وزارة التعليم العالي والبحث العلمي

Université Badji Mokhtar Annaba

Faculté:Technologie Département : Informatique

Domaine : Mathématique-Informatique

Filière: Informatique

Spécialité: Système Informatique et Decision (SID)

جامعة باجي مختار – عنابـــــــة

Mémoire

Présenté en vue de l'obtention du Diplôme de Master

Thème:

FairWeighted Dynamic Ensemble Based on Hybrid Drift Detection and Champion Selection

Présenté par: Laouabida Sellami Mohamed Slimene

Encadrant: Pr Cherif Tolba — Université Badji Mokhtar-Annaba

Jury de Soutenance:

Cherif Tolba	Pr	Université Badji Mokhtar-Annaba	Encadrant
Sari Toufik	Pr	Université Badji Mokhtar-Annaba	Examinateur
Yakoubi Mohamed Amine	MCB	Université Badji Mokhtar-Annaba	Président

AnnéeUniversitaire:2023/2024

Dédicace

“

DU profond de mon cœur Je dédié ce modeste travail à tous ceux qui me sont cher,

*À mon père **Zinnedine** pour son encouragement,*

*À ma maman **Linda** pour tous ses sacrifices ,*

*À mon cher frère **Nazim** et a tous mes amis spécialement **Jed , kwilli et Timimoun** et a toute la famille Pour leur présence et leur soutien qu’ils m’ont apporté. Avec toute mon affection et ma reconnaissance.*

”

-Mohamed Slimene

Remerciements

Tout d'abord, je remercie Allah le tout puissant de m'avoir donné le courage et la patience nécessaires à mener ce travail à son terme.

La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes à qui je voudrais témoigner toute ma gratitude.

Je voudrais tout d'abord adresser toute mes sincères remerciements à mon encadreur ***M. Cherif Tolba*** *pour m'avoir encadré et orienté .*

Je souhaite aussi remercier l'équipe pédagogique et administrative de l'Université de Badji Mokhtar-Annaba pour leurs efforts dans le but de nos offrir une excellente formation.

J'adresse aussi mes remerciements aux ***membres du jury****, pour leur rigueur scientifique, leur professionnalisme et le sérieux lors de l'évaluation de ce travail.*

Et toutes les personnes qui ont contribué de près ou de loin au bon déroulement de ce travail, qu'elles voient en ces mots l'expression de notre gratitude pour leur présence, pour leur dévouement et pour l'aide inestimable qu'elles nous ont apportées tout au long de ce parcours. Un petit bout de chemin certes, mais un grand enrichissement.

Dedication 1

acknowledgement 2

Table of content 3

Abstract 7

3 Introduction 8

4 Concept Drift

4.1 Definition of Concept Drift 9

4.2 Historical Background

4.2.1 Early Developments 9

4.2.2 Development in the 1990s 10

4.2.3 Advances in the 2000s 10

4.2.4 Recent Developments 11

4.3 Types of Concept Drift

4.3.1 Sudden Drift 11

4.3.2 Incremental Drift 12

4.3.3 Gradual Drift 12

4.3.4 Recurrent Drift 12

4.3.5 Blip Drift 13

4.4 Effects of Untreated Concept Drift

4.4.1 Decreased Accuracy 13

4.4.2 Model Misinterpretation 13

4.4.3 Increased Error Rates 13

4.4.4 Customer Dissatisfaction 13

4.5 Strategies for Handling Concept Drift

4.5.1 Data Monitoring and Analysis 14

4.5.2 Model Updating and Retraining 14

4.5.3 Ensemble Methods 14

5 Related Works

5.1 Dynamic Classifier Selection for Drift Detection (Pinho and Santos, 2012)

5.1.1 Dynamic Classifier Selection (DCS) 15

5.1.2 Exponential Ensemble Decay Model (EEDM) 15

5.2 Online and Non-Parametric Drift Detection Methods Based on Hoeffding's Bounds (Frías-Blanco et al., 2015)

5.2.1 Hoeffding's Inequality and its Application 15

5.2.2 Moving Average with Hoeffding's Bound 15

5.2.3 Weighted Moving Average with Hoeffding's Bound 16

5.3 Hoeffding Trees: Scalable Decision Trees for Streaming Data (Domingos and Hulten, 2000)

5.3.1 Adaptive Node Splitting 16

5.3.2 Handling Concept Drift 16

5.4 Adaptive Random Forests: Ensemble Methods for Concept Drift (Gomes et al., 2017)

5.4.1 Incremental Learning in Adaptive Random Forests 17

5.4.2 Handling Gradual Concept Drift 17

5.4.3 Integration with Drift Detection Techniques 17

6 Overview on the RiverML Framework

6.1 Architecture of RiverML

6.1.1 Core Components 18

6.1.2 Integration and Extensibility 18

6.2 Key Features of RiverML

6.2.1 Incremental Learning 19

6.2.2 Efficiency and Scalability 19

6.2.3 Real-time Processing 19

6.2.4 Ease of Use 19

6.3 Use Cases of RiverML

6.3.1 Real-time Recommendations 20
6.3.2 Predictive Maintenance 20
6.3.3 Healthcare Monitoring 20
6.3.4 Financial Market Analysis 20
6.3.5 IoT Data Processing 21
6.4 Datasets Provided by RiverML
6.4.1 Synthetic Datasets 21
6.4.2 Real-world Datasets 21
7 AdaBoostClassifier
7.1 Key Ideas of AdaBoostClassifier 22
8 Bagging
8.1 Key Ideas of BaggingClassifier 23
9 Drift Detection Techniques
9.1 ADWIN (Adaptive Windowing) 25
9.2 DDM (Drift Detection Method) 26
9.3 HDDM A 28
9.4 EDDM (Early Drift Detection Method) 29
10 Adaptive Random Forest 30
11 SRPClassifier 33
12 Fair-Weighted Dynamic Ensemble Based on Hybrid Drift Detection and Champion Selection
12.1 FDE-HDDCS 35
12.2 Champion Approach 37

13 Overview On The Datasets Used
13.1 CICIDS2017 Dataset 39
13.2 Elec2 Dataset 45
14 Comparative evaluation
14.1 Elec2 Results and Analysis 50
14.2 Cicids2017 Results and Analysis..... 57
15 Conclusion 63
References 64

Abstract

In this study we talk about concept drift, a phenomenon well known in real-world data where data distribution evolves over time and where a certain features combination that used to mean something in another timeline and with the new context those same features will result in something totally different or slightly different which will impact the accuracy of the classical machine learning models. That's why we propose different solutions in this study from adaptive random forest and streaming random patches that both use drift detection techniques like ADWIN and DDM to detect the new concepts and retrains the affected trees on the new data, to online boosting techniques like Adaboost which adapts to model drift indirectly by making weak learners learn from misclassified instances. Last but not least, we propose our new approach "FairWeighted Ensemble Based on Hybrid Drift Detection (FE-HDD)". This new approach combines different ARF's and SRPClassifiers that monitor differently using different detection techniques and each classifier has its own weight to the decision based on its performance on that specific data point. Also, the mix of using abrupt and gradual drift detectors would make the model aware and adaptive to all types of drifts, making it more robust. but the specific combination of detectors and classifiers that we used and gave us the best results at some point and under certain conditions probably won't be the best pick , that's why we developed the approach and made "FairWeighted Dynamic Ensemble Based on Hybrid Drift Detection and Champion Selection (FDE-HDDCS)" which is a heavy approach that explores all the different combinations of base learners group them based on their complexity and choosing one with the highest accuracy from each group as a challenger than we apply the champion selection to have an ultimate champion based on a certain criteria that we will set based on the scenario . the champion model and this approach will run independently the champion model works normally and the new approach will take some hours after taking the new data as a training data to update the ultimate champion or even not update it . This approach will always give us the best pick possible based on the situation and works perfectly in pre-saisons and mid-seasons updates .

Keywords: Concept Drift, Adaptive Random Forest, Streaming Random Patches,Drift Detectors, Adaboost,Bagging, FairWeighted Ensemble, Hybrid Drift Detection, Champion Selection,ultimate champion, data streams.

1 Introduction

In the realm of machine learning, the assumption that the data distribution remains constant over time is often violated in real-world scenarios, where the relationship between features and the target class can totally or slightly differ based on the context and the timeline [1]. Concept drift is the biggest challenge for the traditional classic models like decision trees and random forest, simply because they don't work against it in the field of evolving data streams where the data distribution is dynamic and always changing, making the historical data useless and using those static models would only lead to deteriorating the performance [2].

Understanding the idea of concept drift and developing approaches and techniques that can detect it and make more robust models that can mitigate its effects is a necessity in any real-world application and dynamic environments. Let's consider a predictive maintenance system in industrial settings [3]. The model trained to detect equipment failure is going to use historical sensors for data which may encounter concept drift in different shapes starting from machinery ages to undergo wear and tear. These drifts and changing patterns can lead to costly damages and safety issues if not detected and handled as soon as they happen [4].

Similarly, in financial markets [5], even though the stock market is based on the reputation and the ideas of companies, the models can't really rely on historical market data only if we are talking about investment decisions. The stock market is influenced by economic factors, geopolitical events, and the investor gut itself. All these are drifts that can lead to erroneous predictions and financial losses.

We can even see model drift in the field of healthcare [6]. The use of static predictive models in disease diagnosis that won't put into consideration the evolving patient demographics and changes in disease prevalence, for example, can delay the whole diagnosis and even cost a human being [7].

Addressing concept drift requires the development of adaptive learning algorithms capable of detecting and responding to changes in the underlying data distribution in real-time [8]. These types of algorithms usually use incremental learning and ensemble learning to continuously update the model and adapt to evolving data dynamics [9]. By monitoring data streams using a drift detector and adjusting model predictions dynamically, these algorithms will be immune against concept drift and the models will be robust against all types of changes [10]. Here in this article, we are going to see how we can handle concept drift. Section 2 we will explain concept drift with details from it's different types to the results of an untreated model drift to some common strategies that may work against it . Section 3 will explore the related works that inspired us to do this article.In section 4 we are going to have an overwiew of the RiverML framework which is one of the best frameworks if we are working with real world data . Section 5 will talk about the first indirect approach that we proposed which is AdaBoost to handle concept drift and just after it in section 6 we are going to present another interesting indirect approach that may work against concept drift which is Bagging. In Section 7, we are going to dive into drift detectors theory and all the math behind the different detectors. In Section 8, we will talk about the first direct approache to handle concept drift which is Adaptive Random Forest .In Section 9, we will talk about the second direct approach of the day which SRPClassifier . In Section 10, we are going to introduce our novel approach "FairWeighted Dynamic Ensemble based on Hybrid

Drift Detectors and Champion Selection " and explain the ideas of the approach .In section 11 we are going to have an overview on the 2 datasets that we used for the comparative evaluation which , both elec2 dataset and cicids2017 dataset which are benchmarks used to try techniques and algorithms that handles and adapts concept drift . In Section 12, we will analyze the results from the comparative evaluation made for the elec2 dataset and the cicids2017 dataset and analyze the different choices of our approach based on multiple scenarios and the plots that compares the choosen model with the other basic approaches and indirect approaches . Finally, Section 13 will conclude this article by summarizing our key takeaways and discussing potential future directions for research.

2 Concept Drift

2.1 Definition of Concept Drift

Concept drift , a problem well known in the field of evolving data streams and real world applications . It refers to the phenomenon in which relationship between the input features and the target variable will change over time , which will lead to a degradation in model performance . this could be a huge problematic to classic models that rely on historical data and assumes that the data distribution is static . That's why when drift occurs these models will no longer gives us accurate predicitions, and this could lead to some serious losses if not treated as soon as possible .

2.2 Historical Background

The term model drift or concept drift may seem new and a fresh topic to work on in the last few years but what's really interesting is that the first time they worked on it is in the early 1960s,initially emerging from fields such as adaptive control systems, time-series analysis, and statistical process control. We are going to see with details the historical context of concept drift and the developpements over the years to have a better view on this problematic .

2.2.1 Early Developments

The idea of adaptive systems had initially been considered back in the 1960s and 1970s in an attempt to achieve more efficient methods of controlling complex systems. trol theory. Ecologies of researchers including Simon, Herbert and Allen Newell came up with initial algorithms. that could enable the organisms to change their own environment and/or adapt to that environment. It was these early models that provided the benchmark for all subsequent attempts at public relations theory-building. learning about how system could possibly be developed within response to more anew data. Adaptive control the- ory that was specifically centered on controllers being developed in a fashion that allowed for the modulation of parameter values in real-time and in a manner that would enable the avoidance of instability. Almost all researchers in the CBR system agree that the design intent is to achieve the best possible performance in conditions of dynamic system perturbations. From this approach of learning, one of the valuable findings from this period was the refinement of the Least.

The delta-rule algorithm was proposed by Robert Hecht Nielsen in 1965, whereas the Least Mean Square (LMS) algorithm was proposed by B. Widrow and M.E. Hoff in 1960. The LMS algorithm was the first of a kind bi-parameter adaptive algorithm the goal of which was to reduce the mean square error between the predicted and actual outputs so they can adapt the model parameters in a constantly forward-moving process. This continuous adaptation is one of the main principles that are characteristic to the modern approaches to concept drift detection.

2.2.2 Development in the 1990s

Concept drift was first noted in the early 90s, and the term "concept drift" appeared quite recently as well. Key works by researchers like Widmer & Kubat in 1996 discussed the limitations and possibilities of learning from data streams where the complexity of the concept could change in the course of time. In addition to Aberela, Widmer and Kubat also proposed FLORA (FLexible Online cReamAtion), which is a family of algorithms capable of detecting and adapting to such changes, thereby providing the base for the introduction of contemporary techniques for addressing concept drift.

The FLORA algorithms were among the first to address the issue of concept drift explicitly. In robotics, it is important to recollect that the FLORA algorithms were at the initial stage of addressing this issue. Similarly, in the context of explosions, we reuse the idea that the FLORA algorithms were at the initial stage of addressing the issue of concept drift. In text mining, remember that the FLORA algorithms were at the initial stage of addressing the issue of concept drift explicitly. They used schemes for testing for drift and excluding models that have been stale for some time in the general auto Wiktionary community, as well as incorporating new data dynamically. These algorithms, to a good extent, underlined the fact that it is necessary to preserve the balance between stability and plasticity, making it possible to change them without sacrificing notions that are already established learned knowledge.

2.2.3 Advances in the 2000s

Thus, the literature development in the 2000s reached notable changes with the introduction of machine learning techniques. Different ensemble methods that were developed during that period include Bagging and Boosting as new algorithms conceived to identify cases of this phenomenon.

Hoeffding Trees and Adaptive Resonance Theory (ART) networks marked distinct progress. Researchers like Chaudhury and Mahmood (1992) and Gama et al. (2004) helped in attaining a better comprehension of the field. They presented new approaches to the identification and mitigation of different kinds of concept drift.

Aggregation techniques came into focus particularly at this time. Bagging (Bootstrapping and Aggregating) and Boosting are methods for improving performance using multiple models for the base system, enhancing prediction accuracy and robustness. These methods played a role in regulating concept drift by keeping a number of models active, allowing one to easily shift its focus by capturing new patterns in the data.

There is another type of tree called Hoeffding Trees, also known as VFDT (Very Fast

Decision Trees), mentioned by Domingos and Hulten. These trees leverage the application of the Hoeffding bound to derive probabilistically reasonable decisions concerning splits of streaming data. Other advances were made in 2000 when Birger Steen, Claus Vesterager, and Marea Nordahl Hulten pioneered the use of these trees.

2.2.4 Recent Developments

However, since the late 2000s or early 2010s and especially in the context of the Big Data perspective, the field is still emerging. technologies, and a result of the advancements made in the field of deep learning. There are much more elaborated models presented by the researchers these days. instances that practice some of the features of online learning approaches and presence of real time data analysis. The focus has Also, knowledge management paradigms and practices have also moved to focus on engineering systems for sustained learning and learning in-real-time. resilience, which is crucial for matters related to finance along with machine learning and particularly healthcare as well as AI driven systems. Recent advancement has been the creation of tools like River for development of linguistic applications like a language. Its components include numeric variable selection used for feature selection and regularisation, numeric variable transformation for data preprocessing, SMOTE for imbalance proportion and OULI for outlier detection. These Frameworks help to put into practice working models of adaptive nature that in a position to process data in real time. time and in a continuous manner readjust based on the shift in concept timing. Furthermore, the eruption of deep learning ushered in new techniques and questions in handling concept drift. The Author further elaborates how deep neural networks, despite having the ability to accurately learn complex patterns, can be more vulnerable Submitters admit that such regulations may drift due to the very fact of being complex. Yusop and Haslinda asserted that researchers have been examining various approaches and mechanisms, including continual. learning and transfer learning to enhance the performance of deep models to dynamic data set. without catastrophic forgetting.

2.3 Types of Concept Drift

Model drift can be in different shapes and types and manifest in various forms . Understanding which form we are dealing with is a must for designing appropriate strategies from windowing to error based monitoring and using specific drift detectors based on the type .

2.3.1 Sudden Drift

Or Abrupt Drift , it occurs when there's a sudden changes out of no where in the data distribution . it is usually associated with significant events or new rules and policies being added that made the market unstable . This type of drifts is so dangerous because the classic models that is not drift aware will experience a sharp decline in performance .

Sudden drift can be detected using change detection methods such as CUSUM (Cumulative Sum Control Chart) and Page-Hinkley test. These methods monitor statistical properties of the data and trigger alerts when significant changes are detected. Also Error Based Detectors are a good pick such as DDM(Drift Detection Method) or EDDM(Early

Drift Detection Method) . These methods monitors the performance in a continuous way and if a significant decrease occured the model adapts by retraining on the most recent data for exemple .

