Detection, Semi Supervised, Data Streams, Real Drift**

I. Introduction

Concept drift [1] occurs when the underlying distribution of a data source changes over time. It may strongly degrade the model's performances and can occur in all classification domains, at various speeds and recurrences [2], which explains why it has been the subject of intensive research in recent years [3]. Sources of concept drifts have been widely studied and sorted as in [3] and [4].

Much attention has been given to drift detection on labelled data. In [5], [6] and [7], the presence of drift is based on a classifier's error rate with the hypothesis that a drop of classification performance is a consequence of drift. These methods are very efficient in detecting type $P(y \mid X)$ drift but require true labels to operate, which may be not realistic for real-world applications. Other algorithms handle drift by design. For instance, [8] introduces an adaptive ensemble algorithm where individual model contribution is weighted based on recent performance or age. In [9], the models of the ensemble are also updated with each new batch. In [10], a new model is added to the ensemble when a drift is identified using a detector. Those ensemble-based methods provide quality classification performances while handling drift, but strongly depend on the presence of labelled data. To avoid using true labels, some drift-detectors track drift in the feature space instead. In [11], the authors compute weight distances of two neural networks built on separate batches of data.

In [12], authors introduce ADWIN, an unsupervised single dimension drift detection algorithm. The model uses a dynamically sized window that shrinks when a drift is detected while increasing when data distribution is stationary. The instances of the window are split into two groups based on their order of appearance, a drift is detected when the difference of the sub-windows values' averages exceeds a user defined threshold. Another one-dimension window-based unsupervised algorithm is KSWIN [13]. KSWIN uses a window of user-defined size n that is split into old and recent samples. The recent samples' window size is user defined and its size cannot exceed $n/2$, r samples are picked at random from the old window. KSWIN then computes the Kolmogorov-Smirnov distance of the empirical cumulative data distribution of the two windows, if that distance exceeds a user defined parameter α dependant threshold, than a drift is detected.

In [14], the authors introduce Discriminative Drift Detector (D3). D3 detects drift of type $P(X)$ based on the assumption that if a model can differentiate old samples from new ones then there is a drift. D3 attempts to find if a drift has occurred between a training set and a test set. Training samples are attributed the class 0 while testing samples class is set to 1. The samples are randomly shuffled and split into a new training set and a new test set. A model is then fitted on the training set. A drift is detected if the AUC metric on the test set exceeds a user defined threshold.

In [15] the authors use a student-teacher approach to detect drift. The teacher model is trained to predicts the true class of a given instance, while the student model is trained to predict the output of the teacher model. The authors' hypothesis is that a drop in performance of the student model signals that a drift has occurred, getting around the unavailabilities of the true class labels. A drift is detected by a drift detection algorithm such has ADWIN on the error rate of the student model. During the training phase in [16], the authors use an encoder to project the input data in a low-dimension feature-space while maximizing inter-class and minimizing intra-class distances. During inference, concept drifts are detected by monitoring distances of new instances to the learned class centroids in the embedding space. In [17], the author introduce a multi-dimension online unsupervised drift detection algorithm based on Minimum Enclosing Balls. During inference, the detector calculates the centroid of a sliding window, if the distance between a new observation and the centroid is greater than a dynamically set threshold a drift is detected, otherwise the window is updated. [18] introduces a time-based split criterion

Mangrove is funding Maxime Fuccellaro's PhD

that partition a feature into two parts that have homogeneous drift characteristics. This is used as the split criterion in a decision tree to partition the data into set that have the same distribution.

Most drift detection work is done in a prequential manner, when true labels are available immediately after inference. We consider this hypothesis to be unrealistic in many real world applications like [15] and [16]. We introduce PSDD, a tree based drift-detector that tracks $P(X \mid y)$ drift type. PSDD is semi-supervised as it doesn't need labels during the detection phase. PSDD creates a Decision Tree of pure leaves that tracks changes within each leaf distribution to detect drift. PSDD works with two user defined parameters that control the drift detection sensibility. Our main contribution is that the model works on unlabeled data in any dimensions, enabling use for real world application drift detection. Some key aspects of concept drift are presented in section 2, the model is detailed in Section 3 and an experimental evaluation is presented in Section 4.

