Bachelor Thesis

Czech Technical University in Prague

F3

Faculty of Electrical Engineering
Department of Cybernetics

Comparing Methods for Detecting Concept Drift in Data Streams

Daniil Barabašev

Supervisor: doc. Mgr. Viliam Lisý, MSc., Ph.D.
Study program: Open Informatics
Specialisation: Artificial Intelligence and Computer Science
May 2024

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name:	**Barabašev Daniil**	Personal ID number:	**498876**
Faculty / Institute:	**Faculty of Electrical Engineering**		
Department / Institute:	**Department of Cybernetics**		
Study program:	**Open Informatics**		
Specialisation:	**Artificial Intelligence and Computer Science**		

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Comparing Methods for Detecting Concept Drift in Data Streams

Bachelor’s thesis title in Czech:

Porovnání metod pro detekci driftu konceptu v datových tocích

Guidelines:

Concept drift, the evolving nature of data, necessitates constant model adaptation. In dynamic environment, understanding and mitigating concept drift ensures the sustained accuracy and reliability of machine learning models. We consider a machine learning model designed to detect malicious applications. Initially, the model is trained on historical data, successfully identifying known threats. However, over time, the nature of benign and malicious applications evolves. Without addressing concept drift, the model's performance may degrade as it fails to recognize these novel trends.
The aim of this project is to test and improve existing methods of detection concept drift by tuning them for a cybersecurity task and aviable dataset for the said field.
(1) Review literature for the concept drift problem.
(2) Research state-of-the-art methods of detecting concept drift especially in the field of cybersecurity.
(3) Decide on suitable datsets and methodologies to perform an evaluation of chosen methods.
(4) Perform experiemnts comparing different concept drift detection methods on the datasets.
(5) Evaluate the approaches and suggest an improvement for a chosen method.

Bibliography / sources:

[1] L. Yang, A. Ciptadi, I. Laziuk, A. Ahmadzadeh and G. Wang, "BODMAS: An Open Datasetfor Learning based Temporal Analysis of PE Malware," 2021 IEEE Security and Privacy Workshops (SPW), San Francisco, CA, USA, 2021, pp. 78-84, doi: 10.1109/SPW53761.2021.00020
[2] Gama, João & Žliobaitė, Indrė & Bifet, Albert & Pechenizkiy, Mykola & Bouchachia, Hamid. (2014). A Survey on Concept Drift Adaptation. ACM Computing Surveys (CSUR). 46. 10.1145/2523813
[3] Khamassi, I., Mouchaweh, M.S., Hammami, M., & Ghédira, K. (2018). Discussion and review on evolving data streams and concept drift adapting. Evolving Systems, 9, 1-23.

Name and workplace of bachelor’s thesis supervisor:

doc. Mgr. Viliam Lisý, MSc., Ph.D. Artificial Intelligence Center FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **01.02.2024** Deadline for bachelor thesis submission: **24.05.2024**

Assignment valid until: **21.09.2025**

doc. Mgr. Viliam Lisý, MSc., Ph.D.	prof. Dr. Ing. Jan Kybic	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

.	
Date of assignment receipt	Student’s signature

Acknowledgements

I want to thank my supervisor, doc. Mgr. Viliam Lisý, MSc., Ph.D., for a great opportunity to explore and experiment with machine learning concepts and provide valuable feedback for finishing this thesis.

My appreciation goes to my family and friends for their unwavering support and encouragement throughout my studies.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 24. May 2024

Abstract

This thesis investigates various methods for detecting concept drift in data streams and testing it on a cybersecurity dataset. It describes the current understanding of concept drift and discusses available literature for concept drift detection. The research evaluates several strategies that address concept drift detection. The experimental evaluation utilizes a cybersecurity dataset to compare how these strategies can handle a real-world scenario. Lastly, I propose an improvement to the capability of MD3 and D3 methods by employing a hybrid of these methodologies. The experiments show that tracking and handling concept drift in data streams improves classifiers' performance for the price of some overhead.

Keywords: machine learning, data streams, concept drift, real-time data processing, cybersecurity

Supervisor: doc. Mgr. Viliam Lisý, MSc., Ph.D.

Abstrakt

Tato práce se zabývá různými metodami detekce driftu konceptu v datových tocích a testuje je na datasetu z oblasti kybernetické bezpečnosti. Popisuje současné chápání driftu konceptu a diskutuje o dostupné literatuře pro detekci tohoto fenoménu. Výzkum porovnává několik strategií, které se zabývají detekcí. Experimenty využívají dataset z oblasti kybernetické bezpečnosti a porovnává, jak si tyto strategie poradí s reálným scénářem. Nakonec tato práce navrhuje vylepšení metod MD3 a D3 využitím hybridu těchto metodik. Experimenty ukazují, že sledování a řešení driftu konceptu v datových tocích zlepšuje výkon klasifikátorů za cenu zvýšené režie.

Klíčová slova: strojové učení, datové toky, drift konceptů, zpracování dat v reálném čase, kyberbezpečnost

Překlad názvu: Porovnání metod pro detekci koncept driftu v datových tocích

Contents

Figures

Tables

Chapter 1

Introduction

The digital revolution has brought a dark underbelly – an escalating wave of cyber threats and attacks. Cybersecurity faces an ever-increasing challenge to protect sensitive information and critical infrastructure as malicious actors continually develop new tactics and sophisticated tools. Integrating machine learning into cybersecurity practices is a promising frontier in this context.

In today's world, machine learning models are broadly applied in cybersecurity. They are crucial in providing innovative tools to detect, prevent or respond to cyber-attacks. The data fed to these models often come in the form of streams, and accommodating large volumes of data is impractical or even impossible. Therefore, models are trained in advance to limit storage capacity and memory load. However, the cybersecurity environment is nonstationary, and parameters can change over time. This introduces concept drift, which refers to changes in the conditional distributions given the input, while the distribution of the input may stay unchanged [1]. The main focus of this paper is to research modern solutions for concept drift and apply them in classifying malware samples.

Chapter 2
Concept drift

In machine learning, the phenomenon of concept drift is known as the change in the properties of the target variable, which the model is trying to predict [1]. As time evolves, so does the connection between input features and target variables, necessitating model changes. These changes can be highly problematic for machine learning systems structured on an assumption of the stationary or fixed relationship between input features and target variables.

The problem of concept drift is genuine in an environment where underlying data distribution is not static. Variations in environmental patterns, user activities or any other factors can cause the change in data patterns, thereby making trained models ineffective with time. It becomes essential to handle concept drift to keep the efficiency and accuracy of machine learning models in dynamic cases.

Handling and adapting to concept drift is crucial for maintaining the performance of machine learning models as its occurrence degrades such models' results. Techniques to handle concept drift include using online learning algorithms that continuously update the model, employing ensemble methods that combine several models' predictions, or monitoring the model's performance over time to detect and react to changes. Firstly, we will discuss the nature of concept drift and then techniques for detecting it.

In data stream classification, we aim to find a function to predict the value of class labels for unseen data [1]. According to the Bayesian Decision Theory, to make a classification decision, for instance X, the class y can be represented as

$$p(y|X) = \frac{p(y)p(X|y)}{p(X)} \tag{2.1}$$

where $p(X) = \sum_{y=1}^{c} p(y_i)p(X|y_i)$ for all classes $y = 1, \dots, c$ and c is the number of classes. In this scenario, the concept refers to the target class [3]. The drift refers to the change in the joint probability over time. Formally, concept drift can be defined as

$$\exists X : p_{t_0}(X, y) \neq p_{t_1}(X, y) \tag{2.2}$$

where the t_0 and t_1 are two different time points, p_{t_0} denotes the joint probability of at time t_0 and respectively p_{t_1}. The concept drift may occur due to changes in variables of the posterior distribution, the distribution of one of the classes, or the class prior may change over time. We can distinguish several types of concept drift [1, 2]:

Real concept drift the posterior probability varies over time for one or more classes, and it can happen either with or without change in the evidence $p(X)$. This situation requires changes in the machine learning model to maintain performance. The real concept drift is modelled in the second graph of Figure 2.1.

Virtual concept drift the distribution of the input data changes without affecting the posterior probability. The decision boundary does not change, and it requires different handling techniques.

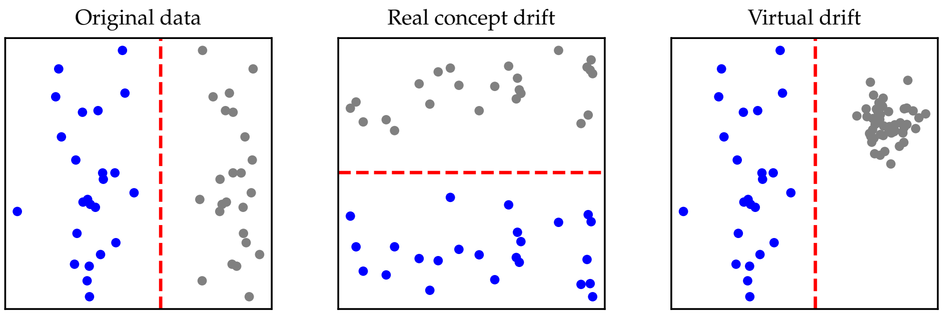

Figure 2.1: Types of drift: different colours represent classes [2]

Further in this paper, we will consider the real concept of drift as the primary focus. This change in probabilities can manifest in different forms over time [1]:

Sudden concept drift , also known as abrupt drift, occurs when there is a sudden and immediate change in the data distribution. Trained models need to adapt quickly to maintain performance

Incremental concept drift involves continuous and gradual changes in the data distribution. The relationship between the target variable and the input feature evolves slowly.

Gradual concept drift happens more abruptly. The change in the data distribution is more noticeable and occurs faster than in incremental drift.

Reoccurring concept drift refers to a situation when the data distribution follows a pattern of periodic change. A predictive model might need to remember these patterns to remain accurate.

An outlier is a one-off deviation or an anomaly in the data. No adapting is needed in this case.

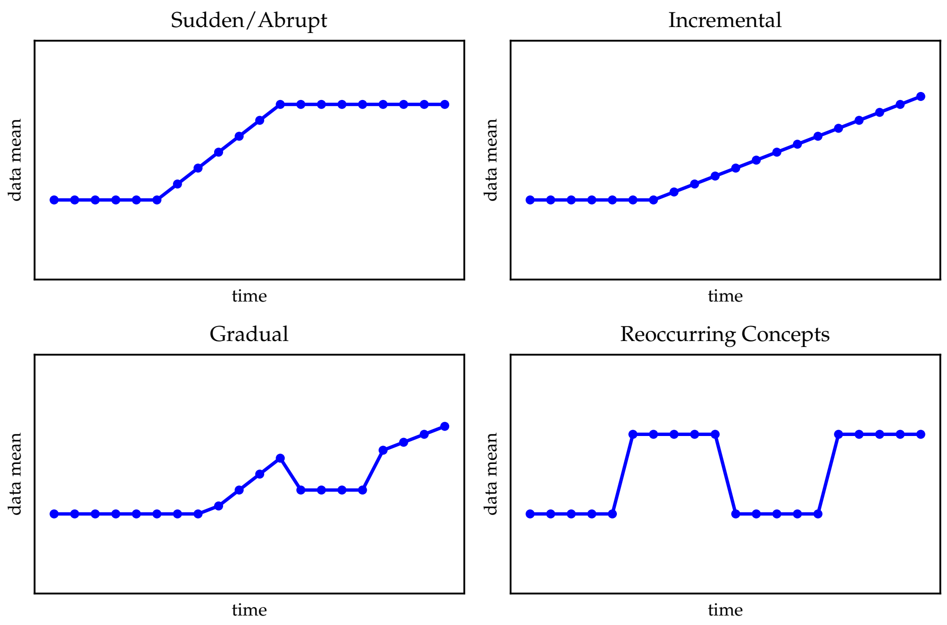

Figure 2.2: Data change over time

2.1 Adversarial drift

Changes in the probabilities can appear organically as the data streams evolve and new classes emerge or disappear. However, in the context of machine learning, particularly in cybersecurity, adverse drift refers to a scenario where the characteristics or nature of malicious attacks (the adversarial input) evolve or change over time [3]. This evolution often occurs in response to the defences employed by cybersecurity systems [4]. Unlike natural concept drift, which occurs due to evolving data over time, adversarial drift is the result of intentional data manipulation. The manipulation is malicious and is designed to evade or mislead classification models.