2.3.2 Incremental Drift

Incremental Drift is the slowest variation from all the different types of model drift. These slow changes may go unnoticed if not using the right detectors , by using abrupt drift detectors or some statistical tests , this could lead to not monitoring the performance of the model as it should be and could lead to serious damages in the long run . From the different exemples we can mention consumer preferences and climate patterns , these 2 exemples won't change statistically over night but slowly and surely there's some changes that will happen and being self aware about those unnoticed small changes is a must .

Incremental drift can be managed using adaptive learning algorithms that update model parameters incrementally. Techniques such as online learning and incremental learning allow models to learn from new data continuously, ensuring they remain relevant as the data distribution evolves.

2.3.3 Gradual Drift

Gradual drift is a combination of sudden and incremental drift. It involves changes that start slowly but accelerate over time. For instance, the shift from traditional to digital media consumption illustrates gradual drift, where the change begins slowly but gains momentum. Detecting and adapting to gradual drift requires models capable of identifying both slow and accelerating changes.

Gradual drift is a mixture between sudden drift and incremental drift , but lean more towards incremental drift but not as slow and more consistent over time . For instance , the shift from watching and consuming traditional media from tv and newspapers to online media and digital content , the change didn't happen over night neither took so long , it just happened slowly and then gained momentum . that's why Detecting and adapting to gradual drift requires models capable of identifying both slow and accelerating changes, using a robust error based detector that handles both slow and fast scenarios is a must such as HDDM-A , also windowing techniques like ADWIN works perfectly in gradual circumstances.

2.3.4 Recurrent Drift

Recurrent drift occurs when the data distribution changes periodically, returning to it's original states after some time. Seasonal trends, for exemple sales spikes of pastries and cakes during ramadan , is a classic example of recurrent drift. Models must be designed to recognize these cycles and recognize the patterns to maintain the performance across different periods.This can be done by using seasonal decomposition and time series analysis techniques which means what's different than the other types that we have seen so far is that we need to have a detailed idea on when the trends will occur and what will usually happen during each period of time and condition the model to adapt to each situation .

2.3.5 Blip Drift

Blip drift happens in a very short period and vanishes due to short lived trends or the boost after a good marketing strategy . the models used must be aware and resilient to this type of drift or it could lead to overfitting to these temporary changes in the data distributions .

Blip drift can be handled by using regularization techniques such as L2 regularization or using ensemble methods that are robust to overfitting .

2.4 Effects of Untreated Concept Drift

Like we already said , Failing to address concept drift can lead to severe consequences on the performance and reliability of machine learning models. We are going to see the effects of untreated model drift :

2.4.1 Decreased Accuracy

The most immediate consequence of untreated concept drift is a significant drop in model accuracy. As the model always assumes that the data distribution is static , but in case of a drift the relationship between input features and the target will change over time like we said earlier , this misunderstanding makes the model clueless and his predictions become increasignly innacurate making it unreliable in almost any situation .

2.4.2 Model Misinterpretation

The changing in underlying data distribution over time will cause the model to provide misleading insights , which will misguide us into take bad strategies and decisions .

This can be specially a huge problem in critical scenarios from medical diagnosis to financial investements. For example, in the field of healthcare, inaccurate predictions along with the misleading clues will generate incorrect diagnoses and then incorrect treatment which will cost the patient's health and even life by making it worse . Also in finance, it will result in some huge losses by putting money on bad investements while thinking that they are good investements.

2.4.3 Increased Error Rates

When the accuracy goes down , of course the error rates will go up , which will turn into a huge production of false positives and false negatives , in a scenario of cancer diagnoses for exemple , if the result is false negative because of a drift , this means that the patient and the doctors will think that there's no disease leading to untreated cancer for much longer time and checking again will be so late.

2.4.4 Customer Dissatisfaction

In customer-facing applications that rely on recommendation systems like youtube or netflix , concept drift can lead to bias suggestions leading to user frustration and even switching to other rival competitors in the market .

That's why consistent feedbacks and frequent updates is a must in this case or the share of the company will be shared between the rivals in the same market .

2.5 Strategies for Handling Concept Drift

Addressing concept drift requires a combination of techniques and strategies. Some of the key approaches include:

2.5.1 Data Monitoring and Analysis

Data monitoring is one of the most effective ways to detect concept drift by either using statistical techniques from control charts to hypothesis testing which can help in identifying changes in data distribution, to tracking key metrics by using specific drift detectors based on statistical tests like Kolmogorov-Smirnov test and the Kullback-Leibler divergence , both can be used to compare distributions and detect significant changes in the data streams. Using these along with plotting and analyzing the results is one of the best ways to know how to treat the different drifts and we will have a clear idea on the trends and different patterns in the data stream .

2.5.2 Model Updating and Retraining

updating the models with new data whenever we had a chance is crucial for maintaining performance over time and won't be dependent on useless historical data that will actually harm the performance.

Techniques such us online learning and incremental learning enable models to adapt to new data consistenly without retraining from scratch , we can mention adaptive random forest (ARF) as an incremental learning technique that only updates the affected trees and replace them with new trees trained on the most fresh data , also an interesting online algorithm is Stochastic Gradient Descent (SGD). Both approaches are commonly used and work perfectly .

Model updating and retraining can be performed periodically by using batches but the best way is to make it triggered by detected drift. mixing it with error based drift detection techniques such as DDM for exmple and if a drift is detected the affected parts are retrained based on the newest data is the best way to do it , classifiers like arf and srpclassifier which will see later in section 8 and 9 uses the drift detection with the retraining and updating .

2.5.3 Ensemble Methods

Ensemble methods such as Bagging and Adaptive boosting (AdaBoost) can work in some situations against concept drift specially if the problem isn't that complex for exemple in temporary drifts . it adapts in a indirect way by either focusing on the misclassified instances using Adaboost or using random sampling in bagging making it more robust to changes . we are going to see both approaches later in section 5 and 6.

but it won't probably work in more complex data where there's a mix of gradual , reccurent and abrupt drifts because the ensemble methods aren't drift aware .

3 Related Works

Numerous research efforts have explored various approaches to handle concept drift. This section provides an overview of four prominent approaches:

3.1 Dynamic Classifier Selection for Drift Detection (Pinho and Santos, 2012)

In their work, Pinho and Santos explored the use of dynamic classifier selection combined with drift detection methods to adaptively manage concept drift. The approach involves maintaining an ensemble of classifiers and dynamically selecting the most suitable one based on the current concept and performance metrics. This adaptive strategy aims to optimize prediction accuracy while efficiently managing computational resources [11].

3.1.1 Dynamic Classifier Selection (DCS)

Dynamic Classifier Selection focuses on dynamically choosing the most appropriate classifier from an ensemble for each new instance based on its performance history and the current concept. This approach helps in maintaining high accuracy by leveraging the strengths of different classifiers under varying conditions .

3.1.2 Exponential Ensemble Decay Model (EEDM)

EEDM is a mechanism used to adapt the ensemble of classifiers over time. It prioritizes classifiers that exhibit superior performance on recent data while phasing out those with declining accuracy. By continuously updating the ensemble with new classifiers trained on the latest data points, EEDM ensures that the model remains robust against concept drift.

3.2 Online and Non-Parametric Drift Detection Methods Based on Hoeffding's Bounds (Frías-Blanco et al., 2015)

Frías-Blanco et al. conducted a comparative study of drift detection methods that utilize Hoeffding's bounds, focusing on their applicability in diverse data scenarios without assuming specific data distributions. Their work highlights the effectiveness of non-parametric approaches in detecting and responding to concept drift in real-time streaming data environments [12].

3.2.1 Hoeffding's Inequality and its Application

Hoeffding's Inequality provides a probabilistic bound on the difference between sample mean and population mean, which is instrumental in drift detection. Various drift detection algorithms based on Hoeffding's bounds leverage this inequality to monitor statistical changes in data distributions over time.

3.2.2 Moving Average with Hoeffding's Bound

The Moving Average with Hoeffding's Bound method tracks the accuracy of predictions over a sliding window of data. It compares the moving average of accuracy with an upper bound derived from Hoeffding's bounds. If the difference between the observed accuracy and the bound exceeds a predefined threshold, it signals potential concept drift, prompting adaptive measures in the model.

3.2.3 Weighted Moving Average with Hoeffding's Bound

This approach enhances the Moving Average method by assigning higher weights to recent data points, reflecting the evolving trends in the data distribution. By emphasizing recent observations, the Weighted Moving Average method improves sensitivity to gradual changes in the data, making it particularly effective in scenarios where concept drift manifests gradually over time.

3.3 Hoeffding Trees: Scalable Decision Trees for Streaming Data (Domingos and Hulten, 2000)

Hoeffding Trees, introduced by Domingos and Hulten, are decision tree classifiers designed for handling streaming data with limited memory and computational resources. Unlike traditional decision trees that require retraining on entire datasets, Hoeffding Trees dynamically grow with incoming data, making split decisions based on statistical confidence (Hoeffding's bound) rather than exhaustively evaluating all data points [13].

3.3.1 Adaptive Node Splitting

Hoeffding Trees use Hoeffding's inequality to determine whether to split a node based on a statistically significant improvement in classification accuracy. This adaptive splitting mechanism allows the tree structure to evolve incrementally as new data arrives, ensuring efficient and accurate predictions without the need for retraining from scratch.

3.3.2 Handling Concept Drift

Hoeffding Trees are particularly suited for environments prone to concept drift, where the underlying data distribution may change over time. By leveraging adaptive node splitting and continuous learning, Hoeffding Trees can adapt to new concepts without discarding previously learned knowledge, thereby maintaining model accuracy and reliability in dynamic data streams.

3.4 Adaptive Random Forests: Ensemble Methods for Concept Drift (Gomes et al., 2017)

Gomes et al. proposed Adaptive Random Forests as an ensemble learning technique tailored for handling concept drift in streaming data scenarios. This approach extends traditional

Random Forests by incorporating mechanisms to dynamically update ensemble members and adapt to evolving data distributions [14].

3.4.1 Incremental Learning in Adaptive Random Forests

Adaptive Random Forests facilitate incremental learning by allowing new decision trees to be added to the ensemble as new data arrives. This incremental approach ensures that the model can adapt to changes in data without requiring retraining from scratch, thereby optimizing computational efficiency and response time.

3.4.2 Handling Gradual Concept Drift

One of the strengths of Adaptive Random Forests lies in their ability to detect and respond to gradual concept drift. By monitoring changes in prediction accuracy and data distribution over time, Adaptive Random Forests adjust the weights and importance of ensemble members, thereby maintaining model performance in dynamic environments.

3.4.3 Integration with Drift Detection Techniques

Adaptive Random Forests integrate with various drift detection methods to trigger adaptive responses when concept drift is detected. This integration enhances the model's ability to maintain accuracy and reliability by updating ensemble members and adjusting decision boundaries in response to changing data conditions.

4 Overviw on the RiverML Framework

RiverML is a novel open-source machine learning environment for massive online learning and concept drift handling. They help models to learn from data received in a stream and update them in real-time in case there is a change in distribution. RiverML is likely to be especially important when data comes in a streaming mode, and the emphasis is made on timely decision-making, for example, stock or currency futures, recommendation systems, and intrusion detection in sensor streams. RiverML can work across different types of applications given they deal with data that changes often and is therefore very dynamic. Some additional features include enhanced drift detection methods and incremental learning for updating the predictive model periodically.

4.1 Architecture of RiverML

RiverML is designed as a plug-in architecture where different components for online learning of a model can be represented in the system, including data preprocessing, model training, and concept drift detection. This kind of architecture has the advantage of flexibility, optimizes resource usage, and can easily interface with other systems and application.

4.1.1 Core Components

The core components of RiverML include:

1. **Data Streams:**

   Streams in RiverML are ideal for continuously feeding a system with data. Third-party sources offer APIs and abstractions for consuming and manipulating incoming data in a streaming mode. The types of data streams that are supported by RiverML include files, sensors, and network feeds of a river. This means that the framework is flexible in that it can manage different types of streams for example, real-time, batched and synthetic data streams thus, compatibility with a variety of data sources.

2. **Preprocessing:**

   In RiverML, preprocessing is interpreting and preparing raw data in a form which can be used in the model. This includes normalizations, feature extractions, intended absence of missing values and the likes. Given that preprocessing is typically performed prior to feature extraction, RiverML also provides preprocessing tools in this order; and these can be used incrementally on data streams. For instance, incremental normalization techniques adapt mean and variance as the data stream enlarges, which again is on the same scale as the stream.

3. **Model Training:**

   Learning in RiverML offers the solutions built for online training of the model. This means that models can be modified in bits through new data to ensure that it evolves with display properties. RiverML offers a range of statistical models for machine learning, to be more specific, it supports linear models, and ensembles, which are optimized for online learning. Some of the umarism algorithms include SGD (Stochastic Gradient Descent), Hoeffding Trees, Adaptive Random Forests, and SRPClassifier, which are suitable for work in online learning spaces.

4. **Concept Drift Detection:**

   Concept drift detection is a critical component of RiverML. The framework includes several algorithms to detect different types of concept drift, such as sudden, gradual, and recurring drifts. These algorithms monitor changes in the data distribution and trigger model updates when drift is detected. Popular drift detection methods integrated into RiverML include ADaptive WINdowing (ADWIN), Drift Detection Method (DDM), Early Drift Detection Method (EDDM), Page-Hinkley Test, and more advanced techniques which we will explore in detail in Section 8.

4.1.2 Integration and Extensibility

Overall, RiverML aims to be as integrated and as modular as possible. They have provided APIs for links with other services and products. multi-disciplinary in the field, ranging from different machine learning frameworks to different systems. There is also an opportunity for users to expand the model by inputting more data into RiverML. tasks, data sources, and

data preprocessing techniques, as well as different categories of machine learning algorithms. This flexibleness makes RiverML suitable for adaptation to different applications and as many research needs as possible could be addressed by the construction of such networks.

4.2 Key Features of RiverML

There also are some distinctive features in the RiverML which will be suitable for handling the concept drift in the online learning context. All these features are to desirable qualities that can increase efficiency, scalability and flexibility in a artefacts.

4.2.1 Incremental Learning

Learning from a few examples at a time is the key to the RiverML. During each data presentation in a learning process, models can be updated every time with new parameters of data distribution. It is far less sensitive than the traditional approach, which reduces the likelihood of an investigation becoming a highly publicized event. Reducing the model's parameters to a few percent of their original size can save computational power and time for retraining from scratch. Incremental learning is particularly disadvantageous in setting with large amounts of data changing frequently as is the case in IoT scenery. and financial trading.

4.2.2 Efficiency and Scalability

Among the main concepts in RiverML it is worth dwelling on the fact that efficiency and scalability are the priorities. It is strategically well aligned for Selangor to enable it to diversify its economy by unlocking the potentials of its various sectors while providing incentives to investors in various ways.

all that by using minimal RAM requirements and quick computation, which makes it easy for large-scale streaming application.Also optimized data structures and online algorithms which proved to be so good against model drift and the incremental learning and updating

4.2.3 Real-time Processing

Strengths speak about the ability to work in real-time,This is important for real time environments and scenarios where we need to take actions almost in a split of a second.

4.2.4 Ease of Use

RiverML has been organized with simplicity in mind and one of its main features is that it possesses a simple API plus extensive documentation.

umentation. It offers a number of utilities and examples to assist users in executing recipes and working with Hexi vectors rapidly.

efficiently. To this end, simplicity of structure is the core of the framework without compromising on the components.

This is a major advantage when it comes to practicing what it preaches because it is written in a way that can be easily understood by beginners and at the same time helpful to the advanced learners.

4.3 Use Cases of RiverML

The concept drift which is a major problem in data stream mining, and RiverML has been shown to be effective in dealing with this problem and can be extended to a variety of real application scenarios. Here are some notable use cases of RiverML:

4.3.1 Real-time Recommendations

orgs involved in e-commerce or content delivery even social media can tap into RiverML to create real-time recommendations. RiverML can reflect the updated user's preferences and behaviours by adapting to the changes. To keep the models accurate and up to date, apps use the latest fashion trends to provide users with suitable recommendations. For example, through RiverML, streaming services can often diagnose the viewers' trend and suggest other suitable materials. built from the most recent user experiences of popularity.

4.3.2 Predictive Maintenance

in the industrial environment, data from sensors placed in the machines and equipment may be employed in making predictive maintenance. The data generated by these signals can also be processed by RiverML in real-time, especially warning signs of wear and potential failures. This makes it possible to carry out timely maintenance actions that may hinder added costs on equipment.For example, manufacturing plants can use RiverML to monitor machine performance and budget limitations to decide which machines to maintenance,minimising recurrent failures before major breakdowns happen.

4.3.3 Healthcare Monitoring

Wireless biosensors, for example, in the monitoring of patient status and/or health parameters, produce steady feeds of data. Consequently, RiverML can interpret this information in real-time, thus identifying changes to the patient conditions. And involving themselves in the delivery of early signs of possible health complication. For example, using RiverML, it can monitor measure parameters such as body temperatures, detecting changes to important physiological activity and alerting health care providers to take action.

4.3.4 Financial Market Analysis

Stock markets have a lot of volatility, and situations transform considerably. RiverML can proquately assessing the effectiveness of these financial data streams, refine existing models in relation to new trends prevailing in the market as well as enhance the accuracy of the results provided by these models.erude ((high risk)) successfully to forecast to stock prices, trading techniques and risk management. Traders and financial analysts may then leverage RiverML to develop trading strategies that update themselves to market shifts in real-time.

