# Diagnosis for Post Concept Drift Decision Trees Repair

**Shaked Almog, Meir Kalech**
Ben Gurion University
shakedal@post.bgu.ac.il, kalech@bgu.ac.il

**Abstract**

Decision trees are commonly used in machine learning since they are accurate and robust classifiers. After a decision tree is built, the data can change over time, causing the classification performance to decrease. This data distribution change is a known challenge in machine learning, referred to as *concept drift*. Once a concept drift has been detected, usually by experiencing a decrease in the model's performance, it can be handled by training a new model. However, this method does not explain the drift harming the performance but only handles the drift's effects. The main contribution of this paper presents a novel two-step approach called APPETITE, which applies diagnosis techniques to identify the feature that has drifted and then adjusts the model accordingly. For the diagnosis step, we present two algorithms. We experimented on 73 known datasets from the literature and semi-synthesized drifts in their features. Both algorithms are better at handling concept drift than training a new model based on the samples after the drift. Combining the two algorithms can provide an explanation of the drift and is a competitive model against a new model trained on the entire data from before and after the drift.

## 1 Introduction

Decision trees and other tree-based models are commonly used in machine learning and preferred for several tasks (Rokach and Maimon 2005). They are accurate and robust classifiers that have been shown to outperform deep learning techniques, especially in environments where the data is presented in the tabular form (Gao et al. 2021; Lee, Cheang, and Moslehpour 2022; Rungskunroch, Jack, and Kaewunruen 2021). Moreover, these models are easily explainable since they can be translated into a set of if-else rules that specify how the model makes decisions (Došilović, Brčić, and Hlupić 2018).

After a decision tree is built, the data samples used to train it may change over time, decreasing the classification performance. This change in the data distribution over time is one of the challenges in machine learning, commonly referred to as *concept drift* (Gepperth and Hammer 2016). The drift can cause instances being incorrectly classified, known as misclassification, and may impact the classification performance until the model is changed and re-adapted to fit the new data distribution.

Using incremental learning can ensure that the model will adapt to the current concept during the learning process. Even though it is frequently used on SVM (Klinkenberg and Joachims 2000), Random Forest (Saffari et al. 2009; Gomes et al. 2017; Zhukov, Sidorov, and Foley 2016; Xie, Peng, and Wang 2016) and other ensemble methods (Castro-Cabrera et al. 2021; Li et al. 2020; Scholz and Klinkenberg 2005; Kolter and Maloof 2007; Wang et al. 2003), there is no equivalent approach for a decision tree to overcome the concept drift effects. Additionally, because incremental learning dynamically updates the model, using it in cases where concept drifts do not frequently occur in the data may be highly inefficient. Therefore, when the data is expected to be mainly drawn from the same distribution, *static models* are often used (e.g., decision trees). In these cases, sudden and unexpected drifts may occur, requiring a different approach to handle the concept drift.

The method to address concept drifts in static models is by training a new model (Singhal, Chawla, and Shorey 2020) that will fit the current concept. The training can be done by using the data after the drift; however, sometimes, there are not enough samples from the new distribution for the adaptation when the drift is detected. Therefore, another way is to combine data from before and after the drift and train a new model. Still, it can be limited because the samples used to train the original model are often not saved due to memory considerations.

A further drawback of both methods mentioned above is that they do not explain the drift itself. In light of the rise in explainability research and the increasing significance of its application in classification models, understanding the drift that caused the misclassification becomes necessary rather than merely adjusting the tree. In this study, **we suggest a novel approach, called APPETITE**, to diagnose faulty nodes in the decision tree model that are most likely to be the ones that caused the misclassification. Unlike the previous ones, this approach is not incremental, it does not train a new model and hence does not require many samples to update the model, and it returns an explanation of the drift.

**The contributions of our paper** are: (1) defining the problem of post-concept drift decision tree repair in terms of a diagnosis problem, (2) presenting novel diagnosis techniques to address the concept drift problem, which aims to identify the faulty node that caused the problem, and (3) suggesting an algorithm to adjust the model accordingly. The code for the algorithms and the experiments is available and can be found in GitHub[1].

We propose two diagnosis algorithms to address the decision tree diagnosis problem; both attempt to identify the faulty node that led to the tree misclassifying new instances. The first algorithm is based on Spectrum Fault Localization (SFL) diagnosis (SFL-DT), originally designed to handle automatic software debugging. The second technique uses statistical analysis to determine which node has the most considerable drift by passing instances across each node (STAT-AN). To adjust the decision tree, we suggest modifying the constraint in the diagnostic node of the decision tree such that it more accurately reflects the distribution of the instances that have passed through it.

We ran experiments on 73 known databases from the literature and semi-synthesized drifts in specific features. The results show that both diagnosis algorithms outperform a competitive algorithm that builds a new model from the data following the drift. Additionally, combining these two algorithms performs significantly better than each separately both in terms of model performance and diagnosis quality; moreover, it succeeds in identifying the drifted features correctly. The combined algorithm is even competitive with a new model that was trained using the entire data from before and after the drift.

The rest of the paper is organized as follows: Section 2 covers the SFL background required for understanding our algorithm. Additionally, we present related work in the fields of Concept Drift and Formal Explainability. Section 3 defines the problem, and Section 4 describes the methodology. Section 5 provides the design of our experiments and presents the results, and Section 6 concludes.

## 2 Background and Related Work

This section will present some background to SFL, which we use as a basis for our first diagnosis algorithm (SFL-DT). Then, we present related work on two topics: (1) concept drift and (2) formal explainability.

### 2.1 SFL Background

Our work is inspired by Spectrum-Based Fault Localization (SFL), a popular approach for diagnosing software systems, aiming to detect bugs in the code based on system tests. We provide a brief background on SFL; for a more comprehensive background on SFL, see (Abreu, Zoeteweij, and Van Gemund 2009).

**Definition 1** (SFL Problem)**.** *An SFL problem is defined by a tuple $\langle COMP, T, SPCT, E\rangle$ where $COMP$ is a set of system components (e.g., functions), each of which may be faulty; $T$ is a set of system tests; $SPCT$ is a binary $A \times B$ matrix where $SPCT_{i,j} = 1$ denotes that component $comp_j$ participated in test $t_i$; and $E$ is a vector of length $A$ where $E_i = 1$ denotes that the $i^{th}$ test's run has failed, and $E_i = 0$ otherwise. An SFL problem arises when $\exists i : E_i = 1$.*

[1]https://github.com/shakedal/decision_tree_diagnosis

The matrix $SPCT$ is called a *spectrum*, and the vector $E$ is called the *error vector*. $SPCT$ and $E$ can be combined into a single table, with $E$ as the table's rightmost column. An example shows this in Table 1a. The table shows the spectrum and error vector for a system with three components ($c_1$,$c_2$,$c_3$) executed through 4 tests ($t_1$,$t_2$,$t_3$,$t_4$). For example, in the second row, components $c_2$ and $c_3$ were involved in the test, and the system test's run has failed.

A solution to an SFL problem is a set of *diagnoses* and a *ranking function* to prioritize them. A diagnosis in SFL is a set of components that *explain* the failed tests. Some SFL algorithms are designed to diagnose problems with a single faulty component, while others are designed to diagnose problems with multiple faults.