Most machine learning models assume that the statistical properties of their input data, such as distribution, will remain constant over time, which is inherent to them. However, adversarial drift challenges this assumption as the nature of the attacks changes. Consequently, the data the model was trained on no longer accurately represents the environment in which it operates. This discrepancy can lead to a decrease in the model's effectiveness in detecting or mitigating attacks.

The impact of adversarial drift on model performance is significant. As malicious tactics evolve, the ability of the machine learning model to detect or counter these attacks can diminish. This decrease in performance is due to the model's training data needing to be updated and relevant in the face of new types of threats. Cybersecurity systems and machine learning models must undergo continuous adaptation to counter this. They need to be regularly updated and retrained with the latest data that reflects the current nature of threats. This necessity leads to an ongoing cycle of adaptation, with defenders constantly striving to keep up with the evolving tactics of attackers.

In Detecting Adversarial Advertisements in the Wild [5], the authors focus on detecting low-quality or harmful adversarial advertisements in large online advertising systems. It highlights the high costs associated with false positives and negatives in this domain and describes a tiered detection strategy combining automated and semi-automated methods. They outline the main challenges in detecting adversarial advertisements:

- Data Skewness: The rarity of adversarial ads in the vast volume of legitimate ads creates a skewed dataset, complicating the detection process.
- Resource Allocation: Efficiently allocating expert human effort for semi-automated detection strategies is a significant challenge.
- Independent Assessment: Independently assessing the detection system's effectiveness is difficult due to the nature of adversarial attacks and the need for constant updates.
- False Positives and Negatives: Balancing the high costs associated with false positives (blocking legitimate ads) and false negatives (allowing harmful ads) is a critical challenge.

The authors mainly focus on the effectiveness of a tiered detection strategy in identifying adversarial advertisements. Their approach combines automated and semi-automated methods, leveraging large-scale machine learning and expert human judgment. They highlight the complexities and challenges in this domain, particularly the balancing act between reducing false positives and negatives and efficiently utilising resources. It underscores the necessity of continuous adaptation and improvement in detection strategies to keep pace with the evolving nature of adversarial attacks in online advertising, further proving the necessity of not ignoring the adversarial drift.

2.2 Concept drift detection

As discussed previously, in settings where predictive models are utilised, these models must be able to identify and adjust to changes in the data over time. Failing to do so can lead to a decline in their predictive accuracy. As data evolves, it may be necessary to either update the decision-making model with new data or completely replace it to suit the altered circumstances. For effective functioning, predictive models must [1]:

- Detect the concept drift and adapt accordingly.
- Differentiate between actual drifts and mere noise, ensuring adaptability to real changes while maintaining resilience against noise.

- Operate within a timeframe shorter than the interval at which new examples arrive and manage this without exceeding a predetermined memory allocation for any storage needs.

2.2.1 Explicit concept drift detection

Explicit concept drift detection, also called ”informed methods,” involves using triggering mechanisms to detect drifts in the data. These methods are particularly suited for situations where it is essential to identify anomalous activities or behaviours that are out of control. Such scenarios are typically approached as detection tasks, where the drift needs to be clearly signalled.

Informed methods are preferred in monitoring and control applications where understanding the occurrence and timing of drifts is crucial. These methods provide detailed information about the drifts, including their onset and duration. This approach is highly relevant in machine learning, where the explicit detection of drifts helps adapt models to changing data patterns more efficiently [1].

Methods in this category are particularly relevant in scenarios for detecting anomalies or out-of-control behaviours. The choice of the method usually depends on the intent behind handling the concept drift. It should involve these considerations:

- Monitoring the performance of the learner over time, mainly accuracy, F-measure, precision or recall. Values such as positive rate, false positive rate, true negative rate and false negative rate can also be incorporated into this technique [6]. A significant decline in metrics can indicate concept drift because the model becomes less effective as the underlying data distribution changes.

- Observing data distribution and its changes can be the first step to indicate concept drift. It involves monitoring changes in the data distribution with statistical analysis of the features and their distribution.

After a drift is signalled, the model is usually retrained from scratch using new data, but it can also be updated using a recent selection of data (data window). Post-adaptation, it is crucial to continuously monitor the model's performance to ensure that the adaptation was successful and to be on the lookout for any future drifts.

Explicit detection includes the Drift Detection Method (DDM) and Early Drfirt Detection Methodology (EDDM) [7]. They monitor the probability of error at the time and its standard deviation. Window-based distribution monitoring methodologies use a chunk-based or sliding window strategy over recent samples to identify changes. Instead of comparing individual samples, models calculate deviation by comparing the distribution of the current chunk to a reference distribution obtained at the beginning of the data stream from the training dataset. This allows for precise detection of the concept drift. However, additional memory is required to store distributions over time.

Adaptive Windowing (ADWIN) [8] is a notable detector introduced in 2007 that uses variable-length sliding windows.

2.2.2 Implicit drift detection methodologies

Contrary to explicit detection, implicit drift detection, also known as “blind methods”, relies on adapting or retraining the learner (or model) at regular intervals without explicit concept drift detection [1, 9]. The model in defined timeframes adjusts to align with the current class distribution. The adjustment is done regularly, without certainty whether the drift has occurred. These methods are particularly effective in environments where labelling data is impossible or expensive. However, the challenge for these methods is the potential for false alarms, where the drift did not manifest as a clear change in the data distribution.

Clustering-based methods are an approach for handling concept drift by analysing unseen data distribution. They group similar data points together and monitor how the clusters evolve. The online Novelty and Drift Detection Algorithm (OLINDDA) [10] employs K-means clustering to adapt continuously to emerging data distributions. By temporarily storing unknown data samples, it can periodically assess them to determine if the model should integrate them with the existing classes or if they should become a new one. The Woo Ensemble and ECSMiner methods also use the concept of micro-clusters, monitoring the density around new samples that fall outside existing clusters. A significant increase in density around these samples indicates a new concept, triggering model retraining and cluster readjustment. The strong aspect of methods based on clustering is identifying new patterns in the data streams and gradual drifts. However, they still might face some challenges when the parameters (number of clusters or the distance metric) are chosen suboptimally, or they may struggle in higher-dimension spaces.

Multivariate distribution monitoring methods focus on tracking the distribution of individual features within unlabeled data. These methods typically operate in chunks, storing condensed information about each training data chunk, such as histograms of binned values. This stored information serves as the reference distribution, enabling the detection of changes in the current data chunk. The main idea behind these methods is to compare the data distribution with the data from an earlier period. While they are robust, they are susceptible to changes in any of the features, even though they might not be relevant for detecting drifts. It is also important to train them on balanced datasets because, within limited information about minority classes, their feature change may be deemed insignificant due to their small proportion in the original dataset.

2.2.3 Examples of concept drift detection

MD3

This approach, named MD3, is a domain-independent technique [11]. Classification techniques rely on the ability to generalise from the training dataset, creating margins in prediction space where uncertainty exists. MD3 addresses the challenge of detecting concept drift in unlabeled data, a common scenario in real-world applications. Traditional drift detection methods rely heavily on labelled data to compute performance metrics such as accuracy or F-measure and monitor these over time. However, the dependency on labelled data could be more practical in many situations due to the cost and time involved in labelling.

In essence, the MD3 algorithm extends the notion of classifier uncertainty for detecting concept drift, developing it as an incremental streaming algorithm for drift detection from unlabeled data. Through comparison and empirical evaluation, it formulates margin density for classifiers with explicit margins and those without clear margins, such as decision trees. When a concept drift is massive yet does not affect the classifier's margin, the MD3 algorithm may face challenges in effectively detecting the drift. This scenario can occur in certain situations, such as:

- **Drift Occurring Away from the Margin:** If the drift happens in areas of the feature space far from the classifier's margin, MD3 might not pick it up. This is because MD3 focuses on changes within the margin or the classifier's region of uncertainty. If the concept drift does not influence this region, the margin density remains unaffected, and MD3 may not trigger an alarm.

- **Subtle Changes in Data Distribution:** In cases where the concept drift involves subtle changes in the underlying data distribution that does not directly impact the decision boundary (and thus the margin), MD3 may not detect the drift.

- **Drift in Non-Margin-Related Features:** If the drift occurs in aspects of the data that are not critical for the classifier's decision-making process (i.e., features not strongly associated with the margin), MD3 may miss it. This type of drift might not change the margin density, as it does not affect how the classifier generalises over the data.

This method utilises the concept of margin density in classifiers for detecting drift in data streams. It monitors the margin, which is the of uncertainty in the classification decision boundary. For classifiers without an explicit notion of margin (such as decision trees and K-means), the model resorts to the computation of Blindspot Density, which is the classifier's uncertainty region. After computing the region, the model computes a reference density value and uses it to evaluate incoming data.

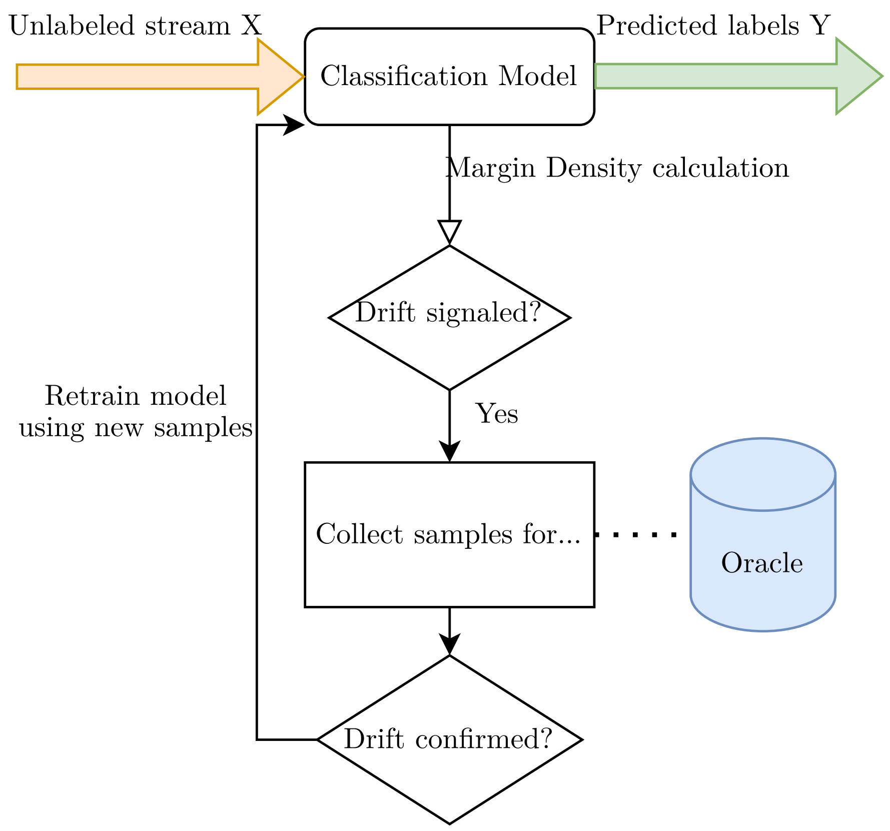

Figure 2.3: Basic MD3 flowchart

If the density of the margin deviates significantly from the expected margin density, MD3 signals a potential concept drift. Further, N samples are requested to be labelled by an external entity, Oracle, that provides true labels for data at a given cost. Then, an accuracy test is performed, and if the performance is degraded, a new model is trained using new labelled samples and a new value for margin density is established. This allows MD3 to monitor potential concept drift without labelled samples, as the detection process is unsupervised. Labelling from external entities is only requested when the drift is suspected, and the newly labelled data is also used in retraining.