4.3.5 IoT Data Processing

One of the main characteristics of Internet of Things (IoT) devices is that the produce a lot of streaming data. RiverML can This data can be processed and probably, analyzed in earnest, in real-time to offer insights and to facilitate actions based. on current conditions. Some of the utilizes of the system are in smart homes, monitoring the environmental conditions, and the likes. industrial automation.

4.4 Datasets Provided by RiverML

RiverML includes a variety of datasets to help users test and validate their models. These datasets cover different domains and types of data streams:

4.4.1 Synthetic Datasets

Synthetic datasets are generated to simulate different types of concept drift. These datasets are useful for testing the performance of drift detection algorithms and model adaptability. Examples include the SEA dataset, the STAGGER dataset, and the Rotating Hyperplane dataset.

4.4.2 Real-world Datasets

RiverML provides access to real-world datasets from various domains. These include:

1. **Financial Data:**

   Datasets containing stock prices, trading volumes, and other financial indicators.

2. **Healthcare Data:**

   Datasets with patient health records, vital signs, and medical history.

3. **E-commerce Data:**

   Datasets with user interactions, purchase history, and product information.

4. **Sensor Data:**

   Datasets from IoT devices, including environmental sensors and industrial equipment.

   These datasets enable users to benchmark their models and assess their performance in real-world scenarios.

5 AdaBoostClassifier

AdaBoostClassifier is an ensemble method that trains weak learners sequentially in this case hoeffding trees , the sequential learning will make each weak learner focusing on the misclassified instances by the previous weak learner and so on which will improve the accuracy and make AdaBoost an indirect way to adapt against concept drift , we are assuming that the weak learners will try to adapt to the new concept if a new drift came .

5.1 Key Ideas of AdaBoostClassifier

1. **Initialization**:

- we start by initializing the ensemble (base-model , how many weak learners...) - in this stage all the training instances have an equal weight

2. **Training Weak Learners**:

weak learners will be trained in a sequential way , that means each weak learner will focus more on the misclassified instances from before and so on .

3. **Updating Weights**:

the key here is after each training , the weights of the instances will be updated , giving higher weights to the misclassified instances so the future weak learners will focus on them .

4. **Making Predictions**: after training all the weak learners, the prediction will be the combination of all their outputs via a voting scheme

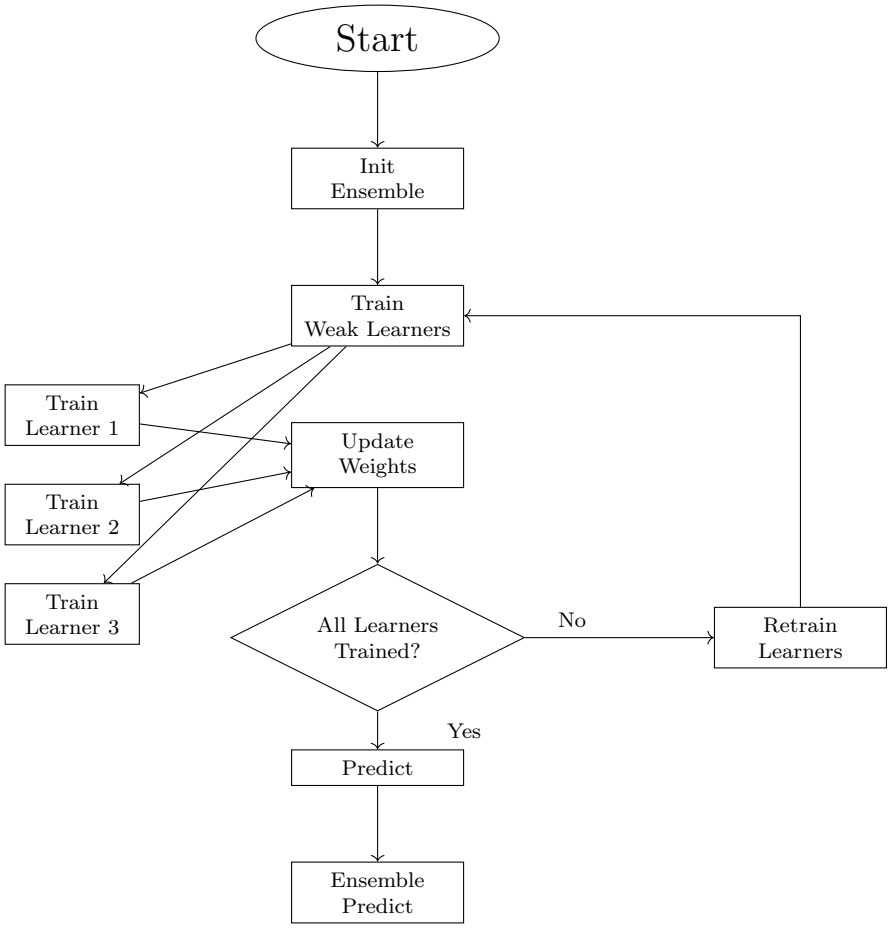


Figure 1: Detailed Steps in AdaBoostClassifier

6 Bagging

Bagging or Bootstrap Aggregating , is an ensemble machine learning technique that combines the predictions of different weak learners trianed on random subsets which make it more robust to overfitting and improves the performance .

We suppose that Bagging will try to adapt against model drift in an indirect way by using bootstrap sampling which makes the data distribution and the idea of trends not as important when we train different weak learners on different samples .

6.1 Key Ideas of BaggingClassifier

1. **Initialization**: Here we initialize the ensemble with a specified number of weak learners 10 for exemple .

2. **Bootstrap Sampling**: Here we will generate bootstrap samples for each weak learner . Those bootstap samples are random samples with replacement from the original dataset .

3. **Training**: Here we will Train each weak learner on its own bootstrap samples .

4. **Making Predictions**:Here Each weak learner makes a prediction on the input data.

5. **Combining Predictions**:

Here we will aggregate and combine the predictions of all weak learners using a voting mechanism

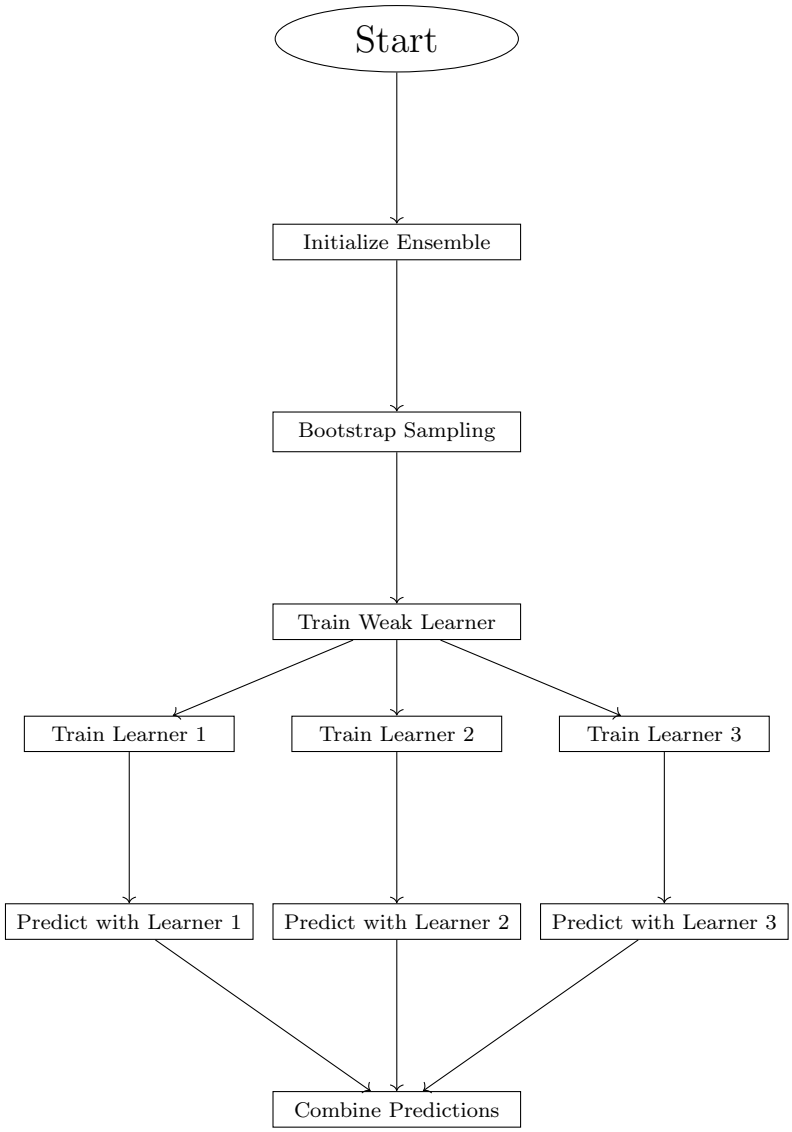


Figure 2: Detailed Steps in BaggingClassifier

7 Drift Detection Techniques

7.1 ADWIN (Adaptive Windowing)

ADWIN (adaptive windowing) is a drift detection method that continuously adjusts and adapts the window size based on changes in the data distribution. It uses Hoeffding's inequality to detect significant changes in the averages of two sub-windows within the current window.

Mathematical Basis

ADWIN relies on Hoeffding's inequality to determine if the averages of two sub-windows within the current window differ significantly. Hoeffding's inequality provides an upper bound on the probability that the sum of random variables deviates from its expected value.

Variables:

- W (pronounced "W"): The current window of size n (pronounced "n").
- W_1 (pronounced "W one") and W_2 (pronounced "W two"): The two sub-windows of sizes n_1 (pronounced "n one") and n_2 (pronounced "n two"), respectively, where $n = n_1 + n_2$.
- $\hat{\mu}_1$ (pronounced "mu one hat") and $\hat{\mu}_2$ (pronounced "mu two hat"): The means of the sub-windows W_1 and W_2.
- $\hat{\mu}$ (pronounced "mu hat"): The overall mean of the window W.
- ϵ (pronounced "epsilon"): The confidence interval for the difference in means, calculated using Hoeffding's inequality.
- δ (pronounced "delta"): The confidence parameter, controlling the probability bounds.

Explanation:

1. **Overall Mean Calculation**:

- The mean $\hat{\mu}$ of the entire window W is calculated as the weighted average of the means $\hat{\mu}_1$ and $\hat{\mu}_2$ of the sub-windows W_1 and W_2.
- Formula:

$$\hat{\mu} = \frac{n_1\hat{\mu}_1 + n_2\hat{\mu}_2}{n}$$

- **Purpose**: This ensures that the mean of the entire window reflects the contribution of both sub-windows.

2. **Difference in Means**:

- The absolute difference between the means of the two sub-windows is computed:

$$|\hat{\mu}_1 - \hat{\mu}_2|$$

- **Purpose**: This measures the discrepancy between the mean error rates of the two sub-windows, indicating potential drift.

3. **Confidence Interval Calculation**:

- The confidence interval ϵ is determined by the confidence parameter δ, the total number of instances n, and the sizes of the two sub-windows n_1 and n_2.

- Formula:

$$\epsilon = \sqrt{\frac{1}{2} \ln \left(\frac{4n}{\delta} \right) \left(\frac{1}{n_1} + \frac{1}{n_2} \right)}$$

- **Purpose**: Hoeffding's inequality is used to calculate the probability bounds for the difference in means, indicating whether a significant change has occurred.

4. **Change Detection**:

- If the difference between the means

$$|\hat{\mu}_1 - \hat{\mu}_2|$$

  exceeds ϵ, a change is detected.

- **Purpose**: This mechanism ensures that the method can detect significant changes in the data distribution, prompting an adaptation in the window size by dropping the older sub-window W_1.

By continuously monitoring and adjusting the window size based on these calculations, ADWIN effectively detects and responds to changes in the data stream, maintaining the statistical properties necessary for accurate drift detection.

7.2 DDM (Drift Detection Method)

DDM is based on monitoring the error rate of a learning algorithm and using the PAC (Probably Approximately Correct) learning model to detect changes. DDM raises an alarm when the error rate increases significantly, indicating potential drift.

Mathematical Basis

DDM monitors the error rate and the standard deviation, detecting drift when significant changes occur.

Variables:

- p_t (pronounced "p t"): The probability of an error at time t.
- $\hat{p_t}$ (pronounced "p t hat"): The observed error rate at time t.
- σ_t (pronounced "sigma t"): The standard deviation of the error rate at time t.
- $\hat{p_{\text{min}}}$ (pronounced "p min hat"): The minimum observed error rate.
- σ_{min} (pronounced "sigma min"): The standard deviation of the minimum observed error rate.

Explanation:

1. **Standard Deviation Calculation**:

- The standard deviation σ_t estimates the variability in the observed error rates at time t.
- Formula:

$$\sigma_t = \sqrt{\frac{\hat{p_t}(1-\hat{p_t})}{t}}$$

- **Purpose**: This helps in understanding the spread or variability in the error rates, which is essential for detecting significant deviations.

2. **Drift Detection**:

- DDM raises an alarm if the observed error rate $\hat{p_t}$ plus its standard deviation σ_t exceeds the minimum observed error rate $\hat{p_{\text{min}}}$ plus twice its standard deviation σ_{min}.
- Condition:

$$\hat{p_t} + \sigma_t > \hat{p_{\text{min}}} + 2\sigma_{\text{min}}$$

- **Purpose**: This indicates a significant increase in the error rate, suggesting potential drift.

3. **Warning Zone**:

- DDM enters a warning zone when the observed error rate $\hat{p_t}$ plus its standard deviation σ_t exceeds the minimum observed error rate $\hat{p_{\text{min}}}$ plus its standard deviation σ_{min}.
- Condition:

$$\hat{p_t} + \sigma_t > \hat{p_{\text{min}}} + \sigma_{\text{min}}$$

- **Purpose**: This provides an early indication of potential drift, allowing for preemptive measures before a full alarm is raised.

By monitoring the error rate and its standard deviation, DDM effectively detects significant changes in the data distribution, ensuring timely responses to drift.

7.3 HDDM_A (Hoeffding's Drift Detection Method - Adaptive)

HDDM_A combines ideas from ADWIN and DDM. It uses Hoeffding's inequality to detect significant changes in the distribution of errors. HDDM_A adapts the detection threshold based on the observed error rates.

Mathematical Basis

HDDM_A maintains a window of error rates and splits it into two sub-windows, calculating the mean error rate for each.

Variables:

- $\hat{\mu_1}$ (pronounced "mu one hat") and $\hat{\mu_2}$ (pronounced "mu two hat"): The means of the two sub-windows.
- ϵ (pronounced "epsilon"): The confidence interval for the difference in means, calculated using Hoeffding's inequality.
- δ (pronounced "delta"): The confidence parameter, controlling the probability bounds.

Explanation:

1. **Means Calculation**:

- The means $\hat{\mu_1}$ and $\hat{\mu_2}$ represent the average error rates of the two sub-windows.
- Formula:

$$\hat{\mu_1} = \frac{1}{n_1} \sum_{i=1}^{n_1} e_i, \quad \hat{\mu_2} = \frac{1}{n_2} \sum_{i=n_1+1}^{n} e_i$$

- **Purpose**: These means help in detecting changes in the error rate over time.

2. **Difference in Means**:

- The absolute difference between the means of the two sub-windows is computed:

$$|\hat{\mu_1} - \hat{\mu_2}|$$

- **Purpose**: This measures the discrepancy between the error rates of the two sub-windows, indicating potential drift.

3. **Confidence Interval Calculation**:

- The confidence interval ϵ is determined by the confidence parameter δ, the total number of instances n, and the sizes of the two sub-windows n_1 and n_2.

- Formula:

$$\epsilon = \sqrt{\frac{1}{2} \ln \left(\frac{4n}{\delta} \right) \left(\frac{1}{n_1} + \frac{1}{n_2} \right)}$$

- **Purpose**: Hoeffding's inequality is used to calculate the probability bounds for the difference in means, indicating whether a significant change has occurred.

4. **Change Detection**:

- If the difference between the means

$$|\hat{\mu_1} - \hat{\mu_2}|$$

  exceeds ϵ, a change is detected.

- **Purpose**: This mechanism ensures that the method can detect significant changes in the data distribution, prompting an adaptation in the detection threshold.

By combining the ideas from ADWIN and DDM, HDDM_A provides a robust method for detecting drift in data streams.

7.4 EDDM (Early Drift Detection Method)

EDDM is designed to improve DDM by focusing on the distance between errors rather than just the error rate. It is particularly sensitive to gradual drift.

Mathematical Basis

EDDM monitors the average distance between errors and the standard deviation of these distances.

Variables:

- d_i (pronounced "d i"): The distance between the i-th error and the $(i-1)$-th error.
- $\overline{d}$ (pronounced "d bar"): The average distance between errors.
- σ_d (pronounced "sigma d"): The standard deviation of the distances between errors.
- $\overline{d_{\max}}$ (pronounced "d max bar"): The maximum observed average distance between errors.
- $\sigma_{\max}$ (pronounced "sigma max"): The standard deviation of the maximum observed distances.

Explanation:

1. **Average Distance Calculation**:

- The average distance $\overline{d}$ is calculated from the distances d_i between consecutive errors.
- Formula:

$$\overline{d} = \frac{1}{m} \sum_{i=1}^{m} d_i$$

- **Purpose**: This helps in understanding the typical interval between errors, which is crucial for detecting gradual drift.