Finding diagnoses in the single-fault case is as follows: a component $comp_j$ is a diagnosis if it participated in at least one failed test. To rank these diagnoses, single-fault SFL algorithms use **Similarity Coefficients** (Hofer et al. 2015) between the spectrum and the error vector to rank the diagnoses. These similarity coefficients are formulas evaluated using four **Similarity Counters** $n_{pq}(j),\ p,q \in \{0,1\}$ defined as:

$$\forall comp_j, n_{pq}(j) = |\{i|SPCT_{ij} = p \wedge E_i = q\}| \quad (1)$$

Table 1b shows these counters with respect to the spectrum in Table 1a. We can see, for example, that $n_{11}(2) = 2$ since the number of test runs for which $SPCT_{i,2} = 1 \wedge E_i = 1$ is 2, meaning that component $c_2$ participated in two failed test.

| | $c_1$ | $c_2$ | $c_3$ | $E$ |
|---|---|---|---|---|
| $t_1$ | 1 | 1 | 0 | 1 |
| $t_2$ | 0 | 1 | 1 | 1 |
| $t_3$ | 1 | 0 | 0 | 1 |
| $t_4$ | 1 | 0 | 1 | 0 |

(a)

| | $c_1$ | $c_2$ | $c_3$ |
|---|---|---|---|
| $n_{11}$ | 2 | 2 | 1 |
| $n_{10}$ | 1 | 0 | 1 |
| $n_{01}$ | 1 | 1 | 2 |
| $n_{00}$ | 0 | 1 | 0 |

(b)

Table 1: (a) Spectrum and Error Vector describing a system of 3 components that had run 4 tests, and (b) the corresponding similarity counters.

SFL algorithms consider various similarity coefficients between the observed execution traces and the error vector, like **Ochiai** (Abreu, Zoeteweij, and Van Gemund 2007), and **Tarantula** (Jones and Harrold 2005) to give scores to diagnoses involving single components. In this work, we use a similarity coefficient called **Faith** (Faith 1983) that will be described in Section 4.

### 2.2 Concept Drift

One of the challenges in machine learning is that the data samples used to build the model can change over time. *Concept drift* is the term used to describe how the distribution of data varies over time. Concept drift can be gradual or abrupt; the latter case is sudden and can cause severe immediate effects on the model's performance. There are two types of changes that can be distinguished: changes merely in the input distribution ($p(x)$), which is known as *virtual* *concept drift* or covariate shift, while a change in the underlying functionality itself $(p(y|x))$ is known as *real concept drift* (Gepperth and Hammer 2016). Real concept drift is challenging since it leads to misclassification, for example, the sudden and unexpected impact of Covid-19 on various aspects such as medicine (Duckworth et al. 2021) and finance (Bholat, Gharbawi, and Thew 2020). A study on energy demand (Gomez-Omella, Esnaola-Gonzalez, and Ferreiro 2020) in Spain has shown that during the first weeks of the pandemic the electric consumption data distribution changed and the data variability decreased. One of the main features that were used to predict the demand was the weekday, but due to the pandemic the electric consumption on the weekdays became similar, leading the forecasting models to work unexpectedly. The drift impacts the classification performance, and immediate adaptation in accordance with the drift is required.

There exist different techniques to address concept drift, such as passive and active methods. One way to actively respond after detecting a concept drift is to create a new model based on the new data after the drift has occurred. This method is mainly used when a static model is being used due to the anticipation that the data will not frequently change. Singhal, Chawla, and Shorey (2020) presented an example of such work. Although their work is specifically designed for malicious URL detection, training a new model is a valid way to ensure that we have a model that fits the new data in other domains as well. One of the drawbacks is that we often *do not have enough data* for training a good model, as we show in the experiments. Passive methods can also be used, in which we smoothly adapt the model's parameters so that the model reliably represents the new distribution. One way to do so is to use incremental learning or online learning techniques. A known example is the incremental version of SVM introduced by (Klinkenberg and Joachims 2000), which propose a new method to recognize and handle concept changes with SVM. Another technique used to overcome the concept drift problem is using an incremental version of ensemble learning, which allows us to dynamically add or remove weak learners during the run (Castro-Cabrera et al. 2021; Scholz and Klinkenberg 2005), and also by re-weighting their voting (Li et al. 2020; Kolter and Maloof 2007; Wang et al. 2003). Because incremental learning dynamically updates the model, these methods are time and resources-consuming, and using them in cases where concept drifts do not frequently occur in the data may be highly inefficient.

One of the well-known ensemble learning algorithms is Random Forest, a classification algorithm that consists of many individual decision trees that operate as an ensemble. Several works overcome concept drift specifically by using Random Forest variants. The adaptations of Random Forest are done by several methods, such as re-sampling the data (Gomes et al. 2017), creating and removing trees from the forest (Saffari et al. 2009; Zhukov, Sidorov, and Foley 2016; Xie, Peng, and Wang 2016), re-weighting trees' votes (Zhukov, Sidorov, and Foley 2016), and updating leaves (Xie, Peng, and Wang 2016). Excluding (Xie, Peng, and Wang 2016), which updates the leaves, none of the others changes the tree's structure. Our approach proposes a way to adjust the tree nodes based on the new distribution and is not limited to leaves only but can fit any node type.

There exist dynamic tree algorithms designed for streaming, such as Hoeffding Tree (Domingos and Hulten 2000) that uses the Hoeffding bound (Hoeffding 1963) to determine, with high probability, the smallest number of examples needed at a node when selecting a splitting attribute. Another dynamic tree suitable for a datastream is the Mondrian Tree (Lakshminarayanan, Roy, and Teh 2014) that uses the Mondrian Process (Roy and Teh 2008) to divide the space into blocks. A Mondrian Tree is used in a forest and not as a standalone single tree. Both algorithms assume that the distribution that generates the examples does not change over time; therefore, they are **not** suitable for concept drift handling. Several works created variations of these trees or used them as a forest to overcome concept drifts (Hulten, Spencer, and Domingos 2001; Hoeglinger and Pears 2007; Rad and Haeri 2019; Khannouz and Glatard 2022). Still, they are all incremental methods requiring constant learning and are unsuitable for cases where unexpected drift occurs.

### 2.3 Formal Explainability

Machine Learning algorithms are vastly used in several domains but are mostly perceived as black boxes. Insights about the model's decision-making process are important for us as humans to trust the models, especially in highly sensitive areas such as healthcare or finances (Burkart and Huber 2021). For these reasons, explainability is a rising research field. Other than the need to explain the reasons behind the classification, it is important to understand concept drifts and the changes in the data in order to adapt the actions taken as a result of the model's decision. Duckworth et al. (2021) have shown that understanding the changes in the data during the Covid-19 pandemic can be helpful in life-supporting decisions.

Many approaches for explaining ML models offer no guarantees of rigor, hence being referred to as non-formal (Marques-Silva 2022). Formal Explainability (Marques-Silva and Ignatiev 2022) uses formal representations to provide a formally-correct and minimal explanation of a decision taken by a classifier. Several studies have created these explanations for various model types: Artificial Neural Networks (Bassan and Katz 2022; Jiang et al. 2022), Decision Trees (Audemard et al. 2022), Tree Ensembles (Ignatiev et al. 2022); and for non-specific models (Cooper and Marques-Silva 2022; Huang et al. 2022).

Our approach tries to use the diagnosis as an explanation; however, we aim to detect faulty nodes to explain the concept drift other than explain the classification itself.