Predict-Detect

The Predict-Detect model is a framework that addresses the issue with dynamic domains such as adversarial attacks [12]. It utilises a dual-component structure consisting of a predictor and a detector, each playing a vital role in maintaining the model's performance against possible attacks. According to the authors, the key to mitigating attacks on models is to design problem-specific solutions, but they also give a domain-independent solution. This method has the same authors as the MD3 approach and can be used in conjunction with the previous description of MD3.

The Predict-Detect design utilises two orthogonal classifiers, each trained on separate subsets of the training features. The predict component is a machine learning component with the primary task of classification applied to the data stream. Its main goal is to maintain high performance on incoming streaming data. The choice of the model and features depends on the specific task and the characteristics of the data stream. The authors anticipate that malicious actors will target this component after its deployment.

On the other hand, the detect component is responsible for identifying the concept drift and does not engage in prediction tasks. It serves as a watchdog for any anomalies occurring in the incoming data stream that may suggest drift. The detect component identifies drift by analysing the data for changes, often including shifts in the distributions of the features, emergence of new patterns or disappearance of old ones. When a drift is detected, the Detect component triggers a response in the system. It typically involves notifying the system to adapt, which may include retraining or tuning the Predictor. The second subset of features is used for the Detection component. Consequently, it remains protected from external adversarial probes. This model holds knowledge derived from the original training data, which remains inaccessible to the adversary through probing.

While the Detect component identifies concept drift, the Predict component must adapt to these changes. Upon receiving signals from the Detect component about a drift, the Predict component can be retrained or fine-tuned to align with the new data distribution. This retraining ensures that the predictive accuracy is maintained even as the underlying data changes. The Predict component works in tandem with the Detect component. While it focuses on prediction, the Detect component monitors for drift. This dual-component approach enhances the overall robustness of the system against adversarial attacks or natural drifts in the data.

The division proposed by Sethi and Kantardzic is based on a feature-ranking approach to ensure that each model is trained on essential and disjoint features. If there is no knowledge of the importance of the features, it is possible to partition them randomly. As the division of the features is not transparent for the adversary, it continues to probe using the entire feature set and depends on misrepresenting the significance of features for classification.

The framework utilises the disagreement between the predictions of the two models to indicate potential adversarial activity. An increase in disagreement suggests a drift in the data, potentially due to adversarial manipulation. This approach allows for the detection of adversarial drifts in an unsupervised manner, reducing the reliance on labelled data.

In this context, the term *Detection Region* refers to a specific area in the feature space where discrepancies between predictions of different models are observed. The Detection Region is used to identify adversarial activities in the Predict-Detect classifier framework. It is a zone where an increase in the number of samples indicates potential adversarial attacks on the model. This concept is crucial for the framework as it helps reliably detect concept drifts caused by adversarial attacks in an unsupervised manner, thereby allowing

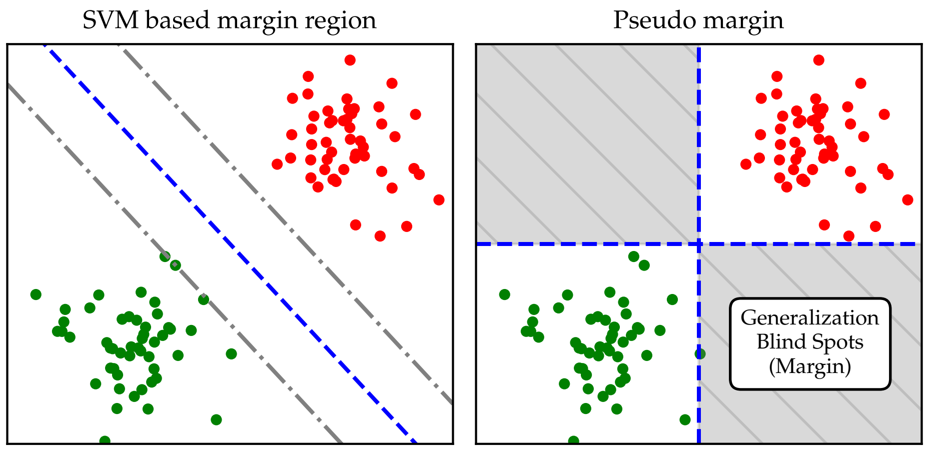

Figure 2.4: MD3 SVM-based classifier (left) compared to a classifier defined as the disagreement region (right).

for appropriate responses to maintain the effectiveness of the model. For each of the components, during the training stage, the framework learns expected disagreement and acceptable deviation along with the expected performance. For each new incoming data sample, the disagreement in predictions between the Predict and Detect models is computed. This is referred to as the signal Dis. This disagreement value is aggregated over time to form the P_{D_t} metric. The aggregation is guided by a time-decaying incremental tracking approach, which is influenced by the chunk size N. This means the framework continually updates the disagreement rate as new data comes in, providing a dynamic view of how the models' agreement levels change over time. If the disagreement rate suddenly increases, it indicates concept drift. After that, the attack is confirmed by collecting new samples and labelling them for a new dataset, which is later used to retrain the model. The model can generate a new split of features or keep the previous one.

RBM-DD

The RBM-DD (Restricted Boltzmann Machine for Drift Detection) [13] is a specialised application of the Restricted Boltzmann Machine (RBM) for detecting concept drift in data streams. This approach involves using the RBM's architecture and learning capabilities to identify changes in the data distribution over time.

The authors consider adversarial drift as malicious due to the intentional nature of such drifts. In the context of data streams and ML in cybersecurity, adversarial drift refers to changes in data distribution deliberately inducted by an attacker of an external entity, intending to compromise the performance of the defender's learning system. Unlike the natural concept drift, which often occurs primarily due to evolving data over time, adversarial drift is the result of intentional manipulation. It also always involves sophisticated strategies to mislead the classifier.

A Restricted Boltzmann Machine (RBM) is an artificial neural network used for unsupervised learning. It is a variant of the more general Boltzmann Machine, with a specific restriction on its architecture. It is a fully trainable drift detector designed for autonomous adaptation to the stream without external help, user-defined thresholds, or other statistical tests. The authors argue that alternative approaches are highly sensitive to any data perturbations and lack robustness against poisoning attacks. They focus on measuring the difference between data distributions without analysing the content of new instances. A significant challenge is differentiating between valid (non-malicious) concept drifts and adversarial (malicious) concept drifts. This is crucial for ensuring that learning algorithms adapt appropriately without being misled by adversarial attacks. The main task of RBM-DD is to solve these shortcomings:

An RBM consists of three layers of nodes: a visible layer and a hidden layer.

- The *visible layer* corresponds to the input data.
- The *hidden layer* is used to capture features or patterns from the input data.
- The *class layer* is used for class representation, typically implemented with one-hot encoding.

The neurons in the same layer are not interconnected but are connected to neurons in the adjacent layers. The training of this RBM involves a mini-batch approach with a gradient descent method, and it is designed to update its parameters based on the incoming data stream. This setup enables the RBM to detect concept drifts robustly, distinguishing between normal and adversarial changes in the data distribution.

The drift detection involves measuring the reconstruction error of the RBM, which indicates how well the RBM can reconstruct the input data. Significant changes in the reconstruction error are indicative of concept drift. The data similarity is calculated by reconstructing the error measure. For stability, the calculation is an average error over a recent min-batch of data. If the new mini-batch differs significantly from the previous one, the RBM-DD detect it by using the Granger causality test on trends. If the hypothesis that the Granger causality relation exists between mini-batches is rejected, then the model signals concept drift.

2.3 Datasets

Having a robust and representative dataset is crucial for effectively testing new models for concept drift, as the quality and diversity of the data directly influence the model's ability to learn and adapt to changing patterns over time. A good dataset ensures comprehensive coverage of possible scenarios, including edge cases, and mimics real-world dynamics where concept drift might occur.

This allows for a thorough evaluation of the model's sensitivity and responsiveness to drift and can reveal its strengths and weaknesses in adapting to shifts in the underlying data distribution. Without a well-constructed dataset that encapsulates the complexity and variability of practical applications, models might seem deceptively proficient in lab conditions but then falter in the face of actual, evolving data streams, leading to poor performance and reliability in real-world deployment. Essentially, a robust dataset for testing concept drift is indispensable for validating and refining models to ensure their resilience and longevity in dynamic environments.

2.3.1 Performance Metrics

The evaluation of data streams is one of the biggest challenges since traditional approaches are unsuitable for data stream scenarios. There are several requirements for such environments, it has to process an example at a time and go through it only once, it is limited in memory capacity and time, and it has to be able to predict at any time.

Authors of *Classifer Concept Drift Detection and the Illusion of Progress* [14] formally set properties for evaluating drift detection methods.

Mean Time between False Alarms (MTFA) is a metric that measures the average duration between incorrect warnings of drift in an environment where no such drift has occurred. The less frequent false alarms, the higher the MTFA, which indicates a better-performing drift detector.

Mean Time to Detection (MTD) is another measure that determines the response of a system to actual change. The lower MTD value is better because the detector quickly detects genuine drifts.

Missed Detection Rate (MDR) calculates the likelihood of the detector failing to signal an alarm when a drift has occurred.

For a fair assessment of change detection algorithms, the evaluation framework must have access to verified instances of change within the dataset—this is the ground truth. Therefore, we need to use synthetic datasets with established ground truth, which is essential for this purpose. One approach to evaluating the effectiveness of a detection method within a real-world dataset is to employ a classification technique that incorporates concept drift detection. By comparing the performance of the baseline classification model without drift detection to that of the same model with drift detection enabled, we can assess the impact and improvement brought about by the concept drift detection mechanism. This comparison highlights the value added by recognizing and adapting to concept drift in maintaining or enhancing model accuracy in dynamic environments. In this situation, we can use a combination of standard metrics for measuring classifier's performance as explored in *BODMAS* [15]. The article uses several metrics to measure the performance of concept drift detection methods for malware classifiers. The metrics they used are mentioned in Table 2.1

Table 2.1: Performance metrics for classification [15]

	Evaluation Parameter	Major Purpose
1	False Positive Rate	Keeping the FPR low is crucial for practical classifier applications, in this case, ensuring benign files are not incorrectly flagged as malware.
2	F1 Score	The F1 score is particularly useful for evaluating the overall effectiveness of a classifier in the presence of concept drift, as it considers both the precision (the ratio of true positive predictions to the total positive predictions) and the recall (the ratio of true positive predictions to the actual positives).
3	False Negative Rate	This measure helps identify how well the classifier can detect new samples of known classes and its ability to generalize to completely new classes

2.3.2 Synthetic Dataset

MOA, which stands for Massive Online Analysis [16], is a significant open-source software framework well-regarded in the data stream mining and analysing community. It facilitates real-time and batch processing of data streams. It is an essential tool for scenarios involving continuous data influx, such as sensor networks, online transaction monitoring, or any domain requiring on-the-fly analytics.