2. **Standard Deviation Calculation**:

- The standard deviation σ_d measures the variability in the distances between errors.
- Formula:

$$\sigma_d = \sqrt{\frac{1}{m} \sum_{i=1}^{m} (d_i - \overline{d})^2}$$

- **Purpose**: This provides insights into the consistency of the error intervals.

3. **Drift Detection**:

- EDDM raises an alarm if the average distance $\overline{d}$ plus two standard deviations σ_d exceeds the maximum observed average distance $\overline{d_{\max}}$ plus two standard deviations $\sigma_{\max}$.
- Condition:

$$\overline{d} + 2\sigma_d > \overline{d_{\max}} + 2\sigma_{\max}$$

- **Purpose**: This indicates a significant change in the error intervals, suggesting potential drift.

By monitoring the distances between errors and their variability, EDDM provides a sensitive method for detecting gradual drift in data streams.

8 Adaptive Random Forest

Adaptive Random Forest is a variation of the random forest classifier , mostly used in the fields of evolving data streams and real-world applications due to its adaptability to model drift and incremental learning.

Key Steps in Adaptive Random Forest:

1. **Initialize the Forest**

- We start by creating the ensemble of decision trees that will make our forest using the n-models parameter and each tree will use a random subset of features for more diversity

2. **Configure Drift Detection**

-ARF have an internal drift detector (ADWIN as default) to monitor inside the forest and if a drift is detected the affected trees will either be resseted or replaced

3. **Incremental Learning**

-With new data coming , ARF thanks to incremental learning mechanism updates each tree in the ensemble and retrain them using the new data

- We need to take into consideration the params "grace period" which is the minimum number of data before considering a split and "lambda value" which is the number of new trees that we can often add to the forest

4. **Decision Tree Splitting**

- the split is decided by the params split-criterion (information gain as default) and delta which is the confidence level for splitting , the smaller the delta the higher confidence to justify the split .

5. **Tree Replacement** - if concept drift is detected while monitoring the data distribution , ARF will either reset or replace the affected trees.

6. **Prediction**

- Most of the time a majority vote is used inside the forest to make predictions

Overall, ARF is a good initial approach against concept drift and by using ensemble learning and a drift detector gradual or abrupt based on the situation to monitor all the time and adapt the model in case of a detected drift by replacing the affected trees with new trees trained on the most recent data , ARF makes sure to always be self aware against model drift in data streams and adaptive to it .

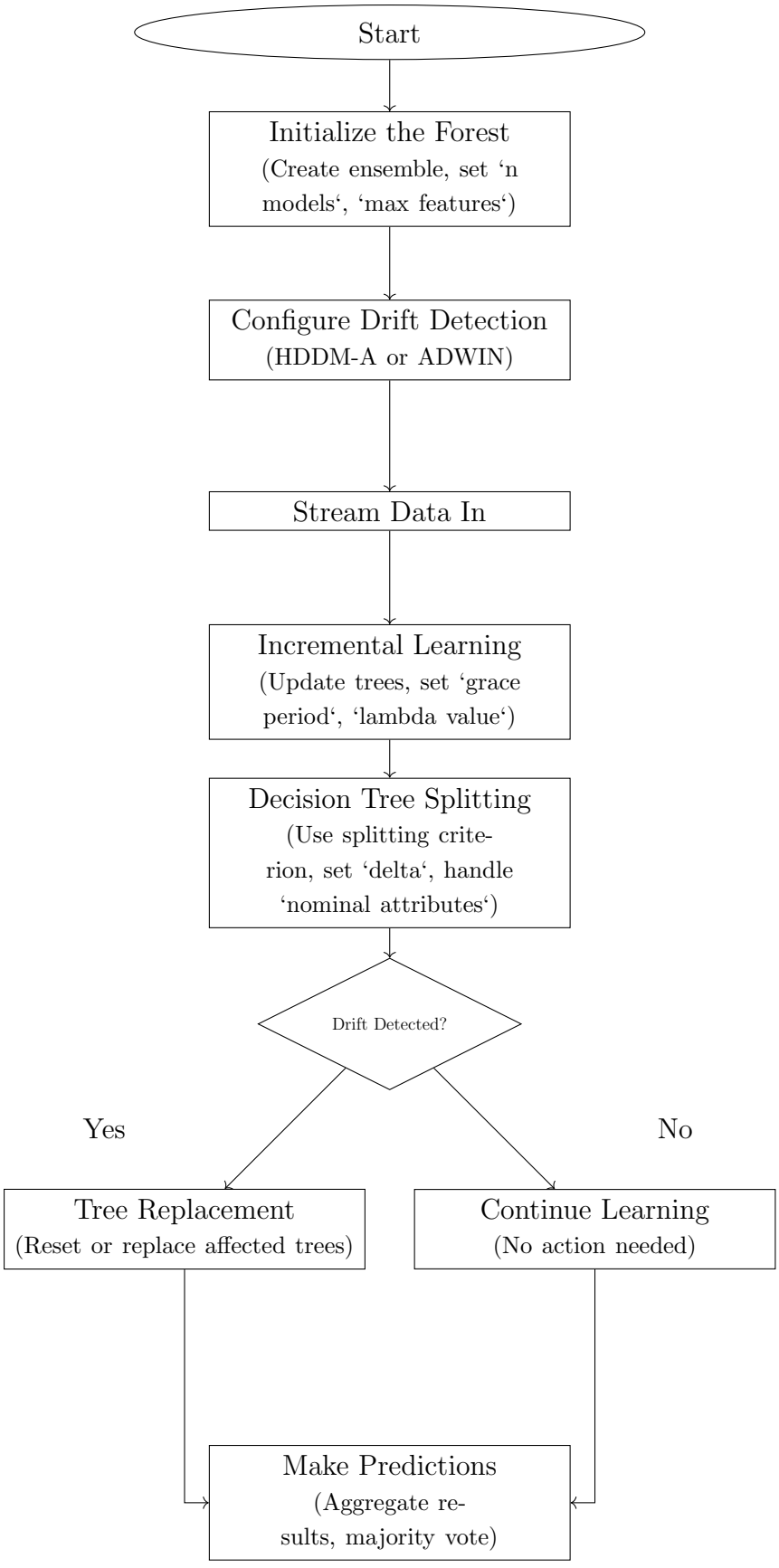


Figure 3: Detailed Steps in Adaptive Random Forest

9 SRPClassifier

Streaming Random Patches Classifier , is an ensemble method commonly used in real world applications and data streams , that combines bagging and random subspaces strategies along with drift monitoring and handling making the model robust against changes in the data distribution over time .

Key concepts in SRPClassifier include:

- **Ensemble Method**:

  The ensemble is made from different models where each model is trained on a different subset of data for more diversity .

- **Random Subspaces**:

  The principe of random patches or random subspaces is that each model is trained on a random subset of features for even more diversity in the ensemble

- **Drift Detection**:

  SRP optionally have drift detectors to monitor changes in the data distribution , in case of a detected drift SRP will react by either retraining the ensemble on the new data or resetting individual models from the ensemble .

Overall, SRPClassifier is well-suited for real-time applications where data patterns may evolve over time. By combining ensemble learning with drift detection mechanisms, it maintains high accuracy and adaptability in dynamic environments.

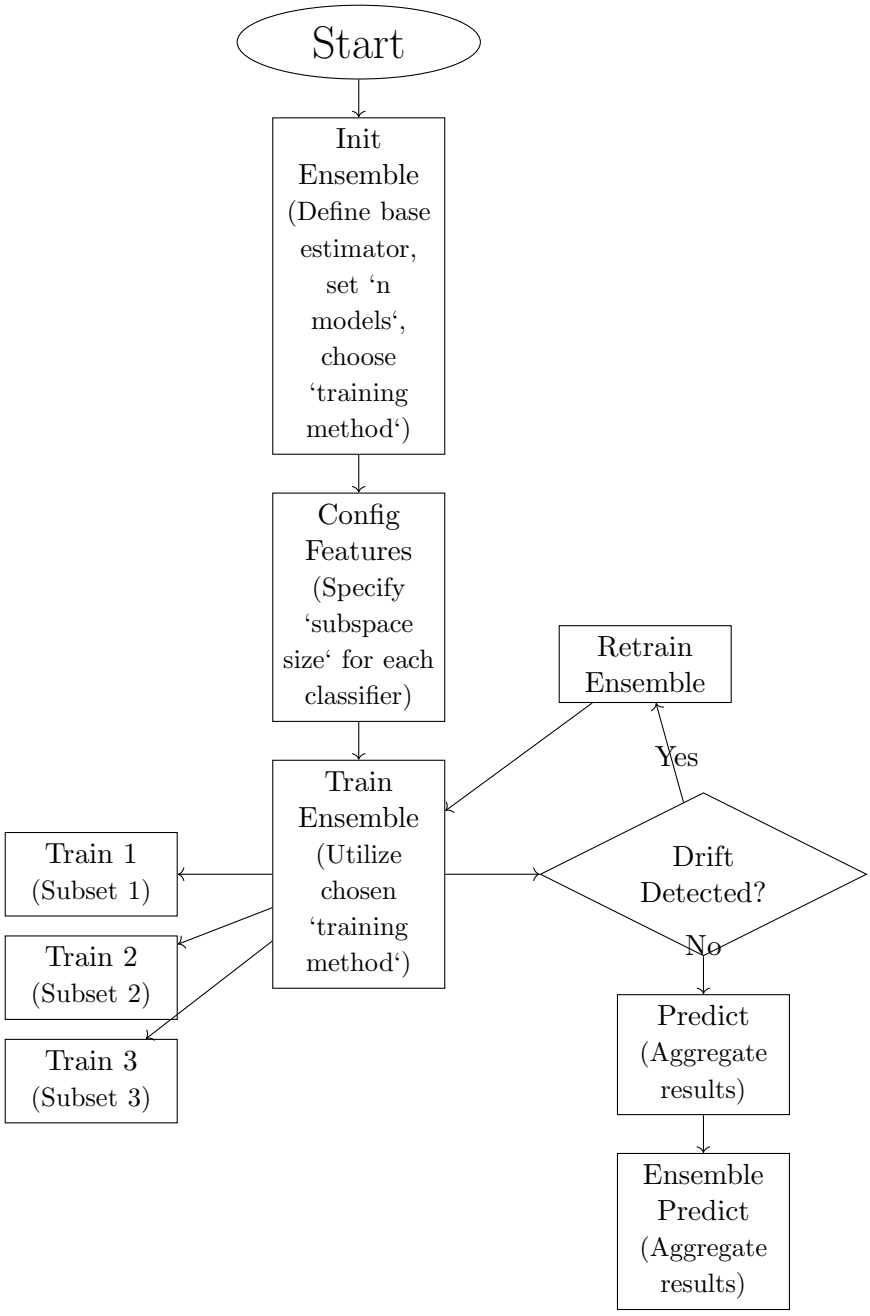


Figure 4: Detailed Steps in SRPClassifier

10 Fair-Weighted Dynamic Ensemble Based on Hybrid Drift Detection and Champion Selection

10.1 Fair-Weighted Ensemble Based on Hybrid Drift Detection

This new approach combines different classifiers with a mix of gradual and abrupt detectors so we won't miss anything in the data stream and uses a voting system where the classifier and the detector with the highest error and least error have the upper hand

Key Steps:

1. **Initialization of Ensemble:** - Define base learners usually arf and srp and set the number of models we can do a 4 classifier combination a 2 classifier combination . - Choose drift detectors for each base learner either sudden detectors like ddm , eddm or gradual like adwin hddma.

2. **Training Base Learners:** - here we are going to simply train our classifiers using our training data 3. **Updating Models :** - For each new sample, we are going to update the base learners and their performance metrics(accuracy,recall,precision,f1-score) .

4. **Calculating Weights:** - Here we're going to Calculate the weights of all the classifiers based on their performance . And the classifiers with the better performance at this stage have a higher weight in the voting system .

5. **Ensemble Prediction:** - Then we make the final ensemble prediction and the probability of each base learner in the voting system is based on it's own weight that we calculated it in step 4.

6. **Updating Ensemble Metrics:** - last , we are going to Update the ensemble's accuracy based on the ensemble's prediction .

Why It Works:

1. **Combining Gradual and Abrupt Drift Detectors:** - the model can adapt to various types of model drift if there's a scenario where there's both gradual and sudden drifts reoccuring in the data stream and the classical arf and src classifiers can only detetct one type and can't detect the other , our approach is aware to both types and adapt to them .

2. **Error-Based Weighting:** - Using the error-based weighting will give the learners with the highest performance as we said before the upper hand and will have the highest weights and will be prioritized in the decision making .

3. **Dynamic Learning:** - The base learners will always monitors and updates it's metrics and weights and the ensemble will update itself in a continuous way with new data coming , this dynamic learning process will make the model roust and valid as new data and new concepts arrives.

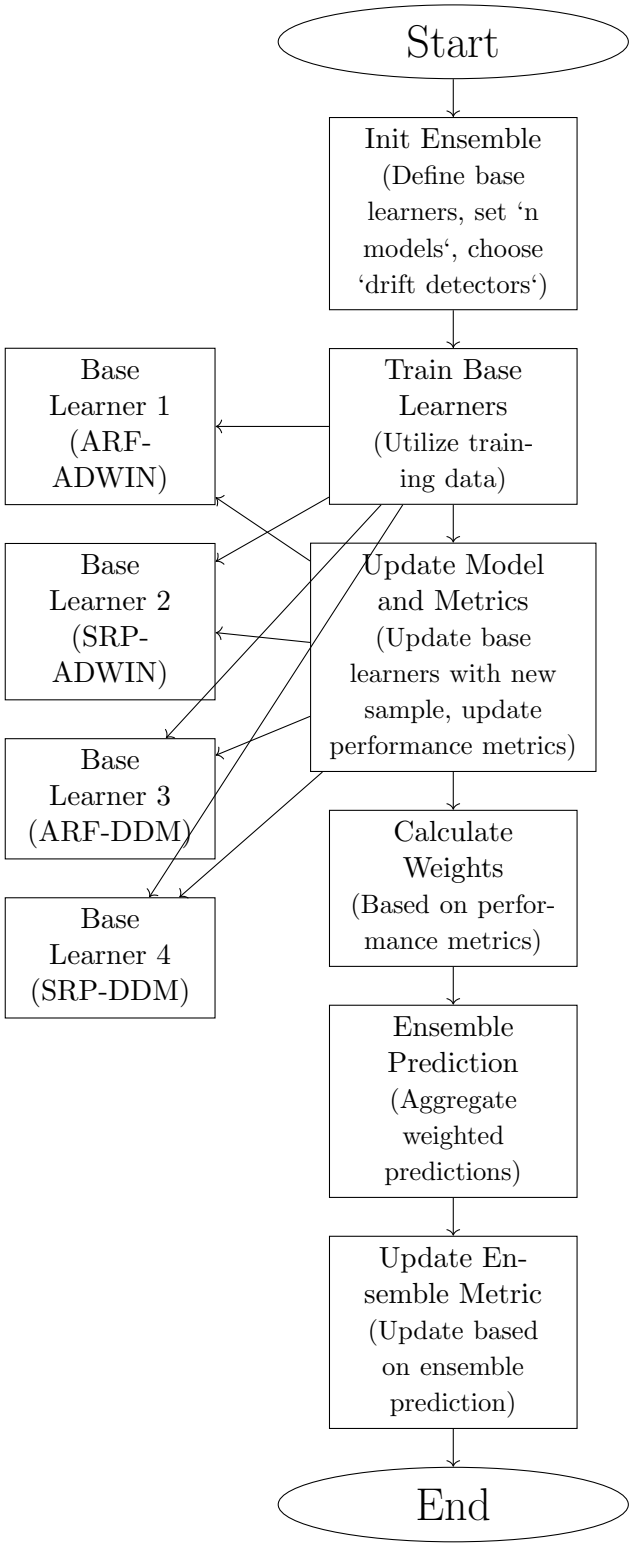


Figure 5: Detailed Steps in Fair Weighted Ensemble Based on Hybrid Drift Detection

10.2 Champion Approach:

Here we are going to do all the combinations possible single base learners like arf with adwin and srp with hddma , 2 base learners 3 base learners 4 base learners 6 base learners and 8 base learners 2 major classifies with 4 detctors can give us 45 different compositions which will monitor and handle slightly different than each other , now to answer what's the best combination possible? is it a solo base learner or a duo base learner or the most complex base learner? and base on what criteria is it the absolute raw performance or the accuracy / time ?, we did the champion mechanism where we regroup the models based on how many base learners and take only from each group the one with the highest accuracy possible and we put them in a list sorted from the least complex to the most complex we start with i=0 and compare it to the i+1 and if the accuracy difference is at least equal to the required-increase which is a value that we should put on based on our needs we can put it really small so we can always have the model with the highest performance possible without taking into consideration the time factor , this work best if we are not in a rush situations , or we can set the value to 0.5% or 1% or even 2 or 3% per step to make sure that the more complex model is worthy enough to be the champion , we compare i with i+1 if the difference is at least 0.5% for exemple that means the i+a which is the 2 base learner is the new champion we reset the index the 2 base learner is now i and we continue doing the same thing , now if the difference between i and i+1 is not at least 0.5% that means that i which is the 1 base learner remains the champion and we will continue comparing now we add another step and we compare it to i+2 which is the 3 base learner and as we said whenever we add a step we add an additional 0.5% so the condition is now at least 1% if the difference is not at least 1% that means again i is still the champion and we continue until we reach the end if the differene between the 8 base learner and whatever model with whatever complexity is at least the required-increase based on the two that means the 8 base learner is the new champ, if not that means the other model will be the new champ and we will use him in that specific season while retraining this dynamic ensemble again with the new data occasionally to always have the best model the best combination possible .