## 3 Problem Definition

In this section, we define the problem of poSt concePt drIft Decision trEes Repair (SPIDER) in terms of a diagnosis problem, starting by defining a classification model.

**Definition 2 (Classification Model).** *Given a set of features $F$ from the feature space $\mathcal{X}$, a set of classes $C$ from the class space $\mathcal{Y}$, and a training set of $m$ samples $(x^j, y^j)$ from some* *distribution $D$, where $x^j \in \mathcal{X}$ and $y^j \in C$, a classification model is a function $\phi : \mathcal{X} \rightarrow \mathcal{Y}$.*

A wildly used technique to generate a classification model is building a Binary Decision Tree.

**Definition 3 (Binary Decision Tree).** *A binary decision tree $\mathcal{T} = (V_\mathcal{T}, E_\mathcal{T})$ is a directed acyclic graph where the root node has no incoming edges, and every other node has exactly one incoming edge. The tree nodes are partitioned into terminal $V_T$ and non-terminal $V_{NT}$ nodes. Terminal nodes denote the leaf nodes with no outgoing edges (e.g. children) and are associated with a class taken from $C$. Non-terminal nodes represent the internal nodes and have two outgoing edges. Each non-terminal node $n_j \in V_{NT}$ is denoted by a tuple $\langle f_j, s\rangle$, where $f_j \in F$ and $s$ is a constraint over $f_j$. We denote the constraint $cnst$ of node $n_j$ as $cnst_{n_j}$. Node $n_j$ has two edges (left and right) connected to other nodes (terminal or non-terminal), such that the right edge represents that $cnst_{n_j}$ is satisfied, and the left edge represents that it is not satisfied.*

Given a decision tree model and a new instance, we can assign a class to it by starting at the root node and proceeding to the left or right child, depending on the constraint of the node, until a terminal node is reached. Once we reach a terminal node, we assign the class that the node is associated with. A perfect classification model will classify each new instance to its correct class. However, in reality, classification models are not perfect, and as a result, some instances may be misclassified. The performance of classification models can be evaluated using different measurements such as accuracy, recall, and AUC. Given a decision tree model $\mathcal{T}$ generated based on a set of samples $S_D$ from a distribution $D$, we denote the performance of $\mathcal{T}$ by the function $\rho : (\mathcal{T}, S_D) \rightarrow \mathbb{R}$.

As described above, the tree and its nodes are built for a specific data distribution $D$. However, a concept drift may occur, and the data distribution can change, causing nodes' constraints to be irrelevant to the new distribution, and as a result, decrease the performance of the whole model. That brings us to the definition of a faulty node in a decision tree.

**Definition 4 (Faulty Node).** *A faulty node is a non-terminal (or terminal) node $n_i$ whose constraint (or class) does not fit the current distribution $D'$ due to changes in the data given to the model.*

A non-terminal node's fault may cause samples that pass through it to follow a different path than they ought to, leading to incorrect sample classification by the model. Furthermore, the result of incorrect class assignments due to a faulty terminal node is misclassifying samples. These misclassifications may finally reduce the performance of the decision tree. This leads us to define the Decision Tree Diagnosis Problem:

**Definition 5 (Decision Tree Diagnosis Problem).** *Given a binary decision tree model $\mathcal{T}$ generated based on a set of samples $S_D$ from a distribution D, and given another set of new samples $S_{D'}$ from a distribution $D'$, the Decision Tree Diagnosis Problem arises if $\rho(\mathcal{T}, S_D) - \rho(\mathcal{T}, S_{D'}) > thr$, where $thr$ is a predefined threshold.*

A Decision Tree Diagnosis is a set of faulty nodes that can explain the drop in the performance of the decision tree. As a result of detecting the faulty nodes, we wish to fix those nodes, which means changing the constraint of the non-terminal node or changing its associated class in the case of a terminal node. We expect that the fixed model will perform better on samples from the distribution $D'$ than the original model that was built for distribution $D$, and more specifically will retain the model's performances within the predefined values.

**Definition 6 (Decision Tree Diagnosis).** *Given a Decision Tree Diagnosis Problem with respect to the decision tree $\mathcal{T}$, a diagnosis $\Delta$ is a set of nodes that adjust the tree $\mathcal{T}$ into $\mathcal{T}'$ by changing the node's constraints (or associated class), and causes $\rho(\mathcal{T}, S_D) - \rho(\mathcal{T}', S_{D'}) \leq thr$.*

## 4 Method Description

We propose a diAgnosis-based aPproach for Post-concEpt drifT decIsion Trees rEpair (APPETITE) to solve the SPIDER problem. The input of the algorithm is a decision tree model $\mathcal{T}$ and a set of samples $S_{D'}$ from the new distribution $D'$. The output is a decision tree model adjusted to the distribution $D'$.

APPETITE is divided into two phases:

1. Diagnosing the decision tree: identify the faulty nodes that can explain the performance reduction on the new distribution $D'$ — **the diagnoser**.
2. Fixing the decision tree: based on the diagnosis found by the diagnoser, we fix the faulty nodes in such a way that will improve the model's performance on the new distribution $D'$ — **the fixer**.

We start by proposing two single-fault diagnosers and then explain the proposed fixer.

### 4.1 Diagnosing the Decision Tree (Diagnoser)

Given a decision tree model $\mathcal{T}$ and a set of samples $S_{D'}$, we want to detect the faulty nodes and get a diagnosis $\triangle \subseteq V_\mathcal{T}$ that will explain the drift. In this paper, we focus on a single fault diagnosis, meaning we assume only one component is faulty, hence the diagnosis algorithm will yield a diagnosis composed of only one node. A single-fault is a realistic scenario in general, and specifically in our field since we are looking for the drifted feature.

We present two diagnosis algorithms for diagnosing decision trees: SFL-DT and STAT-AN.

**SFL-Based Diagnosis for Decision Trees (SFL-DT)** Our approach uses the concept of Spectrum Fault Localization (SFL) to diagnose faulty nodes in the Decision Tree. The components and tests are mapped to nodes and samples. Unlike traditional SFL, we define the spectrum and the error vector differently, based on the notations: *Node Path*, *Node Participation*, *Misclassification*.

**Definition 7 (Node Path).** *Given a node $n_i$, a Node Path is a set of constraints associated with the nodes on the path* *from the root node to $n_i$. Let $p_i$ denote the parent of $n_i$:*

$$Path(n_i) = \begin{cases} \emptyset & \text{if } n_i \text{ is the root node} \\ Path(p_i) \cup \{cnst_{p_i}\} & \text{if } n_i \text{ is a right child} \\ Path(p_i) \cup \{\neg cnst_{p_i}\} & \text{if } n_i \text{ is a left child} \end{cases} \tag{2}$$

**Definition 8** (**Node Participation**)**.** *Given a sample $s_j$ and a node $n_i$, a Node Participation is a predicate $np(s_j, n_i)$ that is True if all the constraints in the set $Path(n_i)$ are satisfied by assigning to them $s_j$:*

$$np(s_j, n_i) = \begin{cases} True & \text{if } Path(n_i)(s_j) = True \\ False & otherwise \end{cases} \tag{3}$$

**Definition 9** (**Misclassification**)**.** *Given a sample $s_j$, misclassification is a predicate $mis(s_j)$ that is $True$ if the real class of the sample $y_j$ is not the same as the class given by the model $\phi(s_j)$.*

$$mis(s_j) = \begin{cases} True & \text{if } y_j \neq \phi(s_j) \\ False & otherwise \end{cases} \tag{4}$$