II. Concept Drift

A. Definition

Concept drift occurs when the underlying distribution of a data source changes. Drift can come from different sources, at various speeds, it can be real or virtual. This diversity explains why recent algorithms are designed to detect or deal with specific drifts rather than to achieve high performance in all contexts.

$$P(y \mid X) = \frac{P(y)P(X \mid y)}{P(X)}$$

With Bayes Formula, one can separate three types of drift.

- $P(y \mid X)$: This drift type occurs when the true underlying classification function changes, it is the most studied type of drift as it has a direct impact on a model's performances. As mentioned above, most techniques to detect this type of drift are based on inferences ground-truth labels being accessible to perform detection, this hypothesis is not realistic in real-world predictive pipelines.
- $P(X \mid y)$: This type of drift occurs when data distribution changes conditioned by y values. This distribution change impacts models with previously unseen values for a given class.
- $P(y)$: This drift is characterized by a shift in class labels. Although a fixed trained model will not be impacted, this is challenging when online learning is required.

B. Drift Speed

Drift can appear at various speeds. It can be sudden or abrupt when the concept changes quickly over a short period of time. The covid pandemic gave us many examples of abrupt drift, such as the drop of airline traffic. Drift can be incremental, where the distribution changes slowly over time, global warming creates an incremental drift in weather data. Drift can be recurrent and oscillate between two or more concepts. Drift recurrence can be dealt with by having a pool of instances of detectors, each corresponding to a concept [19], [8]. PSDD aims to detect $P(X \mid y)$ type of drift regardless of speed. However, our experimental results will focus on abrupt drift detection.

C. Virtual Drift vs Real Drift

Virtual drift [4] can be characterized by a change in distribution that does not impact the model's predictive performances. As opposed to real drift which will decrease the model's performances. Detectors that monitor error rate such seen in [5], [20], [21] or [22] bypass this challenge, as decrease in a model's performances is the triggering signal.

III. Unsupervised Class Distribution Drift Detection

A. Notations and Hypothesis

Let $\mathcal{B}^*$ be a batch of labelled data containing m instances across d features.

$\mathcal{B}^* = (X_1, ..., X_d, Y)$ where the $X_i = (x_1^i, ..., x_m^i)$ are feature vectors and $Y = (y_1, ..., y_m)$ is the class vector.

Let $\mathcal{B}$ be a batch of unlabelled data $(X_1, ..., X_d)$

We assume that $\forall i, j \in [1, d]^2$:

$$X_i \sim \mathcal{N}(\mu_i, \sigma_i) \tag{1}$$

$$Cov(X_i, X_j) = 0, \forall i \neq j \tag{2}$$

B. Description of UC3D: training and detection

1) Our detector: A Decision Tree $\mathcal{T}$ is first trained over labelled data B^* (splits are successively made until leaves are pure or that the minimum sample-split is threshold is reached). We don't use impure leaves for the model detection. For each leaf of $\mathcal{T}$, we get the samples $\mathcal{D}_i^*$ from B^* that belong to it and compute the cluster centre of these points using the euclidean distance.

During inference, we distribute the samples of B into $\mathcal{T}$'s leaves, and then compute the centroid $\mathcal{C}_i$ of the unlabeled data points for each leaf. To detect a drift, we then test the null hypothesis of the means.

2) Detector: We build a decision tree $\mathcal{T}$ on $\mathcal{B}^*$ with m pure leaves of high cardinality (> 20). We achieve this by specifying the min sample split size to 20 and by removing post training impure leaves.

Let $\mathcal{D}^*$ be all the labelled instances of $\mathcal{B}^*$ belonging to class c. $\mathcal{D}^* = \{X_1, ..., X_d \mid y = c\} \in \mathcal{B}^*$, $Card(\mathcal{D}^*) = n_c^*$

Similarly, let $\mathcal{D}$ be the instances of $\mathcal{B}$ where the predicted class by $\mathcal{T}$ is c. $\mathcal{D} = \{X_1, ..., X_d \mid \hat{y} = c\} \in \mathcal{B}$, $Card(\mathcal{D}) = n_c$

Let $\mathcal{D}_k^* \in \mathcal{D}^*$ be the set of labelled observations from $\mathcal{B}^*$, of class c belonging to leaf $\mathcal{L}_k$.