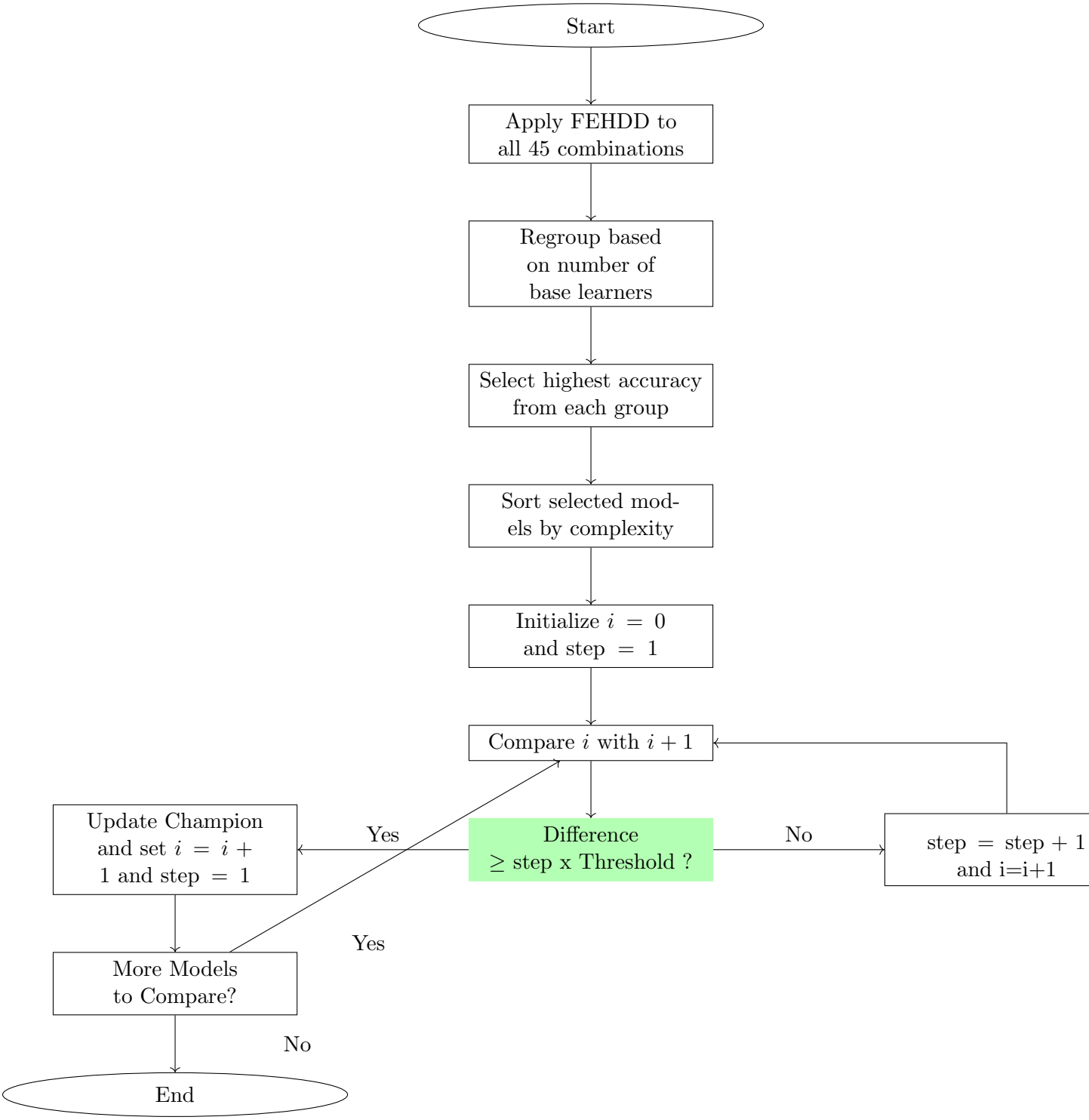


Figure 6: Champion Selection Algorithm Flowchart

11 Overview On The Datasets Used

we used 2 different datasets , the elec2 dataset [15] ,and cicids2017 dataset [16] . We are going to see them in details this section to have a much clearer idea on what we are working with .

11.1 CICIDS2017 Dataset

The current world has provided a environment that has consistantly been dynamism in nature in terms of cybersecurity threats and attacks. vectors emerging regularly. Interested researchers therefore need to come up with proper IDS frameworks and methods in order to reach for the best results. crucial for implementing the sophisticated models to train the computational algorithms on extensive and accurate databases of normal and malignant biomarkers. malicious network traffic. These are CICIDS2017 which is a dataset developed by the Canadian Institute for Cybersecurity (CIC). Institute of Cyber Security (CIC) of the University of New Brunswick New Brunswick in Canada. The CICIDS2017 dataset is an open source dataset quite popular within researchers, who focus on network security. say, specifically for the development, testing, and evaluation of IDS to support its operation. Almost all cybersecurity studies require a high-quality dataset to make their findings accurate as possible. Tra- ditional datasets are found lacking in various perimeters, including the variety of attacks and the fidelity of the traffic models in terms of approximating real traffic patterns, and the level of detail incorporated. The CICIDS2017 dataset addresses this problem by offering diverse and multi-faceted network traffic data samples. This includes a vast cover of normal as well as malicious activities which are displayed in the following figure. This dataset designed specifically for mimicking real network patterns and is therefore very close to real life application. contemporary IDS research.

Another important aspect of the CICIDS2017 dataset is the realistic attack scenarios to which it is sensitive. includes. The dataset seems to contain a large number of attacks including Denial of Service or DoS attacks. Sustained low-level attacks, network scans and probes, data or Web site alterations, impersonation attacks and. to detect DoS, DDoS, malware, network intrusion, p2p traffic, ICMP flood, SCADA systems, botnet traffic, and port scanning. Every attack type is developed deliberately and specifically to follow the given scenario. global threats, it can be inferred that the data obtained encompasses potential threats that are not only diverse, but also credible. This Flexibility is important given that developing IDS models to recognize diverse forms of attack patterns is essential. behaviors. In addition, the dataset is well divided and each flow or packet is labeled with a mark used in the analysis. either benign or malicious. Some extra information for the malicious traffic could include the type of attack it represents.

and sub-type are provided. These labels are incredibly helpful and are used for training and testing of the machines. languages to learn models, since the validation of models requires the known ground truth. accurately

11.1.1 Dataset Features and Explanation

The CICIDS2017 dataset includes several feature as follows that give insights about the network traffic behavior. Here is an explanation of each feature:

1. **Flow Duration:** Represents the total duration of the flow, measured in microseconds. It captures the time elapsed between the first and last packet of a flow.

2. **Total Length of Fwd Packets:** Total length of all forward packets in the flow, measured in bytes. Forward packets are those transmitted from the source to the destination.

3. **Fwd Packet Length Max:** Maximum length of forward packets in the flow, measured in bytes. Provides the size of the largest forward packet within the flow.

4. **Fwd Packet Length Mean:** Mean length of forward packets in the flow, measured in bytes. Represents the average size of forward packets transmitted in the flow.

5. **Bwd Packet Length Max:** Maximum length of backward packets in the flow, measured in bytes. Backward packets are those transmitted from the destination back to the source.

6. **Bwd Packet Length Min:** Minimum length of backward packets in the flow, measured in bytes. Provides the size of the smallest backward packet within the flow.

7. **Flow IAT Mean:** Mean inter-arrival time (IAT) of flow packets, measured in microseconds. Inter-arrival time is the time elapsed between the arrival of successive packets in the flow.

8. **Flow IAT Min:** Minimum inter-arrival time (IAT) of flow packets, measured in microseconds. Represents the shortest time interval between the arrival of successive packets in the flow.

9. **Fwd IAT Min:** Minimum inter-arrival time (IAT) of forward packets in the flow, measured in microseconds. Indicates the shortest time interval between the arrival of successive forward packets.

10. **Fwd Header Length:** Length of headers in forward packets of the flow, measured in bytes. Includes the size of protocol headers (e.g., TCP, UDP) in forward packets.

11. **Bwd Header Length:** Length of headers in backward packets of the flow, measured in bytes. Includes the size of protocol headers (e.g., TCP, UDP) in backward packets.

12. **Fwd Packets/s:** Rate of forward packets transmitted per second in the flow. Indicates the number of forward packets sent over time, providing insights into flow intensity.

13. **Bwd Packets/s:** Rate of backward packets transmitted per second in the flow. Indicates the number of backward packets sent over time, complementing forward packet rate.

14. **Min Packet Length:** Minimum length of packets in the flow, measured in bytes. Provides the size of the smallest packet transmitted within the flow.

15. **URG Flag Count:** Number of packets in the flow with the Urgent flag set. The Urgent flag is used in TCP packets to indicate urgent data.

16. **Down/Up Ratio:** Ratio of the number of packets with a downward direction to those with an upward direction in the flow. Indicates the balance or asymmetry in packet flow direction.

17. **Init_Win_bytes_forward:** Initial window size in bytes for the forward direction of the flow. Represents the size of the initial TCP window advertised by the sender.

18. **Init_Win_bytes_backward:** Initial window size in bytes for the backward direction of the flow. Represents the size of the initial TCP window advertised by the receiver.

19. **min_seg_size_forward:** Minimum segment size for the forward direction of the flow, measured in bytes. Indicates the smallest TCP segment size accepted by the sender.

20. **Labelb:** Binary label indicating whether the flow is benign (0) or malicious (1). Malicious flows include various types of attacks such as DoS, DDoS, brute force, web attacks, etc.

Each of these features provides critical information about the characteristics and behavior of network flows in the CICIDS2017 dataset. They are essential for developing machine learning models that can accurately detect and classify network traffic as benign or malicious. The dataset's comprehensive labeling and diverse attack scenarios make it a valuable resource for cybersecurity research and the development of robust intrusion detection systems.

11.1.2 Dataset Collection

The dataset was captured in a close setting over five days with the experiment starting from the July 3-2017. July 7, 2017. The settings of the collection entailed several networks devices, and traffic generation. they are tools that help simulate real users and automatic attacks aimed at determining the weaknesses of a given website. This section highlights a discussion on the evaluation of the implementation of the strategy, this section includes the sub topics as follows: examine the environment that has been created for the project, equipment applied and the measure undertaken to gather data.

1. **Environment Setup:**

   - **Network Topology:** To recap, the network topology that was employed for data collection incorporated connected machines to notify a real world environment for the learning process of an enterprise network. ment. The topology consisted of:
     - **Servers:** As for local services, different types of servers were created to attend to the public. such as web server computer, data server computer and files server computer etc. These servers were designed to host regular business software as it does not provide extra resources for a virtual machine.

- **Client Machines:** Some of the client machines are operating with different operating systems while others are entirely without such sys- tems (Windows7, Linux and MacOS) was used for making legitimate users traffic and can be used for functions like system access, web surfing, mailing, and file sharing among others.
- **Network Infrastructure:**In terms of networking, the network infrastructures comprised of routers and/or switches. routers, switches, and firewalls to monitor and control the traffic, and to implement security features that have to be applied to the network. This arrangement made the network resemble a real-world corpo-rate system and meant that any protocols returned to their optimal state after being tested.

- **Traffic Generation:**Real world behavior as well as misclicks and use of scripts and bots were also considered. generate network traffic. For typified activity, real users including the authors performed mundane operations with computers especially browsing the web. a web site includes every thing related to the information exchange such as file downloads and email communications. Teleplays were written specifically for create profiles and emulate user behaviors to create normal traffic. Additionally, attack scripts were conducted to mimic different cyber threats in order to test the interactive system.

2. **Tools Used:** For the purpose of utilizing a wide spectrum of approaches, which would facilitate the accumulation of vast amounts of data, the following tools were used:

   - **Packet Capture:** Programming languages and protocols such as Wireshark, tcpdump, and other tools snapped raw. data at the network level of data packets. These tools contain vast amount of data about the students. on each packet along with its header and the payload, so that more deeper analysis can be done.
   - **Flow Generation:** • Flow Generation: Using Argus and nfdump, software programs that are able to transform raw data, the raw data was transformed. packet data into flow-based records.Packets are records that are generated by the analyzer to represent all possible flow regimes. This process involved collecting packets ICT information from different sources and then aggregating them into subject-specific clusters. into flows using the 5-tuple which consists of the Source IP, Destination IP, Source port and the Destination port. socket (transport:TCP/UDP, destination: host, service: port), calculations of the different flow parameters.

11.1.3 Attack Scenarios

The dataset contains various augmentation angles, which represent actual attack threats, we can mention :

1. **Denial of Service (DoS):** DoS attacks intended to cause a system to deny using its services to other users and deny access to its own users. it is a service or a data item which a user has no lawful right to access, or a resource that is beyond the user's access permission, either temporarily or. permanently interfering with connectivity to

a host belonging to the Internet. This type of attack usually tries to send an excessive amount or requests that the target can barely handle. It can be confusing at times, but the key is that it is designed to stop many a genuine request from being met. Examples of DoS attacks in the CICIDS2017 dataset include:Examples of DoS attacks in the CICIDS2017 dataset include:

- **Slowloris:** This attack simply makes multiple connections with the target web server as experienced by the users. and leave them open and operating as long as possible, until all the server resources are depleted. resources.
- **LOIC (Low Orbit Ion Cannon):** LOIC is a tool that will flood the target with TCP, UDP or any other protocol of an organisation's choice. Either to trigger HTTP requests to a server multiple times, or to cause the server to be overwhelmed with too many requests.
- **Hulk:** it is an HTTP LOIC tool that floods HTTP GET and/or HTTP POST requests on the target with a purpose of exhausting the target's server resources.
- **GoldenEye:**
  Like LOIC, GoldenEye is used for accomplishing DDoS attacks because it sends HTTP, UDP, or TCP requests to a target server or performing vulnerability scanning using the OS command injection attacks. attack patterns in the given details in the dataset prove a rich foundation to train IDS to recognize various DoS techniques.

These detailed attack patterns in the dataset provide a rich basis for training IDS to recognize various DoS techniques.

2. **Distributed Denial of Service (DDoS):** Distributed Denial of Service (DDoS): A Distributed Denial of Service (DDoS) attack is a common method where multiple computer systems with infected hosts, they are usually automated bots , are used to attack and take control of a single system, creating a Denial of Service (DoS). In the CICIDS2017 dataset, DDoS scenarios are unknown, and the effect of other attack scenarios is added to mimic attacks coming from many IP addresses, making them difficult to mitigate because the attack traffic is distributed across different locations on the internet. This is important in achieving the objective of recognition mechanisms that will enable an organization to detect and cope with such widespread threats effectively.

3. **Brute Force Attacks:** As for brute-force attacks, the former involves a technique where the hacker tries out all the possible passwords in succession. complicated creations of user names and passwords with an aim of illegally penetrating into the system. Among all the attack types which are present in CICIDS2017 dataset, the brute force attack scenarios are SSH and FTP brute. force attacks. These attacks involve a large amount of traffic for the login related activities in a short time. with the attacker being able to guess the credentials by entering a combination of username and password until correct ones are entered. IDS should detect that brute force attempts by identifying patterns inside the logs. compromised systems early.

4. **Web Attacks:**

   Web attacks are type of cyber threat that attacks web application in order to ravage through its flaws and weak points. gain unauthorized access. In the CICIDS2017 dataset, common web attacks include:In the CICIDS2017 dataset, common web attacks include:

   - **SQL Injection:** Instead, compromised SQL statements are entered in input boxes to manipulate the database in ways that help the crim- ulate the application's database.
   - **Cross-Site Scripting (XSS):** Malicious scripts are introduced to any web addresses being viewed by other users and afterwards capture private data of distinct accounts.

   These attacks are part of the features included within the dataset to assist researchers in the training and evaluation IDS. expensive and powerful protocols that possess the ability to identify and eliminate different web-based dangers.

5. **Infiltration Attacks:** This type of attack imitates an outsider getting into a hostile structure without authorization. the network. Sjamboku, hop, and pivot, other subcategories belonging to this method involve, privilege escalation, and data. The CICIDS2017 dataset comprises of nine types of activity, among them being exfiltration which is addressed by imitating the acts of an insider. a threat or an external attacker who has penetrated to the network boundary layer. Detecting Such attempts to infiltrate the organization is vital in IDS to avoid instances of data breaches and network compromises.

11.1.4 Concept Drift in CICIDS2017

The term concept drift, in the context with regards to a data stream, can is defined as changes over time in because the nature of attacks and network behaviors are changing network security has relevance for intrusion detection. In To ascertain the presence of concept drift in the CICIDS2017 dataset, it becomes apparent in the changes in the distribution of benign and of more than 29 thousand malicious network traffic over the data collection period of July 3-7, 2017. As new attack While new techniques appear and users' behavior changes, IDS need to stay effective to reaching maximum detection accuracy. Several challenges in addressing concept drift in network security include:Several challenges in addressing concept drift in network security include:

- **Dynamic Network Environments:** Real-world networks are not any different and change as they integrate more arising applications. changing the system usage patterns and hydra activities as well as shifting of traffic flows and new barriers to IDS.
- **Emerging Threats:** These are quite common since attackers are always in the process of coming up with new ways of attacking and evading the detections systems. Algorithms, which require updating IDS permanently, involve specific techniques.