Central to MOA's capabilities is its suite of machine learning algorithms encompassing classification, regression, clustering, outlier detection, concept drift detection, and even recommender systems. This breadth of functionality allows it to address various data stream challenges.

Designed with extensibility at its core, MOA allows for seamless integration of new algorithms and experimentation, presenting a flexible environment for users to expand their capabilities. Visualization tools in MOA contribute to a deeper understanding of algorithm behaviours and data stream characteristics over time. These tools are instrumental in both the development of new algorithms and in the elucidation of streaming data properties for analysis.

Moreover, MOA includes tools for generating synthetic data streams, along with concept drift, enabling researchers to simulate specific scenarios to test and study the behaviour of streaming algorithms under controlled conditions.

MOA's widespread adoption in academia and industry is a testament to its utility. It serves as a potent research tool and is commonly used in educational settings to instruct students on the nuances of data stream mining and concept drift. Given its implementation in Java, MOA benefits from cross-platform compatibility, further extending its reach and applicability in the landscape of big data analytics. MOA interface is easy to use and extend for researchers's needs.

2.3.3 Real-world Datasets with concept drift

Testing on real data is fundamental for developing reliable machine learning models because it provides a window into the model's prospective performance in the real world. Real datasets come with the inherent complexity and unpredictability of the actual environment where the model will be deployed. This complexity includes noise, outliers, non-linear patterns, and other real-world anomalies that synthetic data may not accurately replicate.

Real data allows for the validation of a model's robustness and ability to generalize. While synthetic data can be useful for initial proof of concept and controlled experiments, it often lacks the variances associated with real-world data. Models trained and tested on too-clean data may perform well in the lab but can fail spectacularly in practice due to overfitting on overly simplistic patterns.

Furthermore, real datasets often contain a degree of concept drift naturally. Testing on such datasets enables the evaluation of a model's ability to adapt to shifts and changes over time, which is a crucial property for applications where the model must remain accurate as the world changes around it. For example, conditions change constantly in financial markets or predictive maintenance, and historical data may become less relevant, requiring models that can adjust and learn from more recent data.

Electricty dataset

The Electricity Market Dataset[1] is a widely used real-world dataset for evaluating concept drift detection methods. This dataset originates from the Australian New South Wales Electricity Market. In this market, prices and demand are influenced by both the market's demand-supply balance and participant bidding activities. The dataset covers 2016 days between 2015 and 2020, comprising measurements related to electricity prices and demand over time. It includes attributes like the date, time, demand, prices, and whether the price increased or decreased. The dataset is particularly valued for its natural concept drift characteristics. The electricity market is subject to various forms of drift, including cyclical daily and seasonal patterns, unexpected events like equipment failures or sudden demand spikes, and longer-term changes in the market, such as introducing renewable energy sources.

BODMAS

Our tests will utilise ”BODMAS: An Open Dataset for Learning-based Temporal Analysis of PE Malware” [15]. It was created to address limitations in existing Portable Executable (PE) malware datasets, as the authors deemed the need for more recent and timestamped malware samples available for

[1] `https://www.kaggle.com/datasets/aramacus/electricity-demand-in-victoria-australia`

the public. They aim to provide viable data for machine learning malware analysis, especially in studying concept drift and malware family evolution.

The dataset contains 57,293 malware samples and 77,142 benign samples, totalling 134,435. The samples were collected from August 2019 to September 2020. The dataset includes a wide variety of malware family information. Each sample comprises its SHA-256 hash, the original PE (obtainable from the authors directly), and the pre-extracted feature vector. The dataset provides ground-truth labels, curated malware family information, and first-seen time for a sample based on VirusTotal reports.

The authors conducted a preliminary analysis using the BODMAS dataset to illustrate the impact of concept drift on binary and multi-class malware classifiers. The study highlighted the challenges of previously unseen malware families, which led to an increased rate of false negatives in binary classifiers and issues in family classifiers in a real-world setting. In the next section, we will replicate the authors' findings and implement new concept drift techniques to mitigate the effects of concept drift.

Real data embodies the intricate and multifaceted nature of the environment in which a model is intended to operate. Testing on real data not only ensures that the model is practical and actionable but also instils confidence in its users that it will perform reliably when confronted with the full spectrum of challenges in real-world applications.

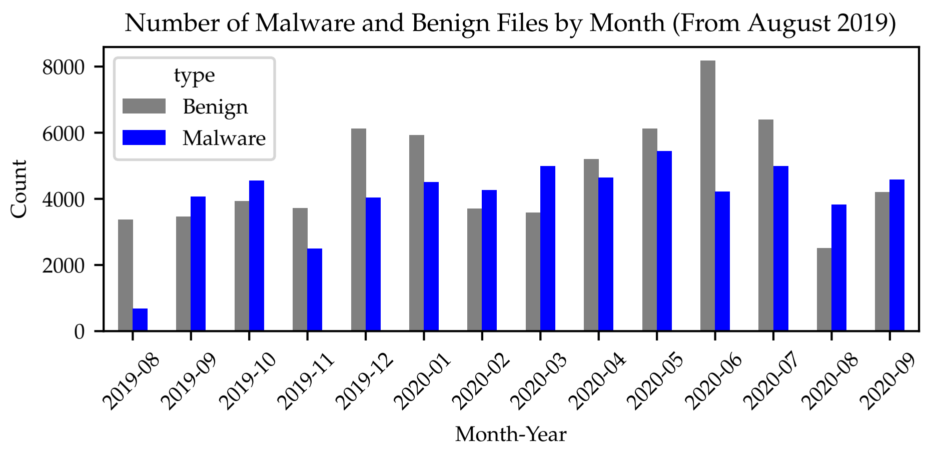

Figure 2.5: Number of Malware and Bening Files in the BODMAS dataset

2.4 Dealing with concept drift in BODMAS dataset

2.4.1 Motivation

The current literature describes several approaches for addressing the concept drift in a data stream. The primary goal is to find a viable solution to minimise the impact of concept drift in a given dataset. For this purpose, I will use the BODMAS malware dataset test as the input data stream, and the goal is to use as little additional computation and labelling as possible. Given the dataset's characteristics and the baseline model's high performance (detailed in the subsequent chapter), I will evaluate the effectiveness of different approaches based on the F_1 score while maintaining a False Positive Rate (FPR) of 0.1%.

2.4.2 Baseline models

Due to the evolving nature of malware and benign software over time, we can assume that the BODMAS dataset contains some concept drift. New malware families and variants emerge, leading to changes in the underlying data distribution. This evolution makes it necessary to constantly update datasets to maintain the accuracy and relevance of machine learning models used in malware analysis. Firstly, I recreated the findings from the original BODMAS article.

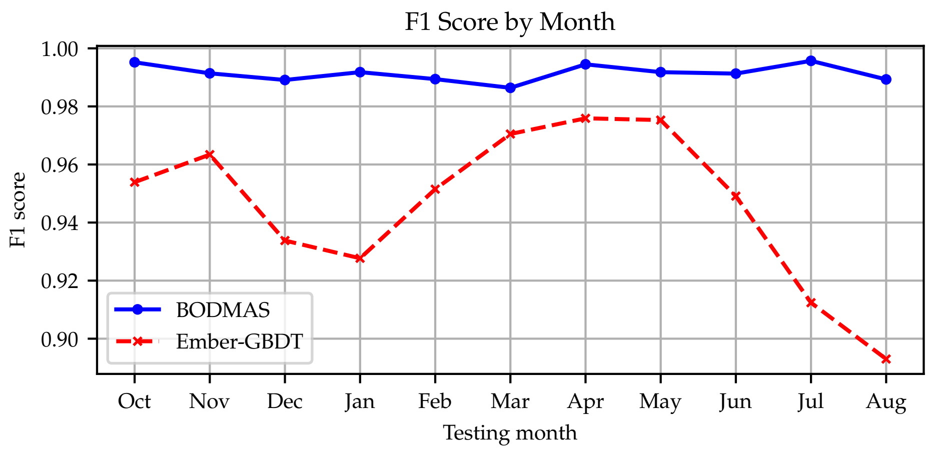

Figure 2.6: Training with the first month of BODMAS (until September 2019) compared to training only on EMBER dataset (2018)

Figure 2.6 presents the results of my experiment where I trained two models. The first model, the Ember-GBDT, is a Gradient Boosted Decision Tree (GBDT) classifier. This baseline was developed using the EMBER dataset[2], adhering to the hyperparameters specified in the original study. For the purpose of testing, the BODMAS dataset was divided into twelve monthly segments, against which the baseline model's performance was assessed. Consistent with the original study's methodology, each classifier's performance is evaluated based on the F_1 score, with a threshold set for the false positive rate (FPR) at under 0.1%, to ensure the classifiers' practicality. The results reveal that despite achieving a relatively high F_1 score, the classifier's performance varies and shows signs of decline over the test period, suggesting the occurrence of concept drift.

The observed decrease in my model's F_1 score when evaluated against later months, in comparison to the original article, could stem from varying

[2]Available at `https://github.com/elastic/ember`

monthly sample selection methods. The original study conducted multiple tests using a range of seeds to acquire an average F_1 score, whereas my evaluation was based on a single seed. This methodological difference led to a maximal performance difference of 2.84% in August. Nonetheless, the pattern of a performance dip in December and January, followed by a rebound in April, remained consistent with the original findings.

The second model used the same parameters but was trained with the data of the BODMAS dataset from September 2019. The performance of the newly trained is apparent. Labelling a month and retraining the model greatly improves the reliability of the classification, as new data provide needed information about new distributions and trends in features. The authors did not provide precise results for this model. My model performs with an average F_1 score at 99.1%.

2.4.3 The source of concept drift in BODMAS dataset

The source of concept drift in the BODMAS dataset arises primarily from the dynamic, evolving nature of malware and benign software. As new malware families and variants appear over time, they constantly introduce changes to the underlying data distribution. This dynamic change means that the models built on the dataset might become outdated as new data diverges from the data distribution on which the models were originally trained. If we know the underlying source of the drift, we can design better solutions to mitigate it.

Based on the performance of the baseline GBDT model in Figure 2.6, there are two notable declines in classifier accuracy, occurring around late December and again in August. Analyzing these dips in performance would provide an optimal strategy for identifying an effective concept drift detection method. Once these periods of significant change are understood, we can determine the most appropriate mitigation strategies. However, this does not guarantee that the next distribution disturbance will behave the same. The dataset distinguishes between benign and malware samples and supports that with family attribution of the sample.