Let $\mathcal{D}_k \in \mathcal{D}^*$ be the set of unlabelled observations from $\mathcal{B}$, of estimated class c belonging to leaf $\mathcal{L}_k$.

Let

$$\mu_k^* = \frac{1}{n_k^*}(\sum_{i=1}^{n_k^*} x_1^i, ..., \sum_{i=1}^{n_k^*} x_d^i)$$

and

$$\mu_k = \frac{1}{n_k}(\sum_{i=1}^{n_k} x_1^i, ..., \sum_{i=1}^{n_k} x_d^i)$$

be the euclidean mean vectors of $\mathcal{D}_k^*$ and of $\mathcal{D}_k$.

Because of (1) and (2), for each leaf $\mathcal{D}_k$ we test:

$$\mathcal{H}_0 : \mu_k^* = \mu_k$$

with risk α.

Let $\mathcal{R}$ be the number of leaves where the null hypothesis is rejected.

If

$$\frac{\mathcal{R}}{m} > \beta$$

then there is a drift. Both α and β are hyper-parameters. A high β value means more leaves are require to show a change in distribution.

To detect a shift in variance we could proceed the exact same way with the null hypothesis $\mathcal{H}_0 : \sigma_k^* = \sigma_k$. In this paper, we implemented and tested the shift in mean. The variance shift detector will be the focus of future work.

IV. Experimental Study

Firstly, we make an overview of data and detector used ; secondly, we introduce the benchmarking procedure and, lastly, we present our experimental results. We challenge our approach on six well established real-world datasets. All of the dataset have been processed to have only binary and numeric attributes; numeric attributes have been normalized and standardized.

A. Datasets and Comparative Methods

We will conduct our benchmark on eight widely used datasets [12], [16].

CoverTypes (Cov) [20]: Each entry describes a 30 x 30 meter cell of a forest. By sorting the cells by increasing altitude, we create a gradual drift as the main tree essence and forest characteristics change with altitude. The classification goal is to predict the most common tree type.

Elec [5]: The ELEC dataset describes electricity demand in 5 fields. Drift exact occurrence is unknown. The goal is to predict whether the price is going up or down in the next half hour.

Adult [23]: The goal is to predict if a person's salary is over 50K a year based on a series of attributes.

Musk: The goal is to determine if a molecule is a musk.

Wine: The goal is to predict where a wine comes from.

Spam: The goal is to classify if an email is spam or genuine.

Bank: The goal is to predict if the client will subscribe a term deposit.

Phishing: The goal is to predict if a web page is malicious or not.

Both the Elec and CoverTypes dataset have been largely used in drift detection benchmarks [5], [7], [12], all other datasets have been used in [16] and [24].

Table I describes the datasets characteristics.

We test PSDD[1] with hyperparameters $\alpha = 0.01$ and $\beta = 0.01$ against:

[1]The source code is available on: https://github.com/mfuccellaro/psdd

TABLE I
Dataset Overview

Dataset	# of instances	# of features	Classification
Cov	110393	50	Multi-class (7)
Elec	45312	13	Binary
Adult	48842	65	Binary
Musk	6598	166	Binary
Wine	6497	12	Binary
Spam	6213	499	Binary
Bank	45211	48	Binary
Phishing	11055	46	Binary

- Discriminative Drift Detector (D3) [14] with the default threshold of 0.75.
- Task-Sensitive Drift Detector (TSDD) [16] coupled with the author's default architecture and their modified z-score as a detector.
- Student-Teacher (ST) Detector [15] with standard decision trees coupled with ADWIN [12].
- ADWIN [12] with $\delta = 0.007$.
- KSWIN [13] with default $\alpha = 0.005$.

All experiments have been run ten times to calculate the different metrics (means and standard deviation). Hyper-parameter tuning has been done to tune ADWIN's delta parameter as well as KSWIN window size. All other algorithms are used with default settings.