- **Model Adaptation:** IDS must then update to better improve its ability to detect such distribution shifts to prevent deterioration. over time in detection performance The median is preferred because the outcome is less sensitive to extreme values compared to mean which falls to a greater extent under the influence of heteroscedasticity and outliers not only of the dependent variable but also of the explanatory ones.

11.2 Elec2 Dataset

The elec2 dataset describes quite a common and crucial type of market concerning electric power, with the following features:

the four crucial elements which include price, demand and flow between NSW and Victoria

(VIC). This subsection aims at highlighting some of the general attributes of the given dataset, as well as the procedures of data gathering and utilization in electricity market analysis.

11.2.1 Dataset Features and Explanations

1. **Date and Day:**

   The **Date** feature provides temporal context to each observation, allowing for time-series analysis of electricity market trends. It includes fractional numbers representing specific dates, enabling researchers to correlate electricity market behaviors with external factors such as holidays or seasonal changes.

   The **Day** feature, encoded as a byte string (e.g., b'2' corresponding to Tuesday), categorizes observations based on the day of the week. This categorical encoding facilitates the analysis of weekly patterns in electricity consumption and pricing, considering factors like weekday versus weekend behaviors or business day impacts on industrial demand.

2. **Period**

   The **Period** feature normalizes time periods within a day, typically represented as a fraction between 0 and 1. This normalization aids in identifying diurnal variations in electricity market dynamics, such as morning and evening peak periods, overnight baseload demands, and transitional periods between peak and off-peak pricing.

   Researchers can analyze period-specific trends to understand consumer behaviors, operational strategies of electricity suppliers, and regulatory influences on pricing mechanisms. For instance, higher normalized periods may indicate peak demand hours requiring load management strategies, while lower periods may reflect minimal consumption during off-peak hours.

3. **Price and Demand:**

   - **NSWPrice and NSWDemand:**
     The **NSWPrice** feature represents the normalized electricity price in New South Wales (NSW) at the time of observation. This metric reflects market dynamics

influenced by supply-demand imbalances, generation mix, fuel costs, transmission constraints, and regulatory policies.

Analyzing **NSWPrice** trends provides insights into price volatility, peak pricing events, and seasonal variations driven by factors such as weather conditions (e.g., heatwaves increasing air conditioning demand) or economic activities affecting industrial consumption patterns. Researchers can identify price elasticity trends, price-responsive behaviors, and the effectiveness of pricing mechanisms in managing peak loads.

The **NSWDemand** feature denotes normalized electricity demand levels in New South Wales (NSW), capturing consumption patterns across residential, commercial, and industrial sectors. Demand fluctuations are influenced by factors including population density, economic activities, weather variations, and consumer behavior changes.

Understanding **NSWDemand** dynamics is crucial for predicting grid stability, optimizing generation capacity, and planning infrastructure investments. Researchers can analyze demand-response initiatives, peak shaving strategies, and the impact of energy efficiency programs on reducing overall electricity consumption.

- **VICPrice and VICDemand**

  Similarly, the **VICPrice** and **VICDemand** features provide insights into electricity market conditions in Victoria (VIC), reflecting regional variations in pricing strategies, demand-supply dynamics, and market integration impacts.

  VICPrice trends highlight price differentials between VIC and NSW, influenced by transmission congestion, interconnectivity limitations, market operator interventions, and regulatory arbitrage opportunities. Analysis of **VICPrice** variations helps assess market efficiency, price convergence trends, and the effectiveness of cross-border electricity trading mechanisms.

  VICDemand metrics reveal consumption patterns specific to Victoria (VIC), influenced by industrial activities, residential behaviors, and commercial operations. Researchers can analyze demand elasticity, sector-specific consumption trends, and the role of renewable energy integration in meeting regional electricity demand.

4. **Transfer:**

   The **Transfer** feature indicates the normalized amount of electricity transferred between NSW and VIC, reflecting inter-regional energy flow and transmission capacity utilization. This metric is critical for assessing market integration, infrastructure adequacy, and operational efficiency in managing cross-border electricity exchanges.

   Researchers analyze **Transfer** dynamics to identify transmission bottlenecks, congestion management strategies, and regulatory impacts on electricity pricing. High transfer values may indicate peak load transfers during periods of regional supply deficits or surplus conditions requiring market operator interventions.

5. **Class:**

   The **Class** feature serves as a binary label ('UP' or 'DOWN'), categorizing observations based on the direction of electricity transfer between NSW and VIC. This classification facilitates the analysis of energy market trends, market efficiency, and regulatory influences on cross-border electricity trading.

   Analyzing 'UP' or 'DOWN' classifications helps researchers understand transmission capacity utilization, market arbitrage opportunities, and the impact of regulatory policies on energy market outcomes. The **Class** label is essential for studying market dynamics during peak demand periods, supply disruptions, and regulatory interventions affecting electricity price differentials between NSW and VIC.

11.2.2 Dataset Collection

The elec2 dataset was collected using automated data acquisition systems and real-time feeds from electricity market operators. The collection process involved:

1. **Data Sources**: For instance, gaining information concerning the electricity market from the regulatory authorities. Auction buyers, market operators, and transmission network service providers. Data sources include prices of electricity, demand statistics, scheduling of transmission, and other operational statistics from generation facilities.

2. **Tools Used**:

   Adopting the automated scripts and data acquisition in the marketplace significantly allow the management and implementation of various strategies successfully. formation tools to obtain the price, demand and transfer data in real time between NSW and VIC. Tactics like Python scripts, data loggers, and API integration make it easier to monitor, analyze, and compare the performance of institutions based on the data collected. tated frequency of data collection and its integration with the databases containing information about electricity market.

3. **Normalization**: Standardizing and formatting data features to enhance the ability of comparing the different datasets. avoid and reduce any possible complex scale-related bias when interpreting the results. Normalization techniques such as in min-max scaling or z-score standardization was used in case the datasets must be preprocessed to be brought on a unified setting. introduced in the aspects such as accuracy during the process of data modeling and predicting future trends.

11.2.3 Market Dynamics

1. **Price Trends:**

   To look at the price fluctuations, weekly or monthly changes for signs of the peak periods, prices at their highest or lowest. Availability and demand changes due to other factors such as naturally developed inclinations in relation to various weather conditions imbalances, and regulatory interventions.

- **Price Volatility:**
  Detection of price volatility, measurement of its severity in specific periods, and factors that affect them how market events affect electricity pricing in NSW , for instance extreme temperatures or storms and VIC markets. That is why the current paper aimed to investigate such volatility measures as standard deviation, coefficient variant and price range analyses to evaluate risks and the level of financial involvement in the market.
- **Seasonal Variations:** A method of assessment of price fluctuations in the course of a year, by differentiating between summer and winter rates demand patterns, and examining the different types of renewable energy generation and contribution, in the current context. erating seasonal price spikes. The following concepts are conventional and observed in most, if not all, sectors and business undertaking: Seasonal adjustment factors, weather normalization methods, and the social trends analysis for historical information used in the seasonal forecast models. long-term energy planning strategies.

2. **Demand Patterns:**

   Insights into consumption patterns across different times of the day and week, highlighting peak demand periods, load shedding strategies, and consumer behavior impacts on market dynamics.

   - **Peak Demand Analysis:**
     Reviews of demand response measures, determination of peak time periods, and assessing the impact results to determine whether energy cons and energy management goals and programs are adequate for peak load conditions. Load factor analysis was one of the key demand metrics employed in the peak demand analysis, as well as other demand. frequency regulation, demand elasticity coefficients, and reactive power or peak shaving techniques for the stability of the grid .
   - **Consumer Behavior:** A related study analyses consumer behavior influence on electricity demand, studying residenecial, business, and industrial uses. Behavioral economics The net benefits subsequent to implementing the chosen principles, consumer preference surveys, and econometric models provided a quantitative measure. effects of price signals and cost, types of tariffs and incentives on energy usage patterns.

3. **Transfer Analysis:**

   Examination of electricity transfer between NSW and VIC, evaluating trends in energy flow, infrastructure utilization, and regulatory influences on market integration.

   - **Inter-regional Flow:**
     Quantification of electricity transfer volumes, assessing inter-regional transmission capacities, and analyzing cross-border electricity trading dynamics. Transmission line utilization rates, congestion management strategies, and market coupling mechanisms informed policy recommendations for enhancing grid reliability and interconnection resilience.

- **Regulatory Impacts:**

  Impact assessment of regulatory policies on electricity transfer dynamics, including market design reforms, pricing mechanisms, and interconnector investment incentives. Regulatory impact analysis (RIA), cost-benefit assessments, and stakeholder consultations informed regulatory decision-making processes to support sustainable energy development and market competitiveness.

11.2.4 Applications

1. **Price Forecasting:** Creation of mathematical models aimed at making a forecast of the electricity prices tendencies according to the historical patterns analysis, machine learning algorithms, and econometric modeling techniques form part of the analysis.

   - **Modeling Approaches:**

     Evaluations of time series forecasting techniques with special reference to autorejective integrate mover average (ARIMA), neural networks and ensemble learning algorithms such as ADADELTA, ADAM, and RMS Prop. gorithms. Therefore, Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) were the performance evaluation metrics chosen for model evaluation. to improve the accuracy of prediction, a measure of variance such as mean square error (RMSE), and forecast accuracy measures. performance and reliability.

   - **Forecasting Horizon:**

     Univariate time series forecasting techniques to forecast short term, medium term, and long term price trends of electricity. to enhance proficiency in energy trading choice, risk management and financial tactics and plans. hedging activities. It involves the horizon planning, scenario analysis and sensitivity. knowledge given by the analysis helped market participants and policymakers, and investors to know the future. necessary changes in market conditions and investing opportunities.

2. **Demand Prediction:**

   Intended to predict electricity usage of customers so that resources, facilities, and investments can be properly allocated and planned for. and energy management policies as well as efficient energy use that has been adopted across the two NSW and VIC regions.

   - **Demand Modeling:**

     The development of demand forecasting models involving regression or time series analysis and promotion. decomposition techniques, and machine learning algorithms in simulation-based environments. Model inputs in-tained sample demographic data, data characterizing the economic situation, weather data, and history. consumption behaviors to forecast the new energy demand of a country or region with high precision. reliability.

   - **Scenario Analysis:** Delphi method, remote forecasting, intuitive methods, and heuristic methods such as, Monte Carlo simulations. the four important statistical

tools are, the average forecast, the trend forecast, the seasonal forecast, the probabilistic forecast, and the sensitivity analysis. Scenario planning techniques that evaluated demand volatilities, market shocks response, and resilience measures. A plethora of projects exist in order to improve the flexibility and the operational condition of the grid in an envisaged high energy market volatility environment.

3. **Market Analysis:**

   Knowledge about how markets work, how regulations affect them, and what policies are aimed at them retail and wholesale electricity tariffs, demand and supply structure of electricity and market segment across the region.

   - **Policy Evaluation:** Effectiveness of energy market policies and regulatory measures is aimed at affecting the electricity market factors. Policy evaluation frameworks involve analyses with cost-benefit analysis, regulation impact statements, and stakeholder management. Stakeholder engagement influences the basis of any policy-making process and the entire market reforms.
   - **Market Efficiency:** Microstructure indicators such as market clearing prices and pricing convergence would also have been examined. The model calculates product testbeds, merging indices, convergence indices, and market power assessments. Efficiency benchmarks, competitive flaws in price restrictions, bidding and purchasing guidelines, and the restructuring of competition policy reviews and market surveillance tools exposed market distortions. Actions to encourage acceptable market competition, including anticompetitive cooperation, were critical for enhancing competition and ensuring consumer welfare.

12 Comparative Evaluation

12.1 Elec2 Dataset Results and Analysis

The performance of various models on the Elec2 dataset has been evaluated. The following table summarizes the accuracy of different models:

Model	**Accuracy (%)**
Hoeffding Tree	76.41
AdaBoost	76.69
Bagging	77.38
(1 base learner) SRP: DDM	85.51
(2 base learner) SRP: HDDM_A & EDDM	85.99
(3 base learner) SRP: HDDM_A & ADWIN & EDDM	85.40
(4 base learner) SRP: HDDM_A & DDM & ADWIN & EDDM	85.53
(6 base learner) ARF and SRP: HDDM_A & DDM & EDDM	85.93
(8 base learner) ARF and SRP: HDDM_A & DDM & ADWIN & EDDM	85.08

Table 1: Results for the elec2 dataset

12.1.1 Analysing the results

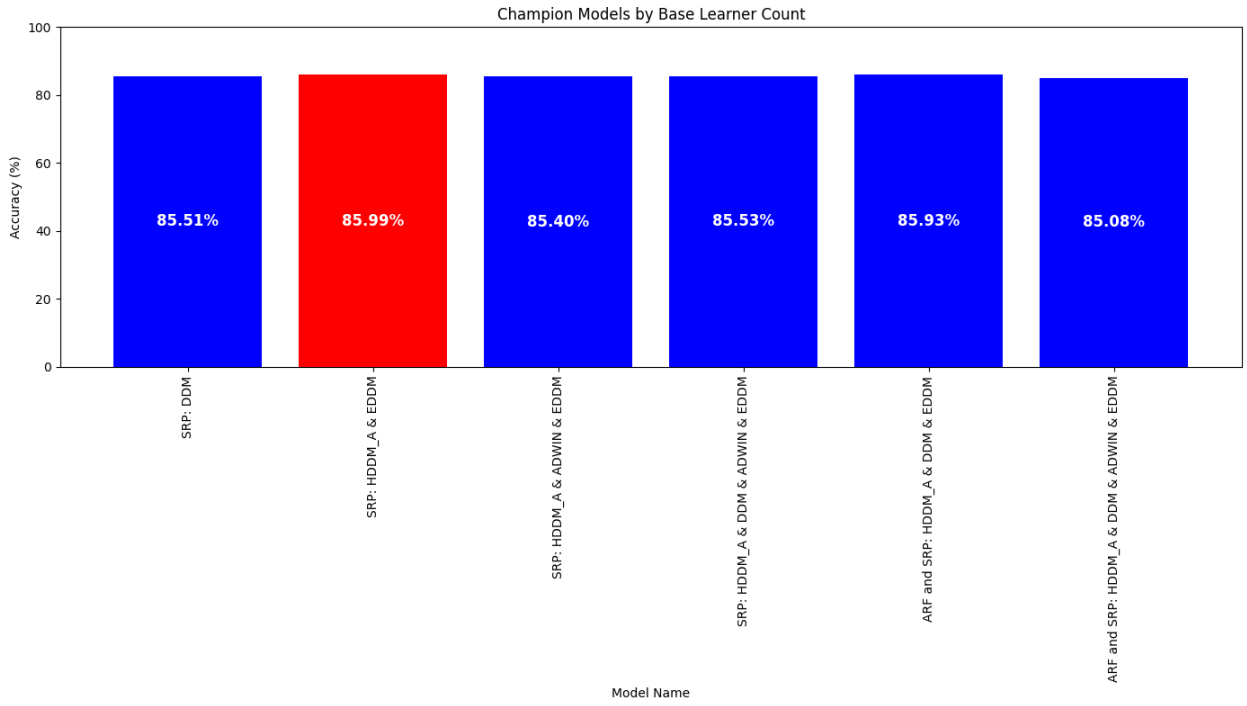
We start with the most basic classifier which is hoeffding tree with an accuracy of 76.41% the smallest accuracy possible , becuase even tho hoeffding tree is usually good for incremental learning and in the fields of data streams but the fact that he has 0 drift awareness and won't even try indirectly like Bagging and Boosting made him the least effective against concept drift , then we have Adaboost and Bagging which both are ensemble methods and very underrated approaches that can work against concept drift in a indirect way , AdaBoost by focusing on the misclassfied instances and Bagging by trying to introduce diversity into data distribution by random sampling and both made slightly better than hoeffding tree with an accuracies of 86.69% and 77.38% , but still not enough because of their lack of drift awareness . now if we comeback to what we said earlier about our new approach is that we try all the combinations possible to figure out which is the best of the best right now , with 2 classifiers and 4 detectors we can have until 45 different combinations the approach will choose only 6 (one from each depth with the highest accuracy) as challengers and one of them will be the ultimate champion based on certain criteria we will talk about it in details in the next subsection , now what we can see that the accuracies are between 85% to 86% which is way higher than the simple approaches , there's no huge differences between the models and what's really interesting is that heavier and more complex models like ARF-SRP using all the 4 detectors which is a 8 base learner performed worse than all the other challengers with an accuracy of 85.08% for exemple , which proves the need of our new approach instead of just using different classifiers and combine different detectors while claiming to have the best decisions and monitorings from different types of drifts and sensitivities . The champion selection which will talk about in the next subsection can even choose the most simple and non complex model using a single specific drift detector like he could choose a more complex combination of course based on the data coming and that's why using this heavy approach is a must from time to time whenever we have a good amount of data to always have the best composition possible .

also the types of drifts that work best based on what we're seeing from all the depths are HDDMA and EDDM who were in every single depth almost , then DDM who proved to be a good choice in solo and in team compositions . and the one with the least influence here in this specific dataset is adwin which is very interesting because in another dataset or after adding new data Adwin can turn from the least favourite to the most picked detector , here because Adwin isn't used a lot maybe it indicates that the data have more abrupt than gradual drifts and that the windowing approaches isn't the best choice specially in lower depth in higher depths we can see adwin makes it worse when he enters the conversation for exemple the 6 base learner ARF-SRP when using HDDMA and DDM and EDDM we had 85.93% but when we just added adwin and the composition turned into an 8 base learner the accuracy actually dropped to 85.08% same with SRP when we used HDDMA and EDDM it is a duo base learner and we had 85.99% but when we added ADWIN the accuracy went down using the 3 base learner composition(SRP using HDDMA and DDM and ADWIN) to 85.40%.