Figure 2.7 shows the number of samples for the dataset's top 3 most common malware families. Throughout the dataset, the *sfone* group accumulates the number with the largest number of samples in June 2020. Compared to that *wacatac* malware family count is decreasing throughout the dataset. A classifier that detects malware and benign samples does not have to predict the malware family attribute accurately. However, this indicates that the underlying distribution of the malware class in the dataset changes over time. A method that detects concept drift has to be robust enough to distinguish between a real concept drift that would occur in changes of malware and benign samples and a virtual one that is prominent only in the malware class.

The other possible source for the concept drift might be new malware families. Figure 2.8 shows the amount of previously unseen malware families in a given month. The first three months were skipped as they were used as a

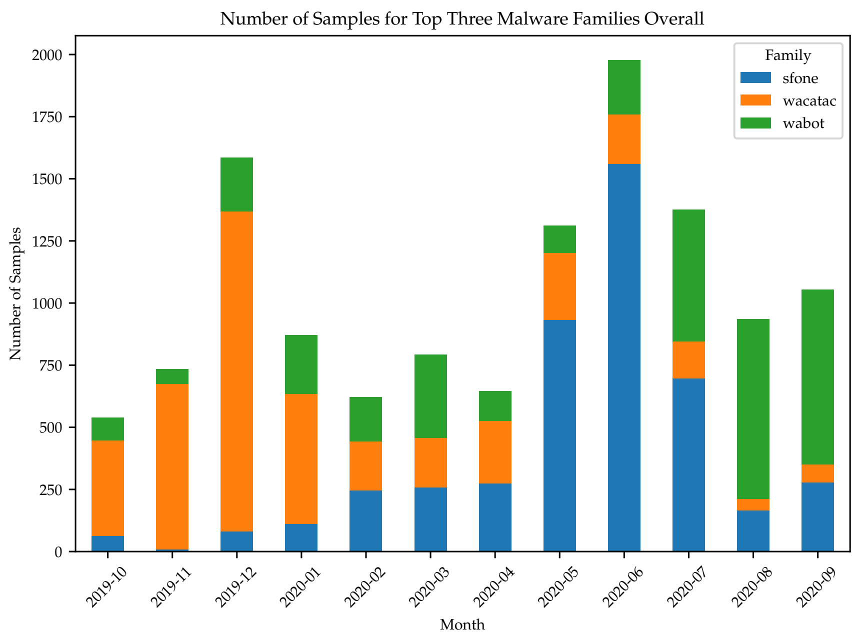

Figure 2.7: Number of samples for top three malware families in the BODMAS dataset

baseline for the graph. Within the whole dataset, new families are appearing. The arrival of previously unseen malware families can significantly increase false negatives in binary malware classifiers and challenge the accuracy of malware family classifiers in an open-world setting. This reflects the broader concept drift issue where the testing set distribution shifts away from the training set distribution, complicating the task of accurate classification over time.

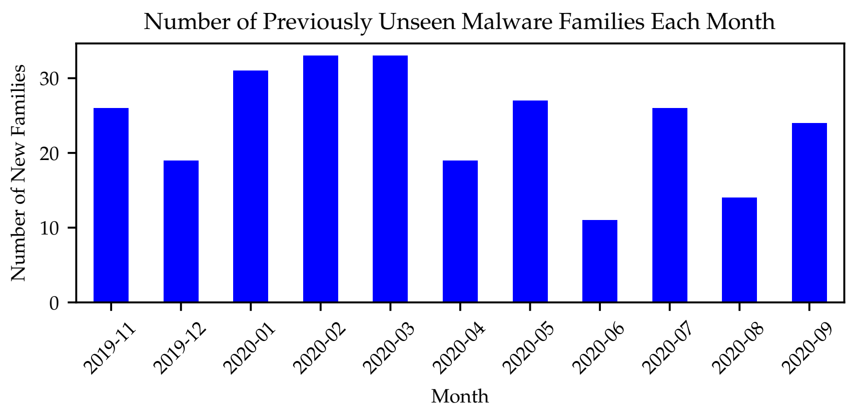

Figure 2.8: Number new malware families in each month

2.4.4 Incremental retraining

Incremental retraining is an easy strategy to implement for addressing concept drift in dynamic data environments, ensuring predictive models remain accurate as underlying data changes. This approach allows models to continuously learn and adapt by incorporating new data into the training process in real time. Incremental retraining adjusts the model's parameters or structure as new information becomes available, enhancing its responsiveness to new patterns and trends.

Incremental retraining can improve model robustness by effectively managing the stability, which helps maintain an optimal balance between adapting to new data and retaining valuable old information. This capability is critical for recognizing recurring patterns or cyclic changes, common in many real-world applications. The method's flexibility allows for adjustments in the frequency and intensity of updates, catering to the specific needs of various applications, particularly those requiring real-time data processing.

In the context of the BODMAS dataset, authors apply incremental retraining as a strategic response to concept drift in malware classification. This approach periodically updates the classifier by incorporating a small percentage (1%) of newly labelled data each month. The intention is to ensure the classifier remains current and effective despite evolving malware characteristics and distributions.

Incremental retraining offers several benefits in this scenario. First, it allows the classifier to adapt to new threats progressively, reducing the risk of becoming outdated. By integrating fresh data regularly, the model is better equipped to handle newly emerging malware families and variants, which might not have been present in the initial training set. This inclusion of new data helps maintain the classifier's accuracy over time, demonstrating a practical approach to mitigating the challenges posed by concept drift in dynamic threat landscapes like malware detection.

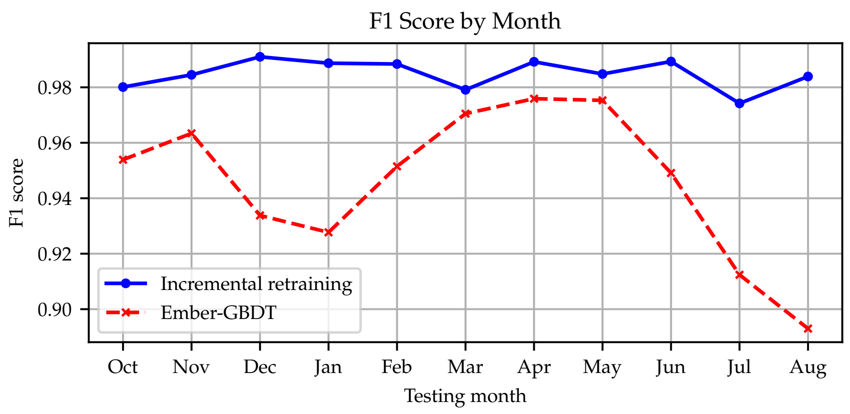

Figure 2.9: The baseline Ember-GBDT compared to the incremental retraining approach

For this experiment, the code from the original paper was reused. This strategy is easy to implement as sampling and labelling the data is the only overhead computation needed. The *probability* sampling strategy was used, where samples are ranked based on the current classifier's probability [17] output (the low probability ranks the sample higher) as it performed the best on the dataset. As seen in Figure 2.9, the incremental retraining strategy outperforms the baseline model trained on EMBER. There are clear improvements every month in the performance. This approach can reduce the integration difficulties with a real concept drift detection method and costs only one monhtly model retraining. This approach seems a viable alternative to a more specific detection and a domain-independent solution. In total 1027 new was needed in addition to the base model's dataset.

2.4.5 D3 with BODMAS dataset

The D3 (Discriminative Drift Detector) [18] method is an unsupervised approach to detecting concept drift in data streams. D3 addresses this challenge by employing a discriminative classifier to distinguish between old and new data within a sliding window, thus continuously monitoring and adapting to data distribution changes.

Specifically, D3 uses a sliding window technique to maintain a current subset of the data, divided into two sections: one containing older data and the other containing the most recent data. A discriminative classifier, such as logistic regression, is trained to differentiate these two datasets based on their features. The performance of this classifier is then evaluated using metrics such as the Area Under the ROC Curve (AUC). If the AUC exceeds a predefined threshold, the classifier can effectively distinguish between the old and new data, signalling a significant shift in the data distribution and identifying concept drift. This method allows for dynamic model adjustment to new data trends, ensuring the model remains accurate and robust.

As the input, the algorithm takes three hyperparameters that influence how a machine learning algorithm behaves, and they must be defined before training models. These are hyperparameters in the D3 setup:

- Size of the sliding window w: This setting determines the total number of samples in the sliding window. Larger windows may delay drift detection but reduce noise sensitivity.
- Proportion of new data ρ: its purpose is to specify the fraction of the window considered new data. The model can adapt quicker with higher values but is potentially more volatile.
- Threshold for AUC τ: the cutoff value for the AUC to determine if drift has occurred. Higher thresholds decrease false positives but may miss subtle drifts.

In their study, the authors introduce the D3 method as a robust solution for detecting concept drift in data streams, employing an unsupervised approach. A key advantage of D3 is its adaptability, as the sliding window ensures the classifier is continuously updated with the most relevant data, allowing the model to adapt dynamically to new patterns. This feature is crucial in environments where data properties frequently evolve. Furthermore, the unsupervised nature of D3, which does not rely on labelled data for detecting drift, makes it particularly beneficial in scenarios where obtaining timely labels is challenging, broadening its applicability. The authors also emphasize the practicality and efficiency of D3 and its customizability through adjustable AUC thresholds that allow users to balance the sensitivity and specificity of the drift detection according to their specific needs. The study concludes that D3 effectively and efficiently manages high-volume data streams, making it a viable and flexible tool for dynamic and evolving data environments.

My next experiment (code available at B) focused on testing the viability of the sliding window approach for detecting concept drift in the BODMAS dataset. I tested tuning the D3 with multiple parameters of w = [100, 200, 500, 2500], ρ = [0.1, 0.2, 0.3, 0.5], and τ = [0.8, 0.9, 0.99, 0.999]. The Algorithm 1 is a simplified description of the implemented code. For the discriminative classifier, I tested the linear regression and GBDT. During retraining the model. I utilized the EMBER dataset as a foundation and substituted 1% of newly observed data from BODMAS dataset.

Algorithm 1: D3: Discriminative Drift Detection

```
Input: Data stream $D$, window size $w$, new data proportion $\rho$, AUC
       threshold $\tau$
Output: Drift detection status
Function DiscriminativeDriftDetection($D, w, \rho, \tau$):
    Initialize sliding window $W$ of size $w$
    while new data available in $D$ do
        Add new data to $W$, maintaining size $w$
        Split $W$ into old data $W_{\text{old}}$ and new data $W_{\text{new}}$ using $\rho$
        Label $W_{\text{old}}$ as 0 and $W_{\text{new}}$ as 1
        Train classifier $C$ on $W$ to distinguish old from new data
        Compute AUC of $C$ on $W$
        if AUC of $C \geq \tau$ then
            Signal drift detected
            Retrain model
        end
        Remove oldest data from $W$ based on $\rho$
    end
    return Drift detection status
```

D3 results

As mentioned previously, to maintain some memory for the D3 algorithm and reduce its reactivity, the window size w is set to 2500. Values $\rho = 0.1$ and $\tau = 0.9$ were chosen for the other setting, as they performed the best on the BODMAS dataset. Although other combinations of values $w = 500$, $\rho = [0.1, 0.2]$ and $\tau = 0.999$ showed acceptable results, they exhibited generally lower performance than the original hyperparameters, prompting me to use the initial configuration.