B. Experiment Setup

When considering concept drift as a supervised problem, the accuracy of a base learner is often used to benchmark the pertinence of drift detection methods. The accuracy of a base learner can be necessary to detect drift as in [5], [6] [7]. When methods don't use labelled data for detection or when detection is done in a prequential manner [25], [26] the accuracy is often used to compare the drift detection methods [13]. Even if prequential evaluation doesn't use true labels for detection, it implies constant retraining of the detector and needs labels for evaluation, which we don't consider realistic in a real world-scenario.

In order to test our detector against the state of the art drift-detectors, we induce artificial drift into the datasets. This is a standard drift detection benchmark [16], [24] as it allows us to know exactly where the drift is taking place and to control its amplitude. This method allows us to test without constant retraining while disregarding true labels completely. In this benchmark we compare detectors' abilities to detect the presence of drift, to avoid false positive and we look at the detection time as in the number of drift-laced samples necessary to trigger a detection.

1) Drift Induction: We start by randomly shuffling the dataset to remove any existing drifting behavior. We then randomly split the dataset threefold. 50% of the observations into the training set, 25% into a test set and 25% to the validation set. Drift is only induced in the validation set in a homogeneous manner with regards to observations class as detailed hereafter. The training set is used by detectors to learn

the initial data distribution while the test set is used with the validation set to accurately compute detection times. Both the training and test set share the initial unmodified distribution.

We induce two different types of drift into the validation set:

- Drift by adding a Gaussian noise $\mathcal{N}(1, 1)$ to a subset of features.
- Drift by shuffling the values of a subset of features.

The features in which drift is induced are selected by a trained decision tree over the training set, the model's feature importance list is used to rank features based on their predictive impact. We collect two sets of features, the top 25% most informative features and the bottom 25% least important features.

Each dataset will be used four times to test the different drift configurations: *(least, noise), (least, shuffle), (most, noise), (most, shuffle)*.

2) False Positives: While decreasing trust in the detector, False Positives (FP) can have various negative impacts on a predictive pipeline. For instance, they might trigger superfluous retraining procedures. Some of our benchmark configurations may lead to a virtual drift, that is when the distribution change does not impact predictive performances, as opposed to real drift, when the change has a direct impact on performances. An optimal drift-detector will only trigger an alarm for real drifts. In order to distinguish real drift from virtual drift, we compute the accuracy score over the training set, the validation set and the four modified test set. When the condition $ACC_{test} < ACC_{val} - |ACC_{train} - ACC_{val}|$ is true, we are in presence of a real drift. Mean classification accuracy of a standard decision tree model over the 6 different sets is presented in Table II. In Table II, real drift scenarios are highlighted. As expected, all but one virtual drift takes place when the least informative features are modified, no matter the drift induction process.

TABLE II
VIRTUAL VS REAL DRIFT

Data	Train	Val	Least Important		Most Important	
			Noise	Shuffle	Noise	Shuffle
Cov	.99±.0	.77±.0	.64±.02	.75±.0	**.31±.02**	**.32±.03**
Elec	1.0±.0	.87±.01	.74±.01	.79±.01	**.54±.01**	**.6±.05**
Adult	1.0±.0	.81±.0	.71±.05	.8±.01	**.39±.06**	.63±.11
Musk	1.0±.0	.97±.01	.97±.01	.97±.01	**.51±.02**	**.49±.04**
Wine	1.0±.0	1.0±.0	1.0±.0	1.0±.0	**.62±.01**	**.37±.02**
Spam	1.0±.0	.95±.01	.95±.01	.95±.01	**.49±.05**	**.51±.07**
Bank	1.0±.0	.93±.0	**.75±.03**	.88±.03	**.5±.01**	**.5±.04**
Phishing	.99±.0	.95±.0	**.88±.03**	.92±.02	**.61±.01**	**.47±.14**

3) Detection Delay: We define the detection delay as the minimum number of drift-laced observations needed by a detector to trigger an alarm. If that number of observations is greater than the validation test size, we consider the detector to have missed the drift, and increment a False Negative counter.