12.1.2 Champion Selection Based on Scenario

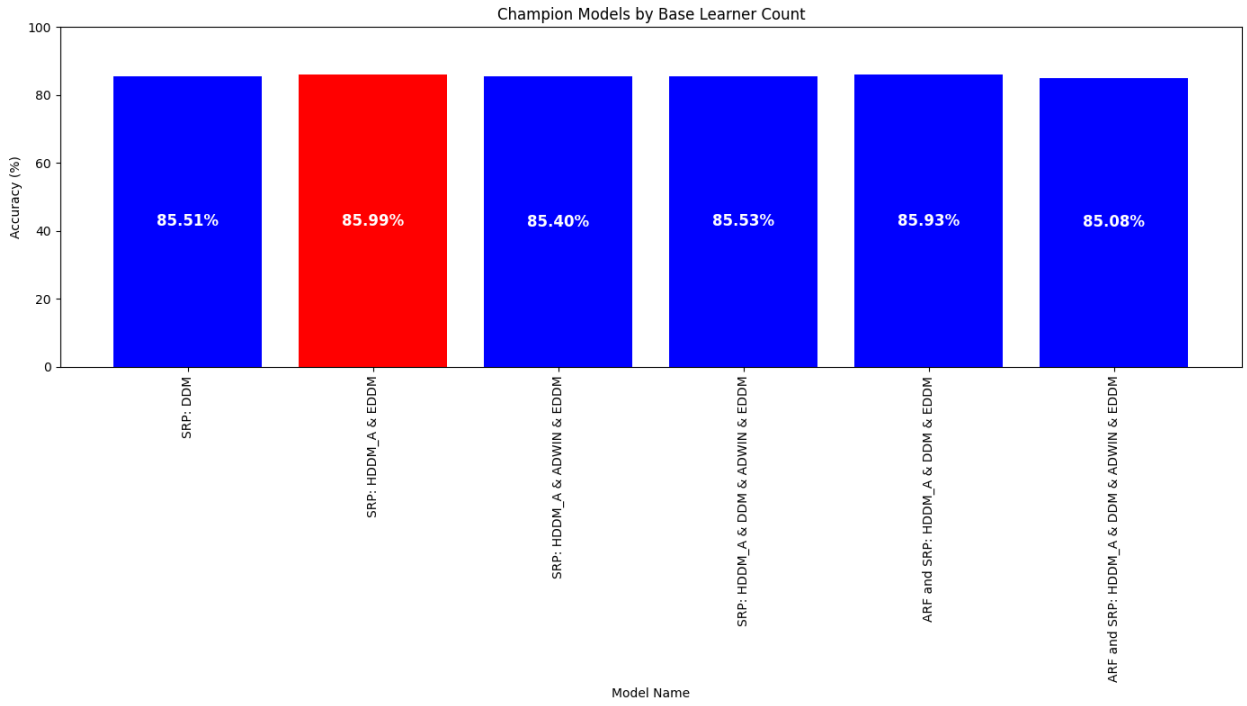
here's the approach choices based on different scenarios from having time pressure which makes a smaller model with decent results the perfect choice even tho he isn't the best performer , to having more time and the more time we have the better model we want it to be even if it's more complex and heavier .

- **Scenario1: Normal circumstances(required-increase=0.2)**



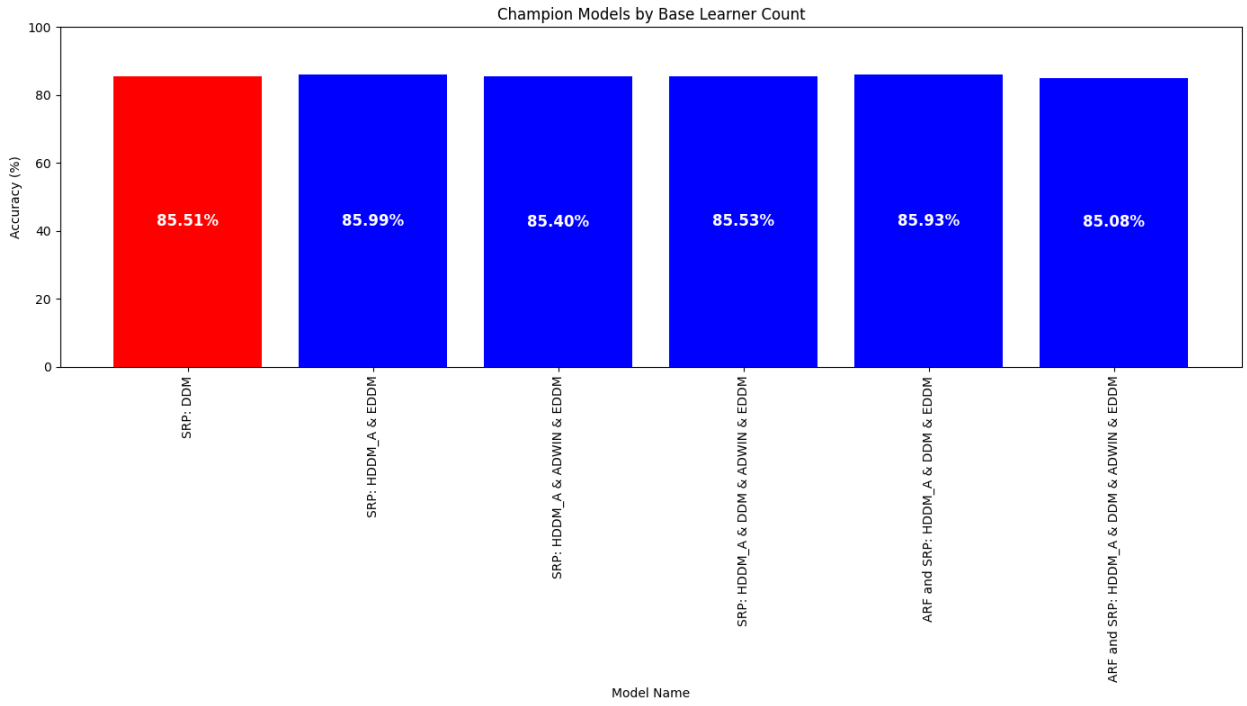
in normal circumstances we want the criteria not to be so high that we prioritize time neither so low that we prioritize the accuracy , here with a required-increase of 0.2% per step which in some datasets isn't even that high if the difference between the depths is huge , here in the elec2 dataset like we saw in the analysis of table1 the results are so close between the depths so the approache compared the first base learner with the second base leanrer (i with i+1). The difference is 0.48¿0.2 which means that SRP with HDDMA and EDDM is the new ultimate champion and after reseting the index we start comparing i with i+1 the differnece between the third and two base learner is actually in negative we move a step we multiply the criteria with the number of steps which is 2 and the difference must be at least the double of the first criteria but still the differnce is in negative so as the rest of the results . which is interesting and that's what we talked about in the problematic , is the more complex and the heavier model always the better ? in this case no , the duo base learner is the ultimate champion with 85.99%

- **Accuracy Over Time(required-increase=0.02)**



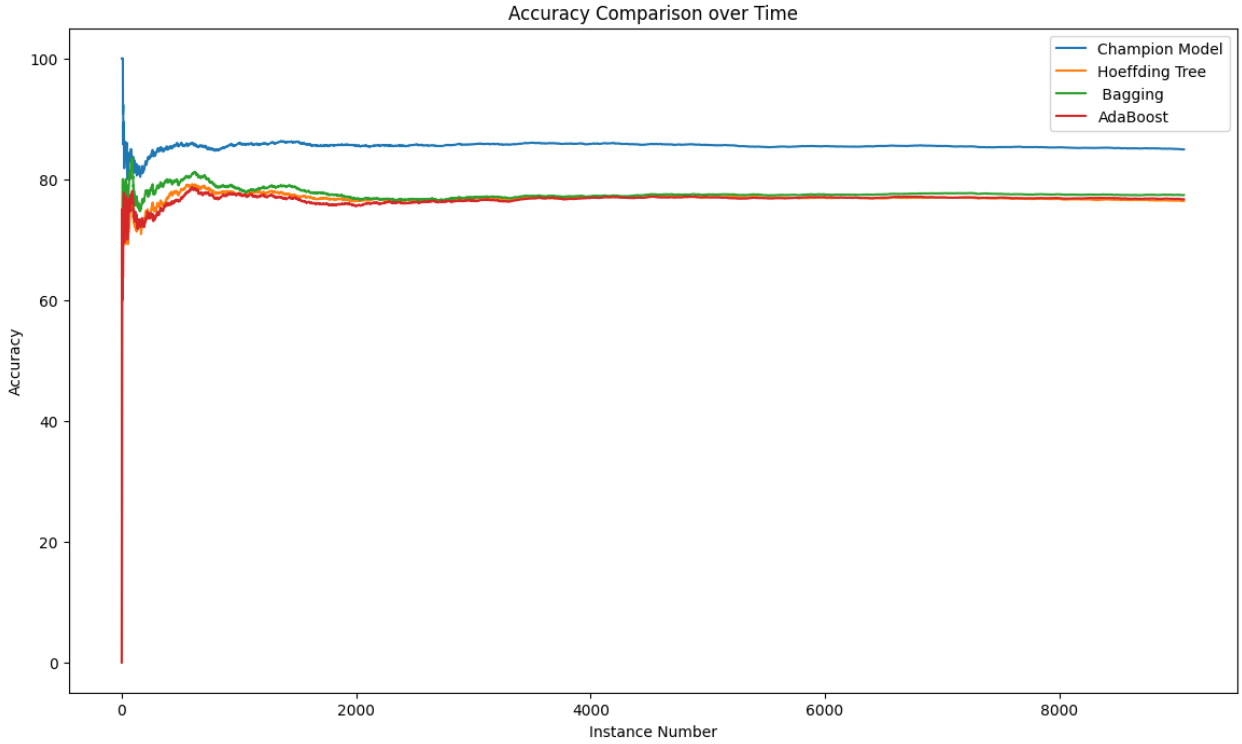
in this types of situations usually we have more time and we always want to choose the best model even if the accuracy isn't worth it in time pressure or even in normal circumstances , here even if the gain is so small it's still a gain , we're gonna make it possible by making the criteria so small that the heavier models will have more chances to be choosen even if they give non impressive gains , here we choose 0.02% and we start comparing i with i+1 the difference between duo base learner and solo base learner is 0.48 which is way more than 0.02 , SRP using HDDMA and EDDM is the new ultimate champion and will still becuase he's the highest in accuracy between all the champions from multiple depth , if we could change some details and made the duo base learner slightly less than the trio base learner the ultimate champion will be the 6 base learner champion , because after making the 3 base learner the champion reseting the index comparing it with the 4 base learner he will become the new ultimate champion for now reseting the index again and compare it to the 6 base learner and the difference is more than 0.02% which means ARF and SRP using HDDMA DDM and EDDM is the new ultimate chanmpion for now , reseting the index again and after the last comparaison the difference is in negative and there's no challenger left which means ARF and SRP using the trio drift detectors HDDMA DDM and EDDM will be the ultimate new champion .

- **Time Over Accuracy**



Usually we need it when we need to be drift aware and adapt in the shortest time possible for exemple 5 minutes or 10 minutes . In this case we actually need to make the criteria much higher so the lower depth models will be choosen and here we decided to set the required-increase to 0.9% which is in this dataset is impossible to hit per step because the elec2 dataset is one of the hardest to make higher gains . We start by comparing i to i+1 the duo base learner as we mentioned earlier is 0.48% higher than the solo base learner which is less than the criteria 0.9% , that means that the new ultimate champion is SRP with DDM . The solo base learner gave decent results and gives results faster than any other challenger here .

- **Plot : Champion Model vs basic approaches**



here we choose SRP with HDDMA and EDDM as the ultimate champion and compared to the most basic classifier which is hoeffding tree and some other underrated approaches like Bagging and AdaBoost that can slightly adapt to drift indirectly , from the plot we can see from the first 100 to 200 instances the more there's up's and down's the more the approach will likely to achieve higher accuracies , let's start with hoeffding tree there's barely reductions in accuracy at first it uses incremental learning so the accuracy is going up slowler than the others , why ? because it's not self aware about the drift and not trying to adapt at some point the accuracy will stop around 76% , then we have AdaBoost which the movement is slightly more dynamic than hoeffding tree there's some reductions in accuracy then boosts in accuracy which means the adaboost was triggered by the reduction and he's trying to adapt by focusing on the misclassified instances but still it lacks that self awareness and the adaptation against the drifts so it end up slightly above hoeffding tree with almost 77%, and then we see the Bagging approach which was more dynamic than the first two , and indeed had slightly better results with almost 78% , and last and not least , the choosen chmpion from our approach FDE-HDDCS had the most dynamic ups and downs in accuracy per 200 instances and in per 2000 instances too , ofc combining the most perfect combination at this specific timeline and by using the most suitable drift detectors in this case HDDMA and EDDM and the best classifier which is SRP only , the model outperformed all the other approaches with 86% accuracy

12.2 CICIDS2017 Results and Analysis

Now, let's consider another dataset and evaluate the models in a similar manner.

Model	Accuracy (%)
Hoeffding Tree	81.31
AdaBoost	91.93
Bagging	87.93
(1 base learner) ARF: EDDM	98.89
(2 base learner) SRP: DDM & EDDM	99.24
(3 base learner) ARF: HDDM_A & ADWIN & EDDM	99.10
(4 base learner) SRP: HDDM_A & DDM & ADWIN & EDDM	99.21
(6 base learner) ARF and SRP: HDDM_A & DDM & ADWIN	99.03
(8 base learner) ARF and SRP: HDDM_A & DDM & ADWIN & EDDM	99.08

Table 2: Results for the cicids2017 dataset

12.2.1 Analysing the results

Here what's interesting from the start is that all the approaches gave better accuracies in this dataset than the Elec2 which is understandable due to the difference in data itself and the number of drifts , starting from the basic one Hoeffding Tree with an accuracy of 81.31% so we had a very decent result even tho it's not perfect in the context of iot data streams but still better on what we had in elec2 . If we are going to ask a questu=ion it's probably gonna be why and the answer is that having this accuracy with just incremental learning with zero drift awareness can only make us conclude that the drift problem isn't that complex here like we saw in elec2 , and what reassure this conclusion is the huge improvement of both the ensemble methods Bagging and AdaBoost with an accuracies of 87.93% and 91.93% , the improvement here is almost 7% from Hoeffding tree to bagging and almost 11% from Hoeffding Tree to Adaboost , the improvements even tho aren't that perfect either but still decent results that are not only better than the elec2 , the percentage of improvements is way more in cicids dataset (11% using Boosting and 7% using Bagging) than the elec2 adding just barely 0.30% for boosting and almost 1% for Bagging . So even without any drift awareness these underrated approches actually worked and adapted dynamically to model drift , proving that these approaches works against concept drift in certain situations where the drift problem isn't a huge issue and isn't that complex but the more complex the drift problem is and the more it is frequent the less these underrated approaches will add to the performance .

Now what we can see from the proposed challengers by our new approach FDE-HDDCS is that the difference between them and other non drift aware approaches is huge , for exemple the least performer of the challengers which is the solo base learner ARF:EDDM has an accuracy of 98.89% which is almost 18% higher than Hoeffding Tree wich is a huge achievement .

Now for all the other more complex challengers their accuracy goes around 99.03% to 99.24% which is higher than the solo base learner, it is understandable at the end because

they are more complex and uses different classifiers and detectors but the difference isn't even that huge . Also from all the challengers the one with the highest accuracy is SRP:DDM-EDDM with an accuracy of 99.24% and the fact he is just a duo base learner that means it isn't that complex , plus the most complex compositions like ARF-SRP:HDDMA-DDM-ADWIN and ARF-SRP:HDDMA-DDM-ADWIN scoring one of the lowest accuracies (99.03% and 99.08%) proves that the more classifiers and detectors doesn't mean it's the best pick neither the best performer .