Using the definitions above, we define SFL's spectrum and the error vector as used in our method. The spectrum is denoted as **Node Spectrum** and is filled based on the node participation in the samples:

**Definition 10** (**Node Spectrum**)**.** *Given a set of $A$ samples $S_{D'}$, and a tree with a set of $B$ nodes, a Node Spectrum is a matrix SPCT : $\{0,1\}^{A \times B}$ defined as follows:*

$$SPCT_{j,i} = \begin{cases} 1 & \text{if } np(s_j, n_i) = True \\ 0 & otherwise \end{cases} \tag{5}$$

The error vector is denoted as **Error Vector** and is filled based on the misclassification:

**Definition 11** (**Error Vector**)**.** *Given a set of $A$ samples $S_{D'}$ an Error Vector $E : \{0,1\}^A$ is a vector defined as follows:*

$$e_j = \begin{cases} 1 & \text{if } mis(s_j) = True \\ 0 & otherwise \end{cases} \tag{6}$$

Given a spectrum *SPCT* of size $A \times B$ and an error vector $E$ of size $A$, we can explain the SFL-DT algorithm. In particular, we rank all components (nodes) with a similarity coefficient using a formula that is based on the Similarity Counters defined in Equation 1. In SFL-DT we use Faith (Faith 1983), an asymmetric binary similarity measure:

$$S_{Faith} = \frac{n_{11} + 0.5 \cdot n_{00}}{n_{11} + n_{10} + n_{01} + n_{00}} \tag{7}$$

**STATistical ANalysis Based Diagnosis (STAT-AN)** The main idea behind this algorithm is to use Statistical Analysis to determine which node has the biggest drift by passing instances across the nodes. To further explain this method, we say that $cnst_{n_i}(s_i) = True$ means that the constraint $cnst_{n_i}$ is satisfied by assigning sample $s_i$. Accordingly, $\neg cnst_{n_i}(s_i) = True$ means that the constraint is *violated.* We define a ratio for each node $n_i$ that describes the percentage of the samples of a given set of samples $S$ that violated the node's constraint:

$$violation(n_i, S) = \frac{|\{s_j | \neg cnst_{n_i}(s_j) \wedge np(s_j, n_i), s_j \in S\}|}{|\{s_j | np(s_j, n_i), s_j \in S\}|} \tag{8}$$

Given a binary decision tree $\mathcal{T}$, and two datasets $S_D$ and $S_{D'}$ from two distributions $D$ and $D'$ respectively, we define the node's rank as the difference between the violation given $S_D$ and $S_{D'}$:

$$rank(n_i) = |violation(n_i, S_D) - violation(n_i, S_{D'})| \tag{9}$$

**Choosing the Best Diagnosis** Since we focus on single fault diagnosis, based on the node ranking given by the diagnosis algorithm (SFL-DT or STAT-AN), the chosen diagnosis is the node that has the highest rank. This diagnosis will be used as an input to the fixer, which aims to adjust the model to fit the new distribution $D'$.

### 4.2 Fixing the Decision Tree (Fixer)

Given a decision tree model $\mathcal{T}$ and the diagnosis $\triangle \subseteq V_{\mathcal{T}}$ found in the previous step, we aim to change the decision tree in a way that will fit better for the distribution $D'$.

The fix is done for each node in the diagnosis, but the changes we make in the nodes depend on the node's type. If $n_i$ is a **terminal node** (a leaf), the class associated with the node will be changed to the most frequent class in the node based on the dataset after the drift ($S_{D'}$). This is a cautious fix, and it is possible that the node's associated class will not change due to the fix.

If $n_i$ is a **non-terminal node**, we change the constraint of the node according to the feature type. For nodes associated with binary or categorical features, we negate the node's constraint $cnst'_{n_i} = \neg cnst_{n_i}$. As for nodes associated with a numeric feature, assume $f \in n_i$ is $f_j$, we change the constraint $cnst_{n_i} = (f_j \leq x_j)$ to be $cnst'_{n_i} = (f_j \leq x_j + diff_j)$, where $diff_j$ is the difference between the feature's $f_j$ average of the samples that pass through the node $n_i$ after and before the drift:

$$diff_j = E[f_{j_{D'}}] - E[f_{j_D}] \tag{10}$$

### 4.3 Complexity Analysis

Denote $S$ as the number of samples, $X$ as the number of features, and $V$ as the number of nodes in the tree. The proposed algorithms' time and space complexity analysis compared to the alternative approach of training a new tree is presented in Table 2.

## 5 Evaluation

This section evaluates APPETITE by running it on semi-synthesized data with concept drift in specific features. We will evaluate the diagnoser by measuring the diagnosis correctness and evaluate the fixer by comparing the fixed model to two alternative re-trained models. We describe the experimental setup, including the datasets and the drift synthesizing process, and present the results.

| Algorithm | Complexity | |
|---|---|---|
| | Time | Space |
| SFL-DT Diagnoser | $O(V \cdot S)$ | $O(V \cdot S)$ |
| STAT-AN Diagnoser | $O(V \cdot S)$ | $O(V + S)$ |
| The Fixer | $O(1)$ | $O(V)$ |
| Training a new tree | $O(S \cdot log(S) \cdot X)$ | $O(V + S)$ |

Table 2: Time and space complexity analysis.

### 5.1 Experimental Settings

**Datasets** One of the challenges in the research area of concept drift is the data used to evaluate the algorithms handling it. As described in (Li et al. 2020; Brzezinski and Stefanowski 2013), the datasets used to test algorithms are intended for static environments; therefore, they typically do not contain any concept drift, and even if they do, the drifts' locations are not labeled. In terms of real-world data, some studies (Duckworth et al. 2021; Wang et al. 2003) have used private data that can not be replicated, and there is still a lack of suitable benchmark datasets that are freely accessible. For these reasons, concept drift handling is mainly tested on synthetic datasets containing drifts in known positions. A popular way to synthesize these datasets is using MOA (Bifet et al. 2010), an open-source framework for data stream mining, that allows, among other things, to generate data streams with concept drifts. The drifts generated with the tool affect **the entire dataset**, and since our motivation is to detect the changes in **specific features**, the data generated by MOA is not suitable for our problem. Tahmasbi et al. (2021) created semi-synthetic datasets by taking an existing dataset without drifts and synthesizing drifts within the samples. Similar to that approach, we used static datasets and synthesized drifts in specific features, as described below.

To evaluate APPETITE we chose a variety of 73 supervised classification data sets taken from the public datasets TRIO (Cohen-Shapira and Rokach 2021), built for several machine learning algorithms tasks, such as classification, regression, and clustering. We used the datasets only targeted for the classification task since our study focuses on decision trees used for classification. First, since the samples in most datasets are ordered by their label or feature values, we shuffled the instances in the dataset to get a better data distribution. Then, we divided the datasets into three parts, where the first one (70% of the data) is used to simulate the data before the drift (original distribution $D$). The other two parts are for the data after the drift ($D'$), where the first part is used by the different algorithms to update or rebuild the model (0.5%-10% of the data), and the last one serves as the test set to evaluate the algorithms after the drift (the last 20% of the instances).