In order to compute the detection delay we proceed in two ways. As D3, TSDD, ST as well as PSDD allow for drift detection in batch mode, detectors are first trained on the training set. A drift batch is created by concatenating a random sample of the test and validation set together, starting with 10% of drift laced observations. While no drift is detected, the proportion of the validation set in the drift batch is increased by another 10%. When the drift set is a subset of the validation set and no drift is detected, then the detector failed to detect a drift and the run is marked as a false negative.

As ADWIN and KSWIN are able to detect drift in one dimension, we proceed in a slightly different fashion. For each feature of a given dataset, we create a new instance of the detectors and feed it the observation of that said feature. We then give the detector the data from the drift laced validation dataset. When a drift is detected, we move on to the next feature. The detection delay kept is the smallest number of observations needed by the detector to flag a drift across all features. If no detection is triggered on the dataset the detector has not detected a drift.

C. Experimental Results

In this experiment, we look at three metrics that are essential in order to assess the quality of a detector in a meaningful way: False Positive rates, Detection Delays for the detection speed as well as False Negative rates for missed detections.

1) False Positive Detection: We consider a False positive (FP) when a drift is detected for cases of virtual drifts. Table III and IV report FP ratios across all ten runs. Overall, the detectors have more trouble ignoring Gaussian noise than feature shuffling.

When Gaussian noise was added to the least informative features, the Bank and Phishing datasets are excluded because of virtual drift. D3 raises 100% FP alarms across all datasets. KSWIN also detects a drift across all datasets, although not systematically on the Elec and Adult datasets. PSDD underperformed the ST detector with four consistent FP vs two for ST. Only the musk dataset, triggered a significant number of FP across all detectors but ADWIN. ADWIN outperforms all methods by having no FP on the Musk and Wine datasets. PSDD came fourth after ADWIN, TSDD and ST when adding noise on the least important features.

When looking at the shuffle of least important features, we take all datasets into account. Both TSDD and PSDD almost never trigger FP with the exception on the spam dataset where PSDD raised a false alarm once and on the CoverTypes dataset for TSDD. KSWIN, D3 and the Student-Teacher detector consistently raised FP for several datasets, all three detectors consistently raised FP alarms on the Phishing dataset. PSDD came first followed by TSDD and ADWIN on FP detection on least important feature shuffling.

When looking at overall results, TSDD has the least FP with 19%, followed by ADWIN 24%, then by PSDD with 29%, ST has 39%. Both KSWIN and D3 have shown the inability to properly distinguish real drift from virtual drift with an overall of 75% and 79% FP rates.

2) False Negatives: A False Negative (FN) is detected when failure to detect a real drift. FN results are reported in Table V and VI.

TABLE III
FALSE POSITIVE RATES

Data	Least Imp. Noise					
	D3	PSDD	ST	TSDD	ADWIN	KSWIN
Cov	1.0	1.0	1.0	0.0	0.7	1.0
Elec	1.0	0.0	0.1	0.0	0.4	0.4
Adult	1.0	1.0	0.0	0.0	0.5	0.8
Musk	1.0	1.0	1.0	0.7	0.0	1.0
Wine	1.0	0.0	0.0	1.0	0.0	1.0
Spam	1.0	1.0	0.2	0.4	0.1	1.0
Bank	/	/	/	/	/	/
Phishing	/	/	/	/	/	/
Total	100%	66%	38%	37%	28%	87%

TABLE IV
FALSE POSITIVE RATES

Data	Least Imp. Shuffle					
	D3	PSDD	ST	TSDD	ADWIN	KSWIN
Cov	0.0	0.0	1.0	0.4	0.9	1.0
Elec	0.0	0.0	0.1	0.0	0.5	0.1
Adult	0.1	0.0	0.0	0.0	0.0	0.0
Musk	1.0	0.0	0.6	0.0	0.0	1.0
Wine	1.0	0.0	0.0	0.0	0.0	1.0
Spam	1.0	0.1	0.0	0.0	0.1	1.0
Bank	1.0	0.0	0.5	0.0	0.2	0.2
Phishing	1.0	0.0	1.0	0.0	0.0	1.0
Total	64%	1%	40%	5%	21%	66%

When shuffling the most important features we don't take into account the adult dataset as the drift is virtual. PSDD, TSDD and ADWIN consistently fail to detect drifts on most of the datasets with less than 20% of True Positives. PSDD consistently detected all drifts on the spam dataset along with TSDD. KSWIN exhibits FN on the Bank and Elec dataset, while ST only fails to detect drift on the Wine dataset. Finally D3 detects all real drifts except for the Elec and CoverTypes datasets.