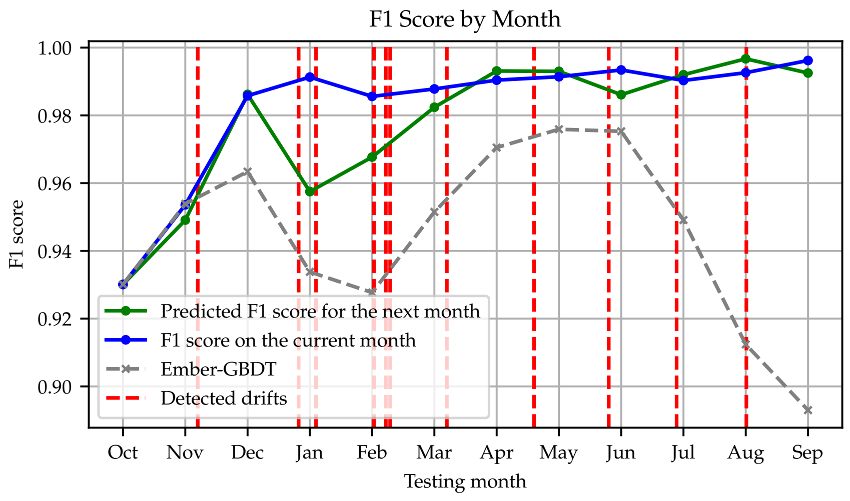

Figure 2.10: D3 results, red lines represent detected drift

D3 encounters limitations under certain conditions. In my experiments, I used the BODMAS dataset, which comprises samples and distributions from real-world scenarios, but the dataset was structured to balance benign and malicious samples evenly. This balance affects the sliding window's effectiveness in the D3 method, and thus, it performed better in the experiments than it would in a real-world situation. Specifically, if the stream lacks balance, it could lead to situations where all samples in the window belong to the same class. Consequently, new trends might develop within this set of new samples that do not reflect genuine differences between classes, potentially leading to incorrect drift detection. We can address this issue by adjusting the hyperparameter w (sliding window size) to an even larger value. Also, his method does not preserve long-term data, which may delay response to recurring drifts. This solution is more suitable for less complex and extremely fast environments, where a gradual or abrupt concept drift is more common.

2.4.6 MD3 with BODMAS dataset

The MD3 algorithm could be the practical and less demanding approach (than constant retraining) to detecting concept drift in the BODMAS dataset. I implemented the MD3 approach, and the code is available in Appendix B. In the next test, the dataset will be treated as a data stream and starting from October 2019, each month will be separated into 10 chunks of equal length and then sent to MD3 to evaluate for concept drift. As the GBDT does not have an explicit margin, it is necessary to use the feature bagging ensemble technique. The feature space is split between 15 GBDT classifiers, each containing 50% of the features. As part of the initialisation, a few metrics need to be calculated:

- The trained model C, along with its corresponding reference accuracy Acc_{Ref} and the standard deviation of accuracy σ_{Ref}. These metrics were then computed based on this initial model configuration of the baseline model trained on the EMBER dataset.
- The reference distribution MD_{Ref} and its deviation σ_{Ref} (also calculated from the baseline model).

The parameters that the user sets are as follows:

- Sensitivity Θ is used to fine-tune the algorithm. The suggested setting is in the range of [0, 3].
- The forgetting factor λ, it is set to usual $\lambda = (N-1)/N$ where N is chunk size.

The main classifier is still the same GBDT used in the BODMAS dataset. The MD3 algorithm processes an unlabelled data stream, denoted as X, using an initially trained model labelled C. It refers to a baseline known as the Reference distribution (MD_{Ref}), accompanied by its standard deviation σ_{Ref}, the reference accuracy (Acc_{Ref}), and the standard deviation of the accuracy (σ_{Acc}). The algorithm operates under defined parameters: a sensitivity threshold (denoted as Θ) and a forgetting factor rate (λ), which is calculated as $\lambda = (N-1)/N$ where N is the chunk size, N_{train} (N by default). The algorithm outputs a stream of predicted labels symbolized by Y.

Margin density with GBDT

The crucial part of using the MD3 algorithm is setting its margin density calculation. I stayed true to the BODMAS article and used its GBDT classifier in the tests. As the GBDT classifier does not have an explicit margin so in order to make it robust, a feature bagging ensemble technique was employed. The original 2381-dimensional space was split into subsets of features, and a classifier was trained on each. Tests were conducted on a split of 5, 7, 10, and 15 different subsets, where each contained 50% randomly selected features. The margin detectors that used 5 to 10 split were not sensitive enough for our use case. Using more than 15 different models is too impractical, as it drastically increases training and retraining time, so I settled for 15 subsets.

To calculate the actual margin density, I used Algorithm 3 that evaluates how often classifiers are unsure about their predictions. For each model, it selects relevant features from the new data, makes predictions, and checks if these lie within a defined range of uncertainty. It treats predictions within this range as uncertain or *in the margin*. It then calculates what fraction of the predictions are uncertain for each classifier and averages these fractions across all classifiers to get an overall margin density. Additionally, it calculates the standard deviation to measure how much the classifiers' uncertainties vary. The function returns this average uncertainty and its variability.

Algorithm 2: The MD3 algorithm

```
Input: Unlabeled stream $X$, Initially trained model $C$, Reference
       distribution $MD_{\text{Ref}}$, $\sigma_{\text{Ref}}$, $Acc_{\text{Ref}}$, $\sigma_{Acc}$
Data: Sensitivity $\Theta$, Stream progression $\lambda = (N-1)/N$ where $N$ is
      the chunk size, $N_{\text{train}}$ ($N$ by default)
Output: Predicted label stream $Y$
$MD_0 = MD_{\text{Ref}}$
$currently\_drifting = \text{False}$
$LabeledSamples = \emptyset$
for $t = 1, 2, 3, ...$ do
    Compute current margin density $MD_{cur}$
    Update $MD$, $MD_t = \lambda * MD_{t-1} + (1-\lambda) * MD_{cur}$
    if $MD_t - MD_{Ref} > \Theta * \sigma_{Ref}$ then
        // Drift Suspected
        $currently\_drifting = \text{True}$
        $LabeledSamples \leftarrow Collect\ N_{\text{train}}$ labeled samples
    end
    if $currently\_drifting$ and $|LabeledSamples| == N_{train}$ then
        // Enough labelled samples to make a decision
        if $Acc_{Ref} - (Acc(LabeledSamples) > \Theta * \sigma_{Acc}$ then
            // Drift Confirmed
            Retrain $C$ with $LabeledSamples$
            Update Reference distribution $(MD_{\text{Ref}}, \sigma_{\text{Ref}}, Acc_{\text{Ref}}, \sigma_{Acc})$
            $currently\_drifting = \text{False}$
        end
    end
end
```

Algorithm 3: Calculate Margin Density

```
Input: New samples $X_{\text{new}}$
Output: Overall margin density across all classifiers, Standard
        deviation of margin densities
Function CalculateMarginDensity($X_{new}$):
    $md\_list \leftarrow$ empty list
    for each classifier $clf$ in models do
        $probabilities \leftarrow clf.predict(X_{\text{new}})$
        $is\_in\_margin \leftarrow (probabilities >$
          $lower\_bound) \wedge (probabilities < upper\_bound)$
        Append mean of $is\_in\_margin$ to $md\_list$
    end
    $margin\_density \leftarrow$ mean of $md\_list$
    $std\_deviation \leftarrow$ standard deviation of $md\_list$
    return $overall\_margin\_density, std\_deviation$
```

Core of the MD3 approach

The MD3 Algorithm 2 continuously monitors an unlabeled data stream to detect concept drift, which occurs when the underlying data distribution changes. As each new data chunk arrives, the algorithm updates the margin density using an exponential smoothing technique, gradually incorporating the latest data while retaining some memory of past observations.

When the algorithm detects that the updated margin density exceeds a predefined sensitivity threshold, it raises an alert for potential drift. At this point, it collects a predetermined number of labelled samples to confirm whether the drift is real. Since I already had access to the labels, I utilized the upcoming batch of data as the set that an Oracle would typically label. If the model's performance on these newly labelled samples shows a substantial deviation from the baseline accuracy, the drift is confirmed. Once confirmed, the model undergoes retraining using these freshly labelled samples.

To effectively simulate a real data stream where labelling can be resource-intensive, I employed a strategy where 90% of the data used for training was from previously utilized datasets (the EMBER dataset and the already-used labelled BODMAS data), and 10% was from fresh data that hadn't been used in training before. This approach mimics the practical challenges and delays of labelling new data in a dynamic environment.

MD3 results

The most significant challenge involved setting the appropriate parameters for the algorithm. Sensitivity Θ is recommended to be set between values of 0 and 3. In this case, any setting with a sensitivity value over 0.5 would not detect a concept drift. Figure 2.11 shows how setting the sensitivity value to 0.1 and 0.25 affected the results. If the value were higher than 0.5, a concept drift would not be detected at all; on the other hand, if the value was set below 0.1 (e.g. 0.05 or 0.01), the drift was detected in every 2 to 3 chunks and then confirmed. However, after retraining the main classifier, the margin detector, the overall accuracy would not increase, and the F_1 score stayed around 99%.

Algorithm 3 outlines the general methodology for calculating the margin. A sample was deemed to be within the uncertainty margin if the prediction by the subset classifier fell within the range of [0.25, 0.75]. Adjusting the margin size had little effect on the performance of the detection method. The likely reason for this is that, unlike SVM, the GBDT classifier lacks a clear, intuitive definition of margin. However, its faster learning capabilities offset this, as GBDT requires fewer features for accurate predictions.

The results of the MD3 method are illustrated in Figure 2.11. This figure includes the baseline model's accuracy, which does not account for concept drift. Additionally, it shows the performance of two models under different conditions: one with a sensitivity threshold (Θ) of 0.25 and another with Θ of 0.1. The *validation* curve indicates the model's accuracy evaluated on data from the current month. The *test* curve represents how the model

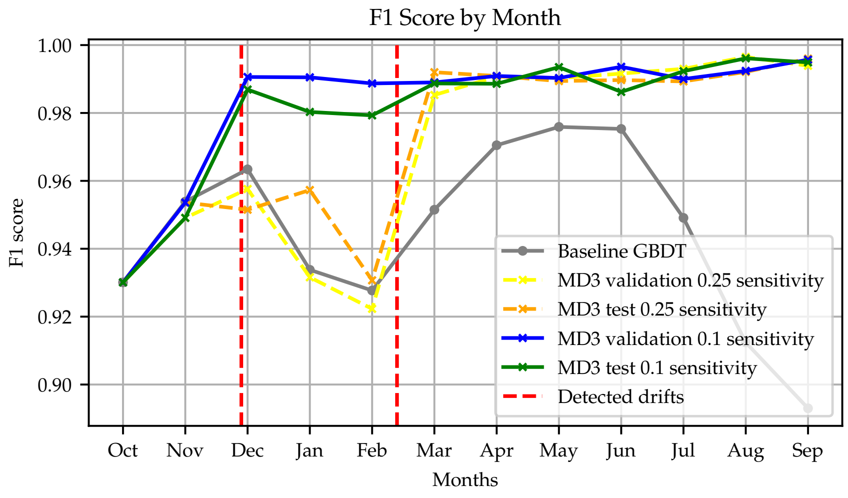

Figure 2.11: Results of the use of the MD3 algorithm on the BODMAS dataset

would behave when applied to data from the subsequent month, assuming no modifications are made to the model. When the model operates with a sensitivity setting of $\Theta = 0.25$, it fails to identify any drift until early February.