TRIO contains 138 classification datasets. Still, we used only 73 datasets since we filtered out datasets (1) that the decision tree model generated using the first 70% had an accuracy lower than 75% since we would like to run the algorithm in cases where the original model created for $S_D$ is good enough. In addition, we filtered out datasets (2) that the drop in the accuracy of the decision tree model $\mathcal{T}$, between samples from the first part before the drift $S_D$ and from the second part after the drift $S_{D'}$ is at least 10%: $\rho(\mathcal{T}, D) - \rho(\mathcal{T}, D') \geq 0.1$. A lower drop indicates an insignificant drift. The complete list of datasets we used for the experiments can be found in Section 6 (Appendix).

**Generating Decision Trees** All trees were generated using the Sklearn library. First, we chose the hyper-parameters splitting criterion [Gini (Tangirala 2020), or Entropy (Loh 2011)] and the maximum number of leaf nodes in the tree [10, 20, or 30] with Grid Search (Larochelle et al. 2007), using cross-validation on the training set. The number of splits is set to the smallest class size, but no greater than 5. After choosing the best hyper-parameters, we trained the tree until it was fully grown and used post-pruning (Mansour 1997) to get the minimal tree that gave a maximum accuracy score on the validation set.

**Concept Drift Synthesizing Process** For each feature, we synthesized concept drift that changed the distribution of that feature, creating a new dataset for the experiments. We changed only the last two parts of the data (the 30% as described in 5.1); therefore, the initial 70% remains the same.

The changes in the distribution depend on the type of the feature; the drift of **numeric features** was done by adding $k$ to the feature in all samples. The value of $k$ was set to be $[-2\sigma, -\sigma, -0.5\sigma, 0.5\sigma, \sigma, 2\sigma]$, where $\sigma$ is the standard deviation of the feature in all the dataset. As for the drifts of **categorical and binary features**, we fixed each value of the feature and changed a proportion $p$ out of the remaining samples to that value. We experimented with several proportions $p$: [0.3, 0.5, 0.7, 0.9]. In total, we created 6 drifts for each numeric feature, and 4·—unique values— drifts for binary and categorical features, bringing us to 4194 total datasets containing varied concept drifts.

We used these distributions to rank the **severity** of the drift. The more the distribution differs from the feature's original distribution, the greater the severity of the drift.

1. **Numeric:** $k = 0.5\sigma, -0.5\sigma$ **Binary/Categorical:** $p = 0.3$
2. **Numeric:** $k = \sigma, -\sigma$ **Binary/Categorical:** $p = 0.5, 0.7$
3. **Numeric:** $k = 2\sigma, -2\sigma$ **Binary/Categorical:** $p = 0.9$

**Compared Algorithms** To evaluate the diagnosis algorithms, we compare **SFL-DT** and **STAT-AN** to a diagnosis algorithm that combines these two algorithms by multiplying the ranks they return for each node. We call this algorithm **SFL+STAT**. Preliminary experiments on SFL+STAT with different similarity measures (Faith, Cosine, Dice, Jaccard, Intersection, and Inner Product) showed that the similarity measure Faith performs the best. Therefore, we present the results of SFL-DT and SFL+STAT with the Faith similarity measure.

To evaluate the APPETITE approach, we run the above three algorithms with the fixer (Section 4.2). In addition, we compare these algorithms to algorithms that train a new model since it is the known alternative in the literature to handling concept drifts in a *non-incremental* approach (Gepperth and Hammer 2016). There are two ways to train a new model that will fit the new distribution $D'$. One is to train a new model only on the samples from the new distribution ($S_{D'}$) (e.g., the second part of the data). We call this algorithm **NEWDRIFT**. The second way is to train the model on all data up to this point, including the data from both $D$ and $D'$ ($S_D \cup S_{D'}$). We call this algorithm **NEWALL**. It is worth mentioning that the second alternative is not always possible because, in many cases, the data used to train the model is not stored but only the model itself due to memory considerations.

**Evaluation Metrics** To measure the diagnoser's performance, and verify if it could be used as an explanation for the essence of the drift, we checked if the feature associated with the node in the diagnosis is the same as the one we changed. We reported the percentage of the cases where we correctly **identified** the feature in our diagnosis. Additionally, we compute the average **Wasted Effort** (Elmishali, Stern, and Kalech 2018): assume that we repair the components in the diagnoses in the order of the diagnoses' rank, Wasted Effort is the number of healthy components examined until all faulty components are repaired. In our case, the diagnosis is a single fault, and our fault is a feature (not a node), so Wasted Effort is the number of nodes with a different associated feature that will be fixed until we reach a node with the faulty feature.

To evaluate the performances of the algorithms as an approach for handling the SPIDER problem, we measured the accuracy of the different algorithms and compared them to the accuracy of the *original model* on the same test set. We report on the **Accuracy Diff** — the difference between the two in terms of percentage.

**Research Questions** We define three research questions that will be examined through the evaluation:
**RQ1.** *Which one of the diagnosis algorithms returns the best diagnosis?*
**RQ2.** *Does* APPETITE *approach perform better than algorithms that retrain a new classification model?*
**RQ3.** *How do parameters such as the percentage of instances used to fix the model or the severity of the drift affect the diagnosis algorithms and the algorithms that solve the* SPIDER *problem?*

## 5.2 Results

Table 3 shows the results. The first column addresses RQ2 and shows the average **Accuracy Diff**. The last two columns address RQ1 and present the performance of the diagnosis in terms of **Wasted Effort** (lower values are better) and the percentage of cases we **Identified** the faulty feature. The best values are shown in bold. The first two rows show the results of training new models based on data before and after the drift (NEWALL) and only after (NEWDRIFT). The last three rows present the results of our diagnosis and repair algorithms: SFL-DT, STAT-AN, and the combined algorithm, SFL+STAT.

To better observe the results, we examined the performances of the different algorithms based on the type of feature that had a drift. Results can be seen in Table 4, where we present the same information as in Table 3 but separated based on the feature's type.

| Algorithm | Accuracy Diff | Wasted Effort | Identified |
|---|---|---|---|
| NEWALL | **10.06%** | — | — |
| NEWDRIFT | -6.24% | — | — |
| SFL-DT | -0.44% | 2.64 | 31.79% |
| STAT-AN | 2.49% | 2.01 | 38.85% |
| SFL+STAT | 8.40% | **0.97** | **71.36%** |

Table 3: Average performances of the algorithms across all modified datasets (addressing RQ1 and RQ2).

| Type | Algorithm | Acc. Diff | WasEff | Identified |
|---|---|---|---|---|
| Categorical/ Binary | NEWALL | **2.25%** | — | — |
| | NEWDRIFT | -11.93% | — | — |
| | SFL-DT | -37.06% | 1.85 | 59.36% |
| | STAT-AN | -3.19% | 1.76 | 33.69% |
| | SFL+STAT | -11.54% | **0.98** | **62.57%** |
| Numeric | NEWALL | 11.27% | — | — |
| | NEWDRIFT | -5.36% | — | — |
| | SFL-DT | 5.22% | 2.76 | 27.53% |
| | STAT-AN | 3.37% | 2.05 | 39.65% |
| | SFL+STAT | **11.48** % | **0.97** | **72.71%** |

Table 4: Average performance of the algorithms for drifts on categorical/binary and numeric features.

**[RQ1:]** Table 3 and Table 4 show that the combined algorithm (SFL+STAT) is much better than SFL-DT and STAT-AN separated. This can be seen by the lower wasted effort and the higher %Identified. The difference is significant and tested using a T-test with $\alpha = 0.05$. The rest of the results were tested with the same statistical test. In terms of diagnosis, we can say that SFL+STAT provides a good diagnosis that can explain the drift in more than 71% of the cases. The algorithm is better when it comes to numeric features, but the results are also satisfying for categorical features.