When adding noise to the most important features, PSDD yields similar detection rate as TSDD with four meaningful detections at of eight, our model fails to detect drift on the Wine dataset, while TSDD only partially detects drift on the Spam dataset. ST does slightly better with five meaningful detections while D3 detects every single drift. ADWIN consistently fails to detect drift across all datasets with 72% FN rate, but manages to find drifts in some runs. KSWIN on the other hand has a very low FN rate of 16%.

When we look at the adding of noise on the least important features of the Bank and Phishing datasets, PSDD comes first with no FN, followed by KSWIN with 10% FN. D3 and ST detect drift on one of the dataset each. TSDD along with ADWIN completely fail to detect drift here.

We clearly see that PSDD, with an overall TP rate of 44%, detects less drifts than D3 (83%), ST (70%) and KSWIN (81%). However, our method outperforms TSDD and ADWIN that have a TP rate of 34% and 25%.

3) Detection Delays: Detection delays results are reported in Table VII. The hyphen indicates that a detector fails to detect the drift across all runs while the forward slash indicates that the drift is virtual and has not be taken into account. When the standard deviation is not shown, it means that only one detection was made. The two fastest detections are marked in bold. Drift detection delays widely depends on the dataset's number of samples.

TABLE V
FALSE NEGATIVE RATES

Data	Most Imp. Shuffle					
	D3	PSDD	ST	TSDD	ADWIN	KSWIN
Cov	1.0	1.0	0.0	1.0	0.5	0.0
Elec	1.0	1.0	0.4	1.0	0.7	0.8
Adult	/	/	/	/	/	/
Musk	0.0	0.9	0.0	1.0	1.0	0.0
Wine	0.0	1.0	1.0	0.9	1.0	0.0
Spam	0.0	0.0	0.0	0.1	0.8	0.0
Bank	0.0	1.0	0.1	0.7	0.7	1.0
Phishing	0.0	1.0	0.0	1.0	1.0	0.0
Total	28%	84%	21%	81%	81%	25%

TABLE VI
FALSE NEGATIVE RATES

Data	Most Imp. Noise					
	D3	PSDD	ST	TSDD	ADWIN	KSWIN
Cov	0.0	0.8	0.0	1.0	0.6	0.0
Elec	0.0	1.0	0.1	1.0	0.9	0.6
Adult	0.0	0.0	0.9	0.0	0.6	0.6
Musk	0.0	0.0	0.8	0.1	0.8	0.0
Wine	0.0	1.0	1.0	0.0	1.0	0.0
Spam	0.0	0.0	0.0	0.5	0.8	0.0
Bank	0.0	0.0	0.0	0.1	0.4	0.1
Phishing	0.0	0.9	0.0	1.0	0.8	0.0
Total	0%	46%	35%	46%	72%	16%
	Least Imp. Noise					
	D3	PSDD	ST	TSDD	ADWIN	KSWIN
Bank	1.0	0.0	0.0	1.0	0.7	0.2
Phishing	0.0	0.0	0.8	1.0	0.7	0.0
Total	50%	0%	40%	100%	70%	10%

When shuffling the most important features, only the adult dataset exhibits virtual drift. ST generally detects drift the fastest on four datasets and second fastest on one. KSWIN comes in second place on three datasets. PSDD detects drift the fastest on the Spam dataset, and third on the Musk dataset. ADWIN comes first once on the Elec dataset and second on the CoverTypes dataset. On several dataset, a high False Negative rate prevent us from accurately comparing detection times. When shuffling features, drift seems harder to detect for all detectors, this may be due to the high count of binary features in the data.