In contrast, when the sensitivity is adjusted to $\Theta = 0.1$, the model is able to detect the drift significantly earlier, in November. Following these detections, neither model registers further drifts, and the F_1 score stabilizes at 99%. This performance appears to be the upper limit for my GBDT model. Exploring alternative methods or deeper trees might be necessary if the goal is to achieve a higher F_1 score or to reduce the False Positive Rate (FPR).

The primary issue with this method can be seen during October and November of 2019. Despite the original model being trained on data from a year prior to the BODMAS dataset, it fails to detect any drift during these months, resulting in lower classifier accuracy. One potential solution is to lower the Sensitivity Θ further. However, this solution has a drawback—it may lead the model to require more frequent retraining, so we have to consider the cost efficiency of constant retraining without a substantial gain in accuracy. A more straightforward solution might be to gather a sample from October 2019 and use this data to retrain our models. This approach would address the accuracy decline in November and establish a new reference point for subsequent data. The likely reason for the initial accuracy drop is that the new data in the BODMAS dataset exhibit variations in features that the models, which monitor changes in margin density, fail to recognize until the variations become big enough to detect.

2.4.7 Improved approach for drift detection

The MD3 and D3 approach each have some weaknesses that can affect their effectiveness in concept drift detection. MD3, while providing a stable approach to drift detection, struggles with drifts occurring away from the classifier's margin. This method relies on changes within the classifier's region of uncertainty, so drifts outside this area may go undetected. On the other hand, D3 is highly sensitive and prone to detecting multiple drifts quickly, even immediately after retraining. This can lead to unnecessary retraining if the performance metrics are already satisfactory. D3's reliance on a sliding window for drift detection does not preserve long-term data, potentially delaying responses to recurring drifts.

To enhance the robustness of concept drift detection, I propose a hybrid approach combining the strengths of both D3 and MD3 methodologies. In this approach, D3 is a preliminary lower bound for detection, utilizing its sensitivity to flag potential drifts. When D3 triggers a concept drift, MD3 is employed to verify the drift. This two-tiered system ensures that a drift is confirmed only if both methods concur, thereby initiating the retraining of the models. This enhancement aims to improve the stability of the detection process and make it possible to do more fine-tuning for the BODMAS dataset.

Initial detection with D3

Due to its high sensitivity, the D3 (Discriminative Drift Detector) algorithm is used as the initial detector. D3 continuously monitors the data stream using a sliding window technique, comparing recent data with older data to identify potential changes in data distribution. The size of the sliding window and the proportion of new data are set to detect even subtle drifts. The purpose of this layer is to detect new emerging tendencies in the stream. For this case, the hyperparameters were set to $w = 1000$, $\rho = 0.2$ and $\tau = 0.9$.

Verification with MD3

When D3 triggers an alert indicating a potential concept drift, the MD3 (Margin Density Drift Detection) algorithm is activated to verify the drift. MD3 is configured with only five sub-classifiers to reduce computational overhead, but its margin sensitivity is increased to ensure thorough validation. Each sub-classifier operates on a different subset of features, enhancing the detection accuracy by focusing on various aspects of the data. Sensitivity Θ is lowered to 0.2, and bounds for the region of uncertainty are set to 0.2 for the lower bound and the upper bound to 0.8.

The most significant modification to the algorithm is the introduction of adaptive reference accuracy. In both MD3 and my improved approach, the final check involves evaluating the model's accuracy on the current data. In the BODMAS dataset, the drift is actually improving the accuracy when a classifier is trained on the EMBER dataset. That means that accuracy-based detection won't detect drift until a base model's accuracy starts to drop. In my improved approach, when drift is suspected by D3 and then by MD3, the reference accuracy will be recalculated based on the latest data. This ensures that reoccurring drifts will be detected and retraining will begin sooner.

Consensus for Retraining

For a concept drift to be confirmed, both D3 and MD3 must agree that a drift has occurred. When D3 detects a drift, it signals a new trend in the data, though this might only represent a virtual drift. To address this, MD3 is employed as a secondary check, utilizing older data to train models for drift detection. While D3 ensures the presence of change in new data, MD3 will not trigger retraining unless there is a confirmed change in the data distribution. If both algorithms agree to detect a drift, the models are retrained using the newly labelled data, ensuring they remain up-to-date and accurate.

Results

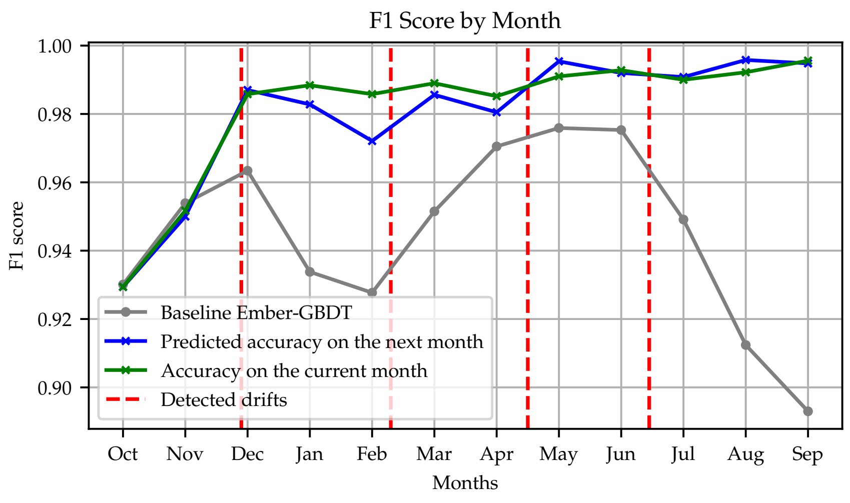

Figure 2.12: Improved detection method results, red lines represent detected drifts

By leveraging the sensitivity of D3 for initial detection and the verification strength of MD3, the improved method ensures that only confirmed drifts trigger model retraining. This dual-check system minimizes the risk of overfitting due to frequent retraining while maintaining high detection accuracy. I achieved the results in Table 2.12. The average F_1 score stayed at 98.1% and did not see many improvements compared to running detection methods separately with optimal hyperparameters. However, drift detection has become more reliable. The improved method detected drifts 4 and needed 28847 samples to achieve this. It correctly signalled a drift in June (where the baseline Ember-GBDT's accuracy drops). This is beneficial if we strive to detect all the drifts and not rely on the model's accuracy. Unfortunately, the improved method does not detect a drift in October and November (a drift is detected in late December, same as only running MD3), which again affects the overall accuracy of the model in these months.

Running methods separately

I ran the MD3 and D3 methods separately with the same settings to test that the proposed method benefits from combining them. Alone, the D3 approach would use 19521 samples and trigger 20 retraining with an average F_1 score of 98%. MD3 used 48923 samples and triggered retraining 5 times, with an average F_1 score of 98.1%. The methods perform worse than shown in Table 2.2 because the hyperparameters were set suboptimally when the models were run separately. However, when combined, the methods yield better performance.

2.4.8 Result analysis

Malware classification is a complicated problem that does not have a straightforward solution. Real-world datasets are messy and are a good benchmark for testing new methods. Compared to other classification tasks, malware is malicious in nature and tries to evade detection.

Table 2.2: Results of different approaches to concept drift

Method	Average F1 Score	Drifts Detected	Samples labelled
Baseline	0.946	0	0
New Data	0.991	0	22407
Incremental Retraining	0.985	12	1027
D3	0.982	11	10737
MD3	0.983	2	32742
Improved method	0.981	4	28847

Table 2.2 represent scores for approaches used in testing. To reiterate, the table describes these methods:

- **Baseline:** This model was trained on the EMBER dataset; it does not detect or retrain itself during classification.
- **New Data:** The model was trained on a combination of old data from the EMBER dataset and new data from the BODMAS dataset, which better represents distributions in the evaluation set. It does not detect drifts.
- **Incremental retraining:** In this approach, the model was retrained monthly with an extracted sample from the last month. It does not detect drifts, so the number under *Drifts Detected* column represents the number of retraining during the evaluation.
- **D3:** This uses the sliding window method and detects drifts based on the separability of new and old data.
- **MD3:** an ensemble-based approach where the drift is detected by the uncertainty of classifiers.

- **Improved method:** combination of the D3 and MD3 approaches.

By examining the average F_1 score (with FPR maintained at 0.1% or lower), it is evident that each method enhanced the classifier's accuracy, demonstrating that any form of concept drift management yields better results. The most effective approach involves using a completely new classifier built with a substantial amount of recent data. This highlights the importance of planning ahead to gather a current dataset for training classifiers. However, a drift detection method is the next best alternative if creating such a dataset is impossible.

Interestingly, incremental retraining emerges as a highly effective solution. This method's strength lies in its independence from prior knowledge about the data stream and its distribution. Even retraining with a small, 1% monthly sample size yields diminishing returns. Incremental retraining is an excellent initial strategy when creating a new classifier with a limited understanding of the data stream. However, a significant challenge of incremental retraining is ensuring that the sampled data comprehensively represents all data stream features.

The other two methods, MD3 and D3, accurately identified a likely drift in January 2020 (shown by Figure 2.11 and Figure 2.10) and triggered a retraining process. However, these approaches fell short as they failed to detect drift in October, negatively impacting the average F_1 score. Excluding the first two months, the average F_1 score for MD3 is 99.1%, and for D3, it is 99.0%. Another critical metric is the number of drifts detected. D3 is significantly more sensitive, detecting multiple drifts in a month even right after retraining. As the F_1 score exceeded 99%, this was not particularly necessary, but it is important to note that if a lower FPR rate (e.g., 0.001%) were desired, this frequent retraining might be deemed necessary. On the other hand, MD3 presents a more stable approach to concept drift detection. Despite achieving similar overall accuracy, MD3 only identified 2 drifts but required the training of 15 additional classifiers on sub-spaces to function effectively, which did not significantly reduce computational overhead.

Several key observations can be made when comparing the improved approach to MD3 and D3. The improved method, combining MD3 and D3, achieves a stable F1 score of 98.1%. This performance metric indicates that while the F1 score did not significantly improve compared to using MD3 and D3 separately, the reliability of drift detection increased. The improved method required fewer samples than MD3 and fewer retraining than D3, making it a balanced solution between reactive and stable approaches. However, more testing is required to determine if there is a real benefit to the combination of D3 and MD3.

Chapter 3

Discussion

When choosing an effective concept drift detection method, several factors are essential, including desired accuracy, responsiveness to drift, resource availability, and the depth of understanding of the data. A proactive approach is encouraged to ensure optimal results.

For scenarios where resources are abundant and the latest data can be accessed, constructing a new classifier using a comprehensive and recent dataset is the most robust option. This method requires considerable planning and resource allocation but leads to significant improvements in performance, as reflected by higher F_1 scores.

Incremental retraining represents a potent alternative, especially useful when new datasets are limited or when there is a need for the classifier to adapt swiftly to new trends without a full grasp of the data distribution. While effective, ensuring that the sampled data adequately represents the entire data stream is crucial.