**[RQ2:]** As for the improvement in the model's performance after the fix, we can see that both STAT-AN and SFL+STAT algorithms (last two rows) improve the original model, with a statistically significant difference. If we differ by feature type, we can see that all three algorithms perform better when fixing a numeric drift, even though they have similar results in the diagnosis phase. We can assume that our fixer is not suitable for categorical/binary features and an alternative solution should be proposed. The numeric results in Table 4 show that all three algorithms improve the original model's result. Additionally, all three algorithms are better than training a new model on the samples only after the drift (NEWDRIFT), and the difference is statistically significant. One of the possible explanations for that is the fact that we do not use many samples, and the new model might suffer from overfitting. The combined algorithm (SFL+STAT) is even better than training a new model on all available data (NEWALL), but the difference is not statistically significant. This result can justify not saving the samples used to train the model and save memory. Additionally, according to the complexity analysis in Table 2, our algorithm is not more complex than the alternative. Moreover, the advantage of our approach is that it is more explainable since it does not only fix the model but also identifies the drifted feature.

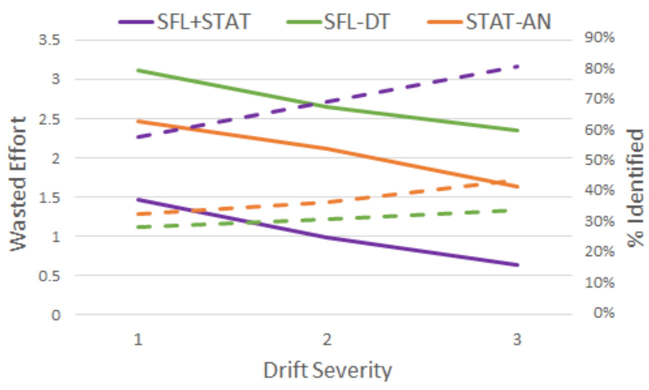

Figure 1: Diagnosis quality as a function of the drift severity. The full line is Wasted Effort, and the dashed line is % Identified.

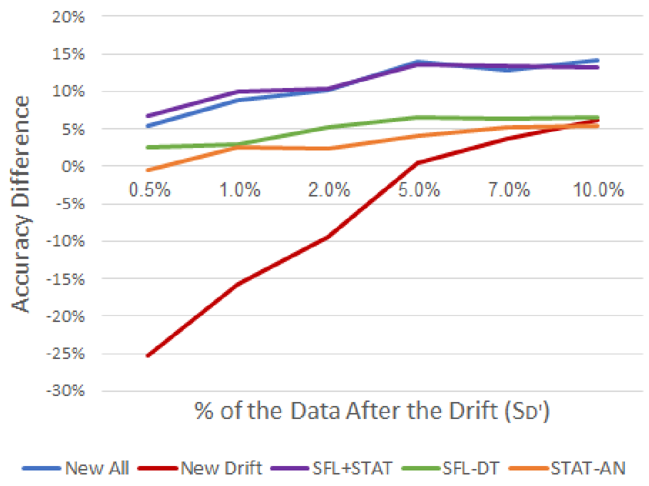

Figure 2: Accuracy difference of the algorithms on numeric drifts as a function of the drift data size.

To address **RQ3**, we present results of (1) the impact of the drift severity on the diagnosis and (2) the impact of the amount of data after the drift used to fix the model.

In Figure 1 the x-axis is the drift severity as described in 5.1, and the y-axis is the average Wasted Effort (for full lines) or %Identified (dashed). Results show that the quality of the diagnosis has a correlation to the severity of the drift. The three algorithms have better diagnoses as the drift is more severe. We can conclude that APPETITE is better as an active approach to address concept drift, which is the preferred solution for abrupt drifts as the ones presented in our experiments.

In Figure 2 we see how the size of $S_{D'}$ affects all algorithms' performances, showing results only for numeric drifts. The x-axis is the percentage of the dataset used by the algorithms, and the y-axis is the Accuracy Difference. We can see that all 5 algorithms perform better when they use larger $S_{D'}$. Results also show that SFL+STAT and NEWALL show the same and most significant improvement, after them SFL-DT and STAT-AN improve the accuracy of the original model but do not get the same performance as the former two. As for NEWDRIFT, it is the worst out of all 5, and when $S_{D'}$ is smaller than 5%, it is even harming the original model and does not improve it at all. In addition, the gap between NEWDRIFT and all other alternatives shrinks as the dataset size increases. From this analysis, we can conclude that our approach has the highest impact in cases when we do not have many samples from the new distribution $D'$, and the old samples from $D$ were not saved.

## 6 Conclusions

In this paper, we presented a novel approach APPETITE to address the problem of post-concept drift decision tree repair. This approach applies two steps. First, a diagnosis process is run to identify the node whose constraint does not fit anymore the new data distribution, then a fixer process is run to fix this node. We presented two algorithms to address the diagnosis process, one is based on SFL (SFL-DT), and the other is based on a statistical analysis of the data passed through the node (STAT-AN). In addition, we presented an algorithm that fixes the faulty node. Experiments on 73 datasets show that these algorithms perform better than the original model when it comes to numeric drifts. Moreover, they are even better than an algorithm that trains a new model based on the data after the drift (NEWDRIFT). Also, we show that a combination of the two algorithms (SFL+STAT) performs much better than each one separately; it can explain the essence of the drift and is competitive with an algorithm that trains a new model based on the entire data, before and after the drift (NEWALL).

For future work we suggest extending the diagnosis algorithms to support multi-fault diagnoses, meaning each diagnosis can have more than one faulty node. Additionally, since our suggested algorithm for the fixer did not provide the outcome we aimed for in categorical and binary features, we wish to improve it.

## Appendix

In table 5, we present a summary of the datasets used for the experiments, including the classification information for each decision tree model trained and tested with the respective data. It includes the number of samples in the dataset [#S], the number of numerical features [#N], the number of binary or categorical features [#C], and the type of the target class in the dataset [CT], i.e., whether it is a binary (marked as B) or a multi-class (marked as C) classification task. Moreover, in regards to the trained decision tree, we include the number of nodes in the tree (tree size [TS]) and the accuracy [Acc] of the decision tree, tested on the test set, in terms of percentage.