When adding noise to the most important features, our model is the first to detect drift on the Musk and Spam datasets, second on the Adult dataset. D3 has the lowest FN rates, but is the second slowest to detect drift in five datasets and slowest on Wine. ST is still the fastest to detect drift, coming first on five datasets. TSDD is the fastest to detect drift on 2 datasets.

On the Spam and Musk datasets, which has the highest number of features, our model is fastest twice and second fastest once to flag drift, this demonstrates are model ability to work efficiently in a high dimension context. When adding noise to the least important feature of the Bank and Phishing

dataset, PSDD comes third and last. KSWIN is the fastest detector in this context.

However, methods average detection delays exhibits great difference. On several datasets the difference is too large to be comparable. We showed that PSDD's detection delays aren't the fastest, but are within the same range of the state of the art.

4) Experiment learnings: Let's note that without careful configuration, both window sized drift-detectors ADWIN and KSWIN performances dropped very significantly. Due to the fact that both ADWIN and KSWIN only accept time series, the detection time linearly slowed down with the number of dimensions in the dataset in our experiment context. When combining False Negative, False Positive and Detection Delay results, we clearly see that D3 slowly detects drifts on most of the datasets, but suffers a very high False Positive rate like KSWIN, which makes them hard to use in a real-world' scenario as they cannot differentiate between real and virtual drift. TSDD has the lowest FP rates followed by ADWIN and PSDD. All of those algorithm showcase good detection times. But, this is counterbalanced by a high FN rate. The student teacher model has a lower FP rate than D3, is faster to detect a shift, but has a moderately high FN rate. Overall PSDD manages to detect real drift and ignore virtual drift in a wide array of datasets. This experiment demonstrated the viability of PSDD as a drift-detector that is fast and won't trigger any false alarms.

TABLE VII
DETECTION DELAY

		Most Important	
Dataset	Model	Noise	Shuffle
Cov	D3	14542±70	-
	PSDD	20831±3902	-
	ST	**924±174**	**2442±2420**
	TSDD	-	-
	ADWIN	**1904**	**6856±3447**
	KSWIN	-	-
Elec	D3	7101±99	-
	PSDD	-	-
	ST	**6050±3220**	**3360±1096**
	TSDD	-	-
	ADWIN	7167	**2868±3285**
	KSWIN	**6932±2417**	8267±1316
Adult	D3	6150±68	/
	PSDD	**1343±386**	/
	ST	12222	/
	TSDD	**1221**	/
	ADWIN	5759±4633	/
	KSWIN	6049±2044	/
Bank	D3	5706±63	**5796±97**
	PSDD	6328±3774	-
	ST	**882±406**	**2210±2365**
	TSDD	**1130±0**	-
	ADWIN	5743±3878	6975±2201
	KSWIN	6297±2276	-
Musk	D3	823±23	841±19
	PSDD	**164**	492
	ST	514±267	**328±119**
	TSDD	**292±383**	-
	ADWIN	959±452	-
	KSWIN	471	**471**
Phishing	D3	1382±36	1802±175
	PSDD	2484±0	-
	ST	**585±387**	**620±163**
	TSDD	-	-
	ADWIN	1631±1176	-
	KSWIN	**790±2**	**860±63**
Wine	D3	905±23	899±16
	PSDD	-	-
	ST	-	-
	TSDD	**162**	**162**
	ADWIN	-	-
	KSWIN	**467±3**	**471±12**
Spam	D3	786±18	793±14
	PSDD	**264±164**	**232±110**
	ST	**130±25**	571±536
	TSDD	961±507	**258±110**
	ADWIN	1087±316	1103±475
	KSWIN	443	466±25
		Least Important	
Bank	D3	5661±79	/
	PSDD	3842±1326	/
	ST	**2529±807**	/
	TSDD	-	/
	ADWIN	4174±226	/
	KSWIN	**1228±613**	/
Phishing	D3	1384±26	/
	PSDD	2263±218	/
	ST	1221±762	/
	TSDD	-	/
	ADWIN	**353±45**	/
	KSWIN	**424±3**	/

V. CONCLUSION

We introduced PSDD, a drift detection method that will not need labels during inference making it a good alternative for real-world applications. We provided mathematical proof that it can detect real drifts, given that a decision tree is able to produce some pure leaves. We empirically showed that PSDD ignores virtual drift due to the structure of the decision tree algorithm. We demonstrated through our experiment that our model is as good as the state of the art. It shows a good compromise between False Positives, False Negatives and Detection Time. We also demonstrated it's ability to work in any dimension. We showed that there is still room for improvement when it comes to detecting drift in an unsupervised environment.