D3 is highly sensitive among the detection algorithms, making it appropriate for environments where capturing every potential drift is crucial. This method may trigger multiple retrainings quickly, which is beneficial for maintaining extremely low false positive rates. However, if the performance metrics are already satisfactory, it might lead to unnecessary adjustments. Conversely, MD3 offers a more stable approach with fewer detected drifts, requiring the training of additional classifiers on sub-spaces, which could be computationally demanding. Its stability is advantageous for applications where abrupt changes in model behaviour due to frequent retraining are undesirable.

In conclusion, selecting a concept drift detection method should align with the application's specific needs, balancing stability, accuracy, and responsiveness. A proactive approach in selecting and implementing these methods can significantly enhance system performance by ensuring the model remains effective over time despite the evolving nature of data streams.

Chapter 4
Conclusion

In this thesis, I examined various methods to detect concept drift in data streams, with a specialised test focusing on their cybersecurity application. The exploration spans a spectrum of methodologies, from developing entirely new classifiers based on fresh datasets to incrementally retraining existing models and employing sophisticated drift detection algorithms like MD3 and D3.

The central takeaway is the importance of proactive adaptation in maintaining the effectiveness of machine learning models amidst dynamically changing data streams. The research underscores that every approach—constructing new classifiers or incrementally adapting existing ones—enhances classifier accuracy. The findings indicate that the most successful strategies involve substantial updates with new data or the vigilant application of drift detection methods when updates are not feasible.

For the best results, it is important to consider a few points. Firstly, regularly monitoring for concept drift is essential to maintain and improve the accuracy of machine learning models. Without it, models may become outdated and less effective as the underlying data evolves. Secondly, choosing the most effective concept drift detection method requires a deep domain understanding. Different domains may exhibit different drifts, and understanding these nuances helps select the appropriate detection strategy. Lastly, employing a concept drift detection mechanism becomes necessary if constant retraining is not desirable. This approach ensures that the model adapts to changes only when necessary, reducing computational overhead and maintaining stability.

My analysis suggests that incremental retraining is a viable initial strategy despite its challenges in sample representation, especially when detailed historical data knowledge is limited. This method's flexibility in data independence and its capacity for gradual adaptation aligns well with the needs of dynamic environments where data properties can swiftly change.

Among the specialized techniques studied, the MD3 offers method stability in detection. Conversely, D3 is highlighted for its high sensitivity, which is beneficial in settings where capturing subtle drifts is crucial, though it may lead to frequent unnecessary retraining. My approach lies between these two and offers a more balanced approach to drift detection.

In this thesis, my primary contribution to the field of concept drift detection lies in developing and validating a hybrid detection approach that combines the strengths of MD3 and D3 methodologies. The experimental results on the BODMAS dataset demonstrate an improvement compared to the MD3 and D3. While the proposed hybrid detection approach shows promising results, further tests are needed to evaluate its performance across datasets and real-world scenarios. Additional experimentation will help define the model parameters and ensure applicability in different applications.

Conclusively, this thesis advances my understanding of concept drift detection in cybersecurity applications and provides a solid framework for selecting appropriate detection strategies based on specific needs and constraints. It was challenging to research all the options and choose a method as the current research lacks real-world testing on datasets that are less common in the machine learning sphere. I gained a lot of experience working with machine learning libraries for Python and developing my testing methods.

4.1 Future work

Two main paths exist for advancing this research. Firstly, the development of a drift detection method designed specifically for the BODMAS dataset should be considered. Existing methods, which rely on older data, failed to identify drifts early, significantly affecting the classifier's performance. A tailored approach could potentially yield better outcomes. However, the scarcity of literature addressing concept drift in malware data streams underscores the need for further research into the characteristics of malware and its evasion techniques, which would facilitate the creation of improved detection methods.

Secondly, future research could explore integrating multiple concept drift detection strategies, with or without including incremental retraining. Certain data streams necessitate responsive solutions combined with classifiers that maintain stable long-term memory to achieve optimal effectiveness. Such investigations could lead to more robust and adaptive systems capable of effectively managing the dynamic nature of data streams.

Appendix A

Bibliography

[1] Imen Khamassi et al. “Discussion and review on evolving data streams and concept drift adapting”. In: *Evolving Systems* 9 (Mar. 2018). DOI: 10.1007/s12530-016-9168-2.

[2] João Gama et al. “A Survey on Concept Drift Adaptation”. In: *ACM Computing Surveys (CSUR)* 46 (Apr. 2014). DOI: 10.1145/2523813.

[3] Tegjyot Singh Sethi and Mehmed Kantardzic. “Data driven exploratory attacks on black box classifiers in adversarial domains”. In: *Neurocomputing* 289 (May 2018), pp. 129–143. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2018.02.007. URL: http://dx.doi.org/10.1016/j.neucom.2018.02.007.

[4] Łukasz Korycki and Bartosz Krawczyk. *Adversarial Concept Drift Detection under Poisoning Attacks for Robust Data Stream Mining*. 2020. arXiv: 2009.09497 [cs.LG].

[5] D. Sculley et al. “Detecting adversarial advertisements in the wild”. In: Aug. 2011, pp. 274–282. DOI: 10.1145/2020408.2020455.

[6] Indrė Žliobaitė. “Change with Delayed Labeling: When is it Detectable?” In: *2010 IEEE International Conference on Data Mining Workshops*. 2010, pp. 843–850. DOI: 10.1109/ICDMW.2010.49.

[7] Manuel Baena-García et al. “Early Drift Detection Method”. In: (Jan. 2006).

[8] Albert Bifet and Ricard Gavaldà. “Learning from Time-Changing Data with Adaptive Windowing”. In: vol. 7. Apr. 2007. DOI: 10.1137/1.9781611972771.42.

[9] Tegjyot Singh Sethi and Mehmed Kantardzic. *On the Reliable Detection of Concept Drift from Streaming Unlabeled Data*. 2017. arXiv: 1704.00023 [stat.ML].

[10] Shon Mendelson and Boaz Lerner. “Online Cluster Drift Detection for Novelty Detection in Data Streams”. In: *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. 2020, pp. 171–178. DOI: 10.1109/ICMLA51294.2020.00036.

[11] Tegjyot Singh Sethi and Mehmed Kantardzic. *On the Reliable Detection of Concept Drift from Streaming Unlabeled Data*. 2017. arXiv: `1704.00023 [stat.ML]`.

[12] Tegjyot Singh Sethi and Mehmed Kantardzic. "Handling adversarial concept drift in streaming data". In: *Expert Systems with Applications* 97 (May 2018), pp. 18–40. ISSN: 0957-4174. DOI: `10.1016/j.eswa.2017.12.022`. URL: `http://dx.doi.org/10.1016/j.eswa.2017.12.022`.

[13] Łukasz Korycki and Bartosz Krawczyk. *Adversarial Concept Drift Detection under Poisoning Attacks for Robust Data Stream Mining*. 2020. arXiv: `2009.09497 [cs.LG]`.

[14] Albert Bifet. "Classifier Concept Drift Detection and the Illusion of Progress". In: *Artificial Intelligence and Soft Computing*. Ed. by Leszek Rutkowski et al. Cham: Springer International Publishing, 2017, pp. 715–725. ISBN: 978-3-319-59060-8.

[15] Limin Yang et al. "BODMAS: An Open Dataset for Learning based Temporal Analysis of PE Malware". In: May 2021, pp. 78–84. DOI: `10.1109/SPW53761.2021.00020`.

[16] Albert Bifet et al. "MOA: Massive Online Analysis, a Framework for Stream Classification and Clustering". In: *Proceedings of the First Workshop on Applications of Pattern Analysis*. Ed. by Tom Diethe, Nello Cristianini, and John Shawe-Taylor. Vol. 11. Proceedings of Machine Learning Research. Cumberland Lodge, Windsor, UK: PMLR, Mar. 2010, pp. 44–50. URL: `https://proceedings.mlr.press/v11/bifet10a.html`.

[17] Dan Hendrycks and Kevin Gimpel. *A Baseline for Detecting Misclassified and Out-of-Distribution Examples in Neural Networks*. 2018. arXiv: `1610.02136 [cs.NE]`.

[18] Ömer Gözüaçık et al. "Unsupervised Concept Drift Detection with a Discriminative Classifier". In: *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. CIKM '19. Beijing, China: Association for Computing Machinery, 2019, pp. 2365–2368. ISBN: 9781450369763. DOI: `10.1145/3357384.3358144`. URL: `https://doi.org/10.1145/3357384.3358144`.

A.1 Used software

Under the *Methodological guideline No. 5/2023*[1], the following AI software was used in creating of this thesis:

- Grammarly[2] for grammar correction throughout the whole thesis.

[1]https://student.cvut.cz/sites/default/files/content/d1dc93cd-5894-4521-b799-c7e715d3c59e/en/20221109-methodological-guideline-no-52022.pdf

[2]`https://app.grammarly.com/`

- ChatGPT[3] for rewriting and rephrasing sentences throughout the whole thesis.
- DeepL Translate[4] for translations and rephrasing.

[3] `https://chat.openai.com/`
[4] `https://www.deepl.com/`

Appendix B
Attachments

This chapter describes the attached files to the thesis.

Table B.1: List of Attached Files

Folder name	Description
thesis-code	Source code used in the experiments.

B.1 Supplementary code

All code for used methods can be obtained from my GitHub page[1]. To run the code it is necessary to download the BODMAS datset[2][15] and put it under the *multiple_data/bodmas_dataset.npz* folder. Some of the experiments require a trained classifier on the EMBER dataset as a baseline. It is possible to run training on GPU, but I encountered a problem with the LightBGM library, where the training crashes with insufficient VRAM. This can be circumvented by using a fraction of the EMBER dataset. A more detailed description of flags and settings can be found on the GitHub page.

Training on new data

```
bodmas_main.py --training-set bodmas --diversity no --setting-name
bodmas_diversity_no --classifier gbdt --testing-time 2019-10,2020-
09 --retrain 1 --seed 0 --quiet 0
```

Description: This code replicates the findings in Figure 2.6. This code was taken from the original BODMAS article.

Incremental Retraining

```
concpet_drift_ember.py --setting-name ember_drift_random_improved
--classifier gbdt --month-interval 1 --testing-time 2019-10,2020-
09 --ember-ratio 1 --seed 1 --sample-ratio 0.01 --retrain 0 --quiet
0
```

[1]`https://github.com/barabashevd/thesis-code`

[2]Aviable at `https://whyisyoung.github.io/BODMAS/`

Description: This command runs the concept drift detection using the Ember dataset with specified settings. This code was taken from the original BODMAS article.

D3

```
d3.py --w 2500 --rho 0.1 --tau 0.99
```

Description: This command runs the concept drift detection using the D3 approach (results of Figure 2.10).

MD3

```
concept_drift_md3.py --setting-name md3 --classifier gbdt --month-
interval 1 --testing-time 2019-10,2020-09 --ember-ratio 1.0 --seed
1 --retrain 1 --quiet 0
```

Description: This command runs the MD3 concept drift detection method with the BODMAS dataset. A baseline model trained on EMBER needs to replicate Figure 2.11.

Improved detection

```
concept_drift_improved.py --setting-name ember_drift_random_improved
--classifier gbdt --month-interval 1 --testing-time 2019-10,2020-
09 --ember-ratio 1.0 --seed 1 --sample-ratio 0.01 --retrain 1 --quiet
0
```

Description: The command runs the improved drift detection based on MD3 and D3. This will result in Figure 2.12.