## Acknowledgments

This research was funded by ISF grant No. 1716/17.

| Name | #S | #N | #C | CT | TS | Acc |
|---|---|---|---|---|---|---|
| Acute-Inflammation | 120 | 6 | 0 | B | 9 | 100 |
| Acute-Nephritis | 120 | 7 | 0 | B | 5 | 96 |
| Analcat-data_boxing1 | 120 | 1 | 2 | B | 21 | 78 |
| Analcat-data_lawsuit | 264 | 4 | 0 | B | 13 | 100 |
| Annealing | 898 | 31 | 0 | C | 51 | 93 |
| Ar4 | 107 | 29 | 0 | B | 17 | 75 |
| Audiology-Std | 196 | 59 | 0 | C | 19 | 87 |
| Bank | 4521 | 16 | 0 | B | 17 | 92 |
| Baseball | 1340 | 15 | 1 | C | 5 | 93 |
| Blood | 748 | 4 | 0 | B | 11 | 80 |
| Bodyfat | 252 | 14 | 0 | B | 3 | 98 |
| Braziltourism | 412 | 1 | 7 | C | 7 | 75 |
| Breast-Cancer-Wisc-Diag | 569 | 30 | 0 | B | 7 | 89 |
| Breast-Cancer-Wisc | 699 | 9 | 0 | B | 19 | 92 |
| Car | 1728 | 6 | 0 | C | 55 | 92 |
| Caradiotocogra-phy10clases | 2126 | 21 | 0 | C | 45 | 84 |
| Cardiotocogra-phy3clases | 2126 | 21 | 0 | C | 23 | 93 |
| Chatfield_4 | 235 | 12 | 0 | B | 17 | 85 |
| Chscase_vine1 | 52 | 9 | 0 | B | 7 | 100 |
| Credit-Approval | 690 | 15 | 0 | B | 3 | 85 |
| Dermatology | 366 | 34 | 0 | C | 13 | 92 |
| Diggle_table_a2 | 310 | 8 | 0 | B | 3 | 100 |
| Echocardiogram | 131 | 10 | 0 | B | 3 | 92 |
| Ecoli | 336 | 7 | 0 | C | 5 | 77 |
| Elusage | 55 | 2 | 0 | B | 7 | 90 |
| Energy-Y1 | 768 | 8 | 0 | C | 19 | 94 |
| Energy-Y2 | 768 | 8 | 0 | C | 15 | 95 |
| Fertility | 100 | 9 | 0 | B | 1 | 79 |
| Heart-Hungarian | 294 | 12 | 0 | B | 29 | 89 |
| Hepatitis | 155 | 19 | 0 | B | 11 | 80 |
| Horse-Colic | 368 | 25 | 0 | B | 13 | 83 |
| Image-Segmentation | 2310 | 18 | 0 | C | 51 | 95 |
| Ionosphere | 351 | 33 | 0 | B | 17 | 87 |
| Iris | 150 | 4 | 0 | C | 9 | 90 |
| Kc3 | 458 | 39 | 0 | B | 5 | 87 |
| Kidney | 76 | 4 | 2 | B | 9 | 93 |
| Low-Res-Spect | 531 | 100 | 0 | C | 19 | 81 |
| Lowbwt | 189 | 9 | 0 | B | 3 | 89 |
| Mammo-graphic | 961 | 5 | 0 | B | 5 | 82 |
| Meta | 528 | 19 | 2 | B | 1 | 87 |
| Mfeat-Karhunen | 2000 | 64 | 0 | C | 57 | 76 |
| Mfeat-Morphological | 2000 | 6 | 0 | B | 11 | 100 |
| Molec-Biol-Splice | 3190 | 60 | 0 | C | 55 | 93 |
| Monks-1 | 556 | 6 | 0 | B | 29 | 77 |
| Monks-3 | 554 | 6 | 0 | B | 5 | 92 |
| Newton_hema | 140 | 2 | 1 | B | 17 | 85 |
| Oocytes_merlucci-us_states_2f | 1022 | 25 | 0 | C | 13 | 85 |
| Oocytes_trispterus-_state_5b | 912 | 32 | 0 | C | 15 | 90 |
| Ozone | 2536 | 72 | 0 | B | 1 | 97 |
| Parkinsons | 195 | 22 | 0 | B | 11 | 82 |
| Pittsburg-Bridges-MATERIAL | 106 | 7 | 0 | C | 13 | 80 |
| Pittsburg-Bridges-T-OR-D_R | 102 | 7 | 0 | B | 25 | 79 |
| Prnn_synth | 250 | 2 | 0 | B | 3 | 88 |
| Rabe_131 | 50 | 5 | 0 | B | 3 | 100 |
| Schlvote | 38 | 5 | 0 | B | 3 | 83 |
| Seeds | 210 | 7 | 0 | C | 15 | 95 |
| Socmob | 1156 | 1 | 4 | B | 29 | 94 |
| Solar-Flare | 1066 | 9 | 3 | C | 1 | 99 |
| Soybean | 683 | 35 | 0 | C | 57 | 84 |
| Spect | 265 | 22 | 0 | B | 5 | 77 |
| Spectf | 267 | 44 | 0 | B | 5 | 83 |
| Squash-Unstored | 52 | 22 | 1 | C | 3 | 100 |
| Statlog-Image | 2310 | 18 | 0 | C | 51 | 96 |
| Synthetic-Control | 600 | 60 | 0 | C | 41 | 78 |
| Tic-Tac-Toe | 958 | 9 | 0 | B | 51 | 94 |
| Transplant | 131 | 2 | 0 | B | 3 | 92 |
| Vertebral-Column-2clases | 310 | 6 | 0 | B | 15 | 77 |
| Vertebral-Column-3clases | 310 | 6 | 0 | C | 13 | 82 |
| Visualiz-ing_livestock | 130 | 0 | 2 | B | 5 | 96 |
| Vote | 435 | 0 | 16 | B | 19 | 97 |
| Wall-Following | 5456 | 24 | 0 | C | 33 | 99 |
| Wine | 178 | 13 | 0 | C | 11 | 88 |
| Zoo | 101 | 16 | 0 | C | 13 | 95 |

Table 5: A description of the datasets used in our study to evaluate the APPETITE approach for concept drift handling. This table includes the dataset names, size, number of features and types, and class type. Moreover, it includes the decision tree model size and its accuracy on the test set.

## References

Abreu, R.; Zoeteweij, P.; and Van Gemund, A. J. 2007. On the accuracy of spectrum-based fault localization. In *TAICPART-MUTATION 2007*, 89–98. IEEE.

Abreu, R.; Zoeteweij, P.; and Van Gemund, A. J. 2009. Spectrum-based multiple fault localization. In *IEEE/ACM ASE*, 88–99. IEEE.

Audemard, G.; Bellart, S.; Bounia, L.; Koriche, F.; Lagniez, J.-M.; and Marquis, P. 2022. On the explanatory power of Boolean decision trees. *Data & Knowledge Engineering* 142:102088.

Bassan, S., and Katz, G. 2022. Towards formal approximated minimal explanations of neural networks. *arXiv preprint arXiv:2210.13915*.

Bholat, D.; Gharbawi, M.; and Thew, O. 2020. The impact of Covid on machine learning and data science in UK banking. *Bank of England Quarterly Bulletin* Q4.

Bifet, A.; Holmes, G.; Pfahringer, B.; Kranen, P.; Kremer, H.; Jansen, T.; and Seidl, T. 2010. Moa: Massive online analysis, a framework for stream classification and clustering. In *Proceedings of the first workshop on applications of pattern analysis*, 44–50. PMLR.

Brzezinski, D., and Stefanowski, J. 2013. Reacting to different types of concept drift: The accuracy updated ensemble algorithm. *IEEE Transactions on Neural Networks and Learning Systems* 25(1):81–94.

Burkart, N., and Huber, M. F. 2021. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research* 70:245–317.

Castro-Cabrera, P.; Castellanos-Dominguez, G.; Mera, C.; Franco-Marín, L.; and Orozco-Alzate, M. 2021. Adaptive classification using incremental learning for seismic-volcanic signals with concept drift. *Journal of Volcanology and Geothermal Research* 413:107211.

Cohen-Shapira, N., and Rokach, L. 2021. Trio: Task-agnostic dataset representation optimized for automatic algorithm selection. In *2021 IEEE International Conference on Data Mining (ICDM)*, 81–90. IEEE.