Future work will consist in adapting the model with new tests to reject or accept the null hypothesis of equal mean in tree leaves. We will focus on building a detector that also monitors a shift in variance that might decrease the False Negative rates that PSDD currently suffers from. Some other way of partitioning the data into pure segment will be investigated. Finally we will be working on incorporating a drift-detector such as ADWIN to monitor each leaf tests' results during inference.

REFERENCES

[1] M. G. Kelly, D. J. Hand, and 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.

[2] I. Žliobaitė, "Learning under concept drift: an overview," *arXiv preprint arXiv:1010.4784*, 2010.

[3] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data*

Engineering, vol. 31, no. 12, pp. 2346–2363, 2018.

[4] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, pp. 1–37, 2014.

[5] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Brazilian symposium on artificial intelligence*. Springer, 2004, pp. 286–295.

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

[7] D. R. de Lima Cabral and R. S. M. de Barros, "Concept drift detection based on fisher's exact test," *Information Sciences*, vol. 442, pp. 220–234, 2018.

[8] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517–1531, 2011.

[9] D. Brzezinski and J. Stefanowski, "Reacting to different types of concept drift: The accuracy updated ensemble algorithm," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 81–94, 2013.

[10] P. Zhao, L.-W. Cai, and Z.-H. Zhou, "Handling concept drift via model reuse," *Machine Learning*, vol. 109, no. 3, pp. 533–568, 2020.

[11] Z. Yang, S. Al-Dahidi, P. Baraldi, E. Zio, and L. Montelatici, "A novel concept drift detection method for incremental learning in nonstationary environments," *IEEE transactions on neural networks and learning systems*, vol. 31, no. 1, pp. 309–320, 2019.

[12] A. Bifet and 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, and F.-M. Schleif, "Reactive soft prototype computing for concept drift streams," *Neurocomputing*, vol. 416, pp. 340–351, 2020.

[14] Ö. Gözüaçık, A. Büyükçakır, H. Bonab, and 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.

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

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

[17] M. Heusinger and 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.

[18] F. Hinder, B. Hammer, and M. Verleysen, "Concept drift segmentation via kolmogorov-trees." in *ESANN*, 2021.

[19] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Tracking recurring contexts using ensemble classifiers: an application to email filtering," *Knowledge and Information Systems*, vol. 22, no. 3, pp. 371–391, 2010.

[20] I. Frias-Blanco, J. del Campo-Ávila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Diaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on hoeffding's bounds," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 3, pp. 810–823, 2014.

[21] S. Yu, Z. Abraham, H. Wang, M. Shah, Y. Wei, and J. C. Príncipe, "Concept drift detection and adaptation with hierarchical hypothesis testing," *Journal of the Franklin Institute*, vol. 356, no. 5, pp. 3187–3215, 2019.

[22] R. S. M. de Barros, J. I. G. Hidalgo, and D. R. de Lima Cabral, "Wilcoxon rank sum test drift detector," *Neurocomputing*, vol. 275, pp. 1954–1963, 2018.

[23] R. Kohavi *et al.*, "Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid." in *Kdd*, vol. 96, 1996, pp. 202–207.

[24] T. S. Sethi, M. Kantardzic, and 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.

[25] Y. Sun, K. Tang, Z. Zhu, and X. Yao, "Concept drift adaptation by exploiting historical knowledge," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 10, pp. 4822–4832, 2018.

[26] M.-R. Bouguelia, S. Nowaczyk, and A. H. Payberah, "An adaptive algorithm for anomaly and novelty detection in evolving data streams," *Data mining and knowledge discovery*, vol. 32, no. 6, pp. 1597–1633, 2018.