Cooper, M. C., and Marques-Silva, J. 2022. Tractability of explaining classifier decisions. *Artificial Intelligence* 103841.

Domingos, P., and Hulten, G. 2000. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, 71–80.

Došilović, F. K.; Brčić, M.; and Hlupić, N. 2018. Explainable artificial intelligence: A survey. In *2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO)*, 0210–0215. IEEE.

Duckworth, C.; Chmiel, F. P.; Burns, D. K.; Zlatev, Z. D.; White, N. M.; Daniels, T. W.; Kiuber, M.; and Boniface, M. J. 2021. Using explainable machine learning to characterise data drift and detect emergent health risks for emergency department admissions during COVID-19. *Scientific reports* 11(1):1–10.

Elmishali, A.; Stern, R.; and Kalech, M. 2018. An artificial intelligence paradigm for troubleshooting software bugs. *Engineering Applications of Artificial Intelligence* 69:147–156.

Faith, D. P. 1983. Asymmetric binary similarity measures. *Oecologia* 57(3):287–290.

Gao, W.; Bai, Z.; Zhu, F.; Chou, C. C.; and Jiang, B. 2021. A study on the cyclist head kinematic responses in electric-bicycle-to-car accidents using decision-tree model. *Accident Analysis & Prevention* 160:106305.

Gepperth, A., and Hammer, B. 2016. Incremental learning algorithms and applications. In *European symposium on artificial neural networks (ESANN)*.

Gomes, H. M.; Bifet, A.; Read, J.; Barddal, J. P.; Enembreck, F.; Pfharinger, B.; Holmes, G.; and Abdessalem, T. 2017. Adaptive random forests for evolving data stream classification. *Machine Learning* 106(9):1469–1495.

Gomez-Omella, M.; Esnaola-Gonzalez, I.; and Ferreiro, S. 2020. Short-term forecasting methodology for energy demand in residential buildings and the impact of the COVID-19 pandemic on forecasts. In *Artificial Intelligence XXXVII: 40th SGAI International Conference on Artificial Intelligence, AI 2020, Cambridge, UK, December 15–17, 2020, Proceedings 40*, 227–240. Springer.

Hoeffding, W. 1963. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association* 58(301):13–30.

Hoeglinger, S., and Pears, R. 2007. Use of Hoeffding trees in concept based data stream mining. In *2007 Third International Conference on Information and Automation for Sustainability*, 57–62. IEEE.

Hofer, B.; Perez, A.; Abreu, R.; and Wotawa, F. 2015. On the empirical evaluation of similarity coefficients for spreadsheets fault localization. *Automated Software Engineering* 22(1):47–74.

Huang, X.; Izza, Y.; Ignatiev, A.; Cooper, M.; Asher, N.; and Marques-Silva, J. 2022. Tractable explanations for d-dnnf classifiers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 5719–5728.

Hulten, G.; Spencer, L.; and Domingos, P. 2001. Mining time-changing data streams. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 97–106.

Ignatiev, A.; Izza, Y.; Stuckey, P. J.; and Marques-Silva, J. 2022. Using maxsat for efficient explanations of tree ensembles. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, 3776–3785.

Jiang, J.; Leofante, F.; Rago, A.; and Toni, F. 2022. Formalising the robustness of counterfactual explanations for neural networks. *arXiv preprint arXiv:2208.14878*.

Jones, J. A., and Harrold, M. J. 2005. Empirical evaluation of the Tarantula automatic fault-localization technique. In *IEEE/ACM ASE*, 273–282.

Khannouz, M., and Glatard, T. 2022. Mondrian forest for data stream classification under memory constraints. *arXiv preprint arXiv:2205.07871*.

Klinkenberg, R., and Joachims, T. 2000. Detecting concept drift with support vector machines. In *ICML*, 487–494.

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

Lakshminarayanan, B.; Roy, D. M.; and Teh, Y. W. 2014. Mondrian forests: Efficient online random forests. *Advances in neural information processing systems* 27.

Larochelle, H.; Erhan, D.; Courville, A.; Bergstra, J.; and Bengio, Y. 2007. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, 473–480. New York, NY, USA: Association for Computing Machinery.

Lee, C. S.; Cheang, P. Y. S.; and Moslehpour, M. 2022. Predictive analytics in business analytics: decision tree. *Advances in Decision Sciences* 26(1):1–29.

Li, Z.; Huang, W.; Xiong, Y.; Ren, S.; and Zhu, T. 2020. Incremental learning imbalanced data streams with concept drift: The dynamic updated ensemble algorithm. *Knowledge-Based Systems* 195:105694.

Loh, W.-Y. 2011. Classification and regression trees. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* 1.

Mansour, Y. 1997. Pessimistic decision tree pruning based on tree size. In *MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-*, 195–201. Citeseer.

Marques-Silva, J., and Ignatiev, A. 2022. Delivering trustworthy AI through formal XAI. In *Proc. of AAAI*, 3806–3814.

Marques-Silva, J. 2022. Logic-based explainability in machine learning. *arXiv preprint arXiv:2211.00541*.

Rad, R. H., and Haeri, M. A. 2019. Hybrid forest: A concept drift aware data stream mining algorithm. *arXiv preprint arXiv:1902.03609*.

Rokach, L., and Maimon, O. 2005. *Decision Trees*. Boston, MA: Springer US. 165–192.

Roy, D. M., and Teh, Y. W. 2008. The Mondrian process. In *NIPS*, volume 21.

Rungskunroch, P.; Jack, A.; and Kaewunruen, S. 2021. Benchmarking on railway safety performance using Bayesian inference, decision tree and Petri-net techniques based on long-term accidental data sets. *Reliability Engineering & System Safety* 213:107684.

Saffari, A.; Leistner, C.; Santner, J.; Godec, M.; and Bischof, H. 2009. On-line random forests. In *2009 IEEE 12th international conference on computer vision workshops, iccv workshops*, 1393–1400. IEEE.

Scholz, M., and Klinkenberg, R. 2005. An ensemble classifier for drifting concepts. In *Proceedings of the Second International Workshop on Knowledge Discovery in Data Streams*, volume 6, 53–64. Porto, Portugal.

Singhal, S.; Chawla, U.; and Shorey, R. 2020. Machine learning & concept drift based approach for malicious website detection. In *2020 International Conference on COMmunication Systems & NETworkS (COMSNETS)*, 582–585. IEEE.

Tahmasbi, A.; Jothimurugesan, E.; Tirthapura, S.; and Gibbons, P. B. 2021. Driftsurf: Stable-state/reactive-state learning under concept drift. In *International Conference on Machine Learning*, 10054–10064. PMLR.

Tangirala, S. 2020. Evaluating the impact of gini index and information gain on classification using decision tree classifier algorithm. *International Journal of Advanced Computer Science and Applications* 11(2):612–619.

Wang, H.; Fan, W.; Yu, P. S.; and Han, J. 2003. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 226–235.

Xie, T.; Peng, Y.; and Wang, C. 2016. hi-rf: Incremental learning random forest for large-scale multi-class data classification. *arXiv preprint arXiv:1608.08761*.

Zhukov, A. V.; Sidorov, D. N.; and Foley, A. M. 2016. Random forest based approach for concept drift handling. In *International Conference on Analysis of Images, Social Networks and Texts*, 69–77. Springer.