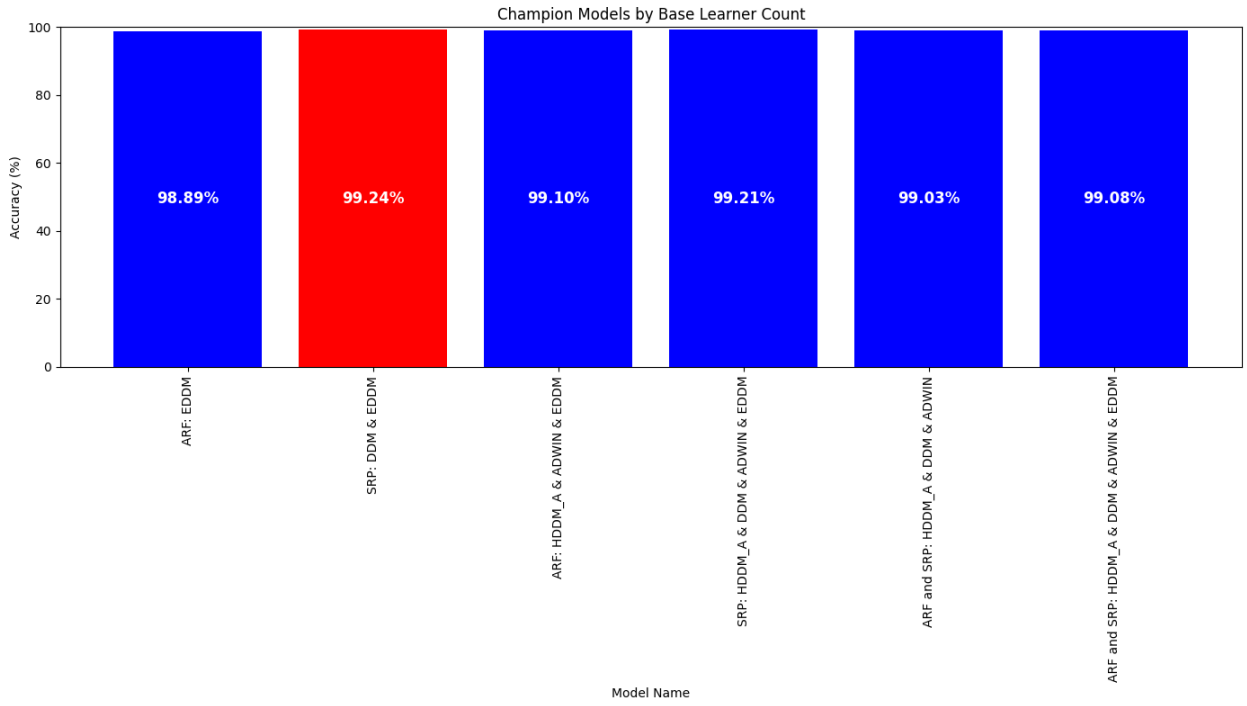
Also a very interesting point is the changes of the challenger from the last dataset , here ARF as a classifier had more recognition and was used in almost every single depth while in elec2 was only used in 6 and base learner compositions , also the best pick for drift used to be HDDMA and EDDM then DDM in elec2 now DDM and EDDM are the best picks then HDDMA and ADWIN in third place , we can also mention that ADWIN had a buff in this dataset and used in almost every single depth except the solo and duo base learners , and HDDMA got a huge nerf here by not being in the solo neither duo base learners which are the most 2 depths that shows the importance of the detector or classifier because the higher the depth the less important they become .

12.2.2 Champion Selection Based on Scenario

here's the approach choices based on different scenarios from having time pressure which makes a smaller model with decent results the perfect choice even tho he isn't the best performer , to having more time and the more time we have the better model we want it to be even if it's more complex and heavier .

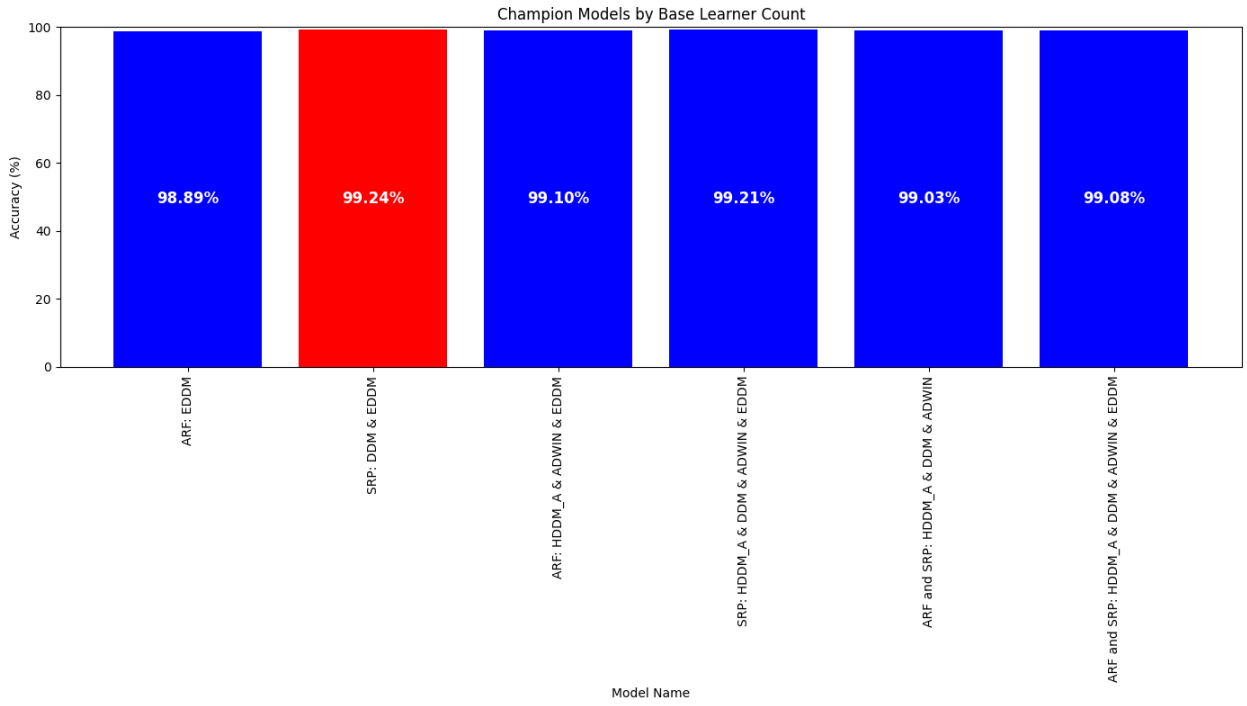
The selection process here is almost identical to what happened in the elec2 but of course with different performances and the classifiers and detectors are different on most depths , what i want to say is that choosing the solo base learner and duo base learner in both datasets doesn't mean they are always the best pick it just happened as a coincidence just to make it clear.

- **Scenario1: Normal circumstances(required-increase=0.2)**



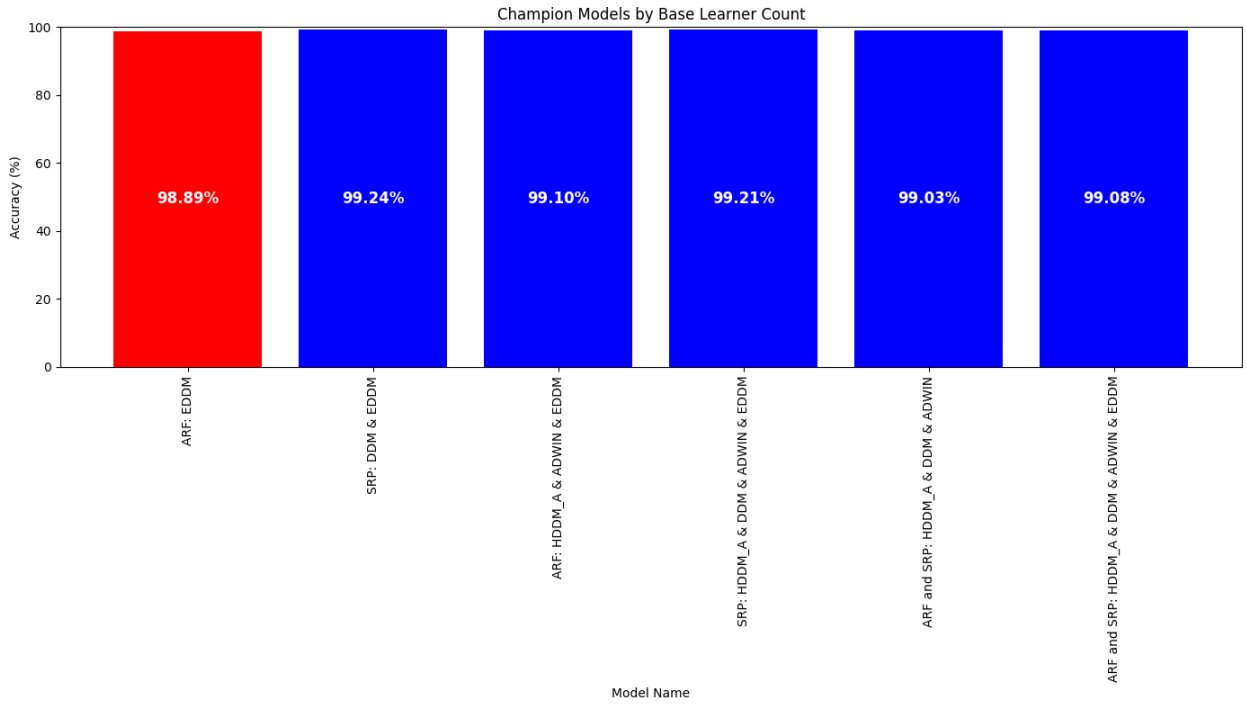
in normal circumstances we want the criteria not to be so high that we prioritize time neither so low that we prioritize the accuracy , here with a required-increase of 0.2% per step which in some datasets isn't even that high if the difference between the depths is huge , here in the cicids2017 dataset like we saw in the analysis of table2 the results are so close between the depths so the approache compared the first base learner with the second base leanrer (i with i+1). The difference is 0.35¿0.2 which means that SRP:DDM-EDDM is the new ultimate champion and after reseting the index we start comparing i with i+1 the differnece between the third (ARF:HDDMA-DDM-ADWIN) and two base learner is actually in negative we move a step we multiply the criteria with the number of steps which is 2 and the difference must be at least the double of the first criteria but still the differnce is in negative so as the rest of the results . which is interesting and that's what we talked about in the problematic , is the more complex and the heavier model always the better ? in this case also the answer is no like we had in the elec2 dataset , SRP:DDM-EDDM is the ultimate champion with 99.24%

- **Accuracy Over Time(required-increase=0.02)**



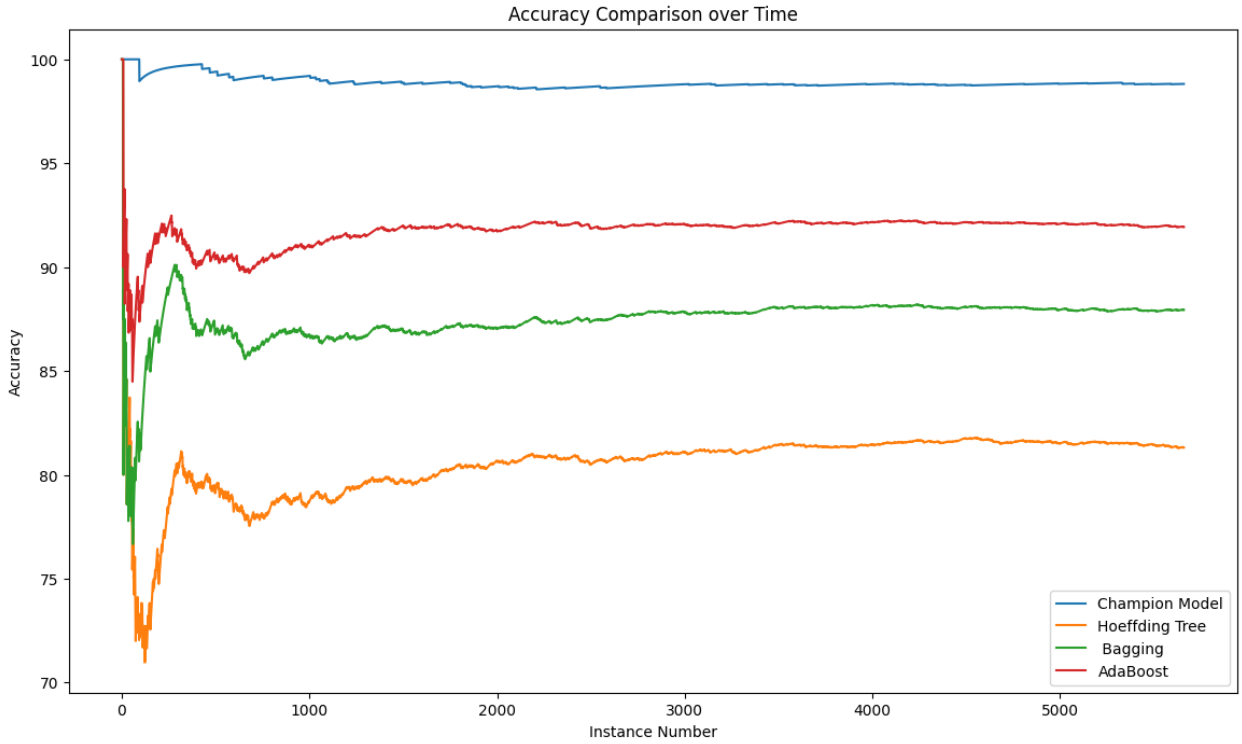
in this types of situations usually we have more time and we always want to choose the best model even if the accuracy isn't worth it in time pressure or even in normal circumstances , here even if the gain is so small it's still a gain , we're gonna make it possible by making the criteria so small that the heavier models will have more chances to be choosen even if they give non impressive gains , here we choose 0.02% and we start comparing i with i+1 the difference between duo base learner(SRP:DDM-EDDM) and solo base learner(ARF:EDDM) is 0.35 which is way more than 0.02 , SRP using DDM and EDDM is the new ultimate champion and will still becuase he's the highest in accuracy between all the champions from multiple depths with an accuracy of 99.24% .

- **Time Over Accuracy**



Usually we need it when we need to be drift aware and adapt in the shortest time possible for exemple 5 minutes or 10 minutes . In this case we actually need to make the criteria much higher so the lower depth models will be choosen and here we decided to set the required-increase to 0.9% which is impossible here and we did that not by mistake but to prove a point .And what mekes it impossible is that the accuracies are so tight and close to each other . We start by comparing i to i+1 the duo base learner as we mentioned earlier is 0.35% higher than the solo base learner which is less than the criteria 0.9% , that means that the new ultimate champion is ARF:EDDM with an accuracy of 98.98% . The solo base learner gave decent results and gives results faster than any other challenger here .

- **Plot : Champion Model vs basic approaches**



Here we choose the best performer possible SRP:DDM-EDDM and we compare it to the other basic approaches in a plot where we see the accuracy per number of samples . The first thing that we notice is that the accracies are far away from hoeffding tree , both Adaboost and Bagging are much higher than Hoeffding tree but still won't be near the proposed ultimate champion .

also a really interesting idea here is that the champion model is surprisingly isn't that dynamic here unlike the other 3 approaches , which proves our point earlier that the drift problem here even tho it exists but isn't that complex so there isn't a lot of downs then ups made by the ultimate champion meaning that he detected much fewer drifts and adapted quickly then the accuracy became stable at almost 2000 instances . Also from 500 to 800 when the other approaches kept going lower and lower the champion model slightly goes down then goes up which means that this phase probably have the most amount of drifts in the entire dataset because the champion models adapts quicly because he is drift adaptive and he uses the best combination of detectors for monitoring while the other basic non drift aware approaches like Hoeffding Tree , AdaBoost and Bagging kept having nerfs after nerfs in accuracy because they aren't aware and adaptive to them . So the drift density in this phase explains the huge reduction in accuracy for the 3 basic approaches .

and of course before and after this phase all the approaches are going either up because all of them are incremental learning based approaches and this clearly happens from 0 to 500 instances, or becomes stable at some point with which is from 2000 to 5000.

13 Conclusion

We talked in this study about concept drift and how it influences the accuracy whenever a new concept is added or an old concept readded, and we saw the different solutions from the most basic ones like Hoeffding Tree Bagging and AdaBoost to models more robust to concept drift like ARF and SRP using the different drift detectors such as ADWIN, DDM, HDDM_A. We saw our new novel approach, FairWeighted Ensemble based on Hybrid Drift Detection (FEHDD), and how the voting system and the combination of both gradual and abrupt detectors gave us better results in some situations. then we saw how we developed the same approach to make it exploring all the possible combinations possible and then applying the champion mechanism on the challengers to choose dynamically one composition which is the best based on certain criteria and under certain specific conditions .

This new approach FDE-HDDCS will always choose the best composition possible based on our needs sometimes a simple solo base learner is what we need sometime a more complex composition with different classifiers and detectors are needed , also the error based voting system and the use of hybrid drift detection proved to have better performance on both datasets than a solo base learner with only one drift detector showing that even the initial approach can gives us some good results too.

Now we're not saying this work is perfect , because even tho the champion model and this approach won't work together and the FDE-HDDCS is used not that frequently maybe sometimes on a daily basis but it's still needs some optimization , and maybe adding new functionalities like adding dynamic feature selection each time a drift is detected or regularizations techniques that updates dynamically . these are some of the features that will work on them in future works .

Also in the near future working on the making the required increase dynamic based on mathematical formula that we need to create that take into consideration statistical data and the accuracies and do different thresholds not only one that multiply whenever we move a step and compare the current champion to a more complex composition .

References

[1] Wang, H., Fan, W., Yu, P. S., & Han, J. (2003). Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 226-235).

[2] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)*, 46(4), 1-37.

[3] Baena-García, M., del Campo-Ávila, J., Fidalgo, R., Bifet, A., Gavaldà, R., & Morales-Bueno, R. (2006). Early drift detection method. In *Fourth International Workshop on Knowledge Discovery from Data Streams*.

[4] Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., & Gavaldà, R. (2007). New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 139-148).

[5] Ross, G. J., Adams, N. M., & Tasoulis, D. K. (2011). Nonparametric drift detection for adaptive learning from streaming data. *Machine Learning*, 82(3), 287-315.

[6] Datar, M., Gionis, A., Indyk, P., & Motwani, R. (2002). Maintaining stream statistics over sliding windows (pp. 41-52). In *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*.

[7] Montgomery, D. C., Peck, E. A., & Vining, G. G. (2001). *Introduction to Linear Regression Analysis* (3rd ed.). John Wiley & Sons.

[8] Kifer, D., Ben-David, S., & Gehrke, J. (2004). Detecting change in data streams. In *Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB Endowment* (Vol. 30, No. 2, pp. 180-191).

[9] Ross, G. J., Adams, N. M., & Tasoulis, D. K. (2012). Exponentially weighted moving average charts for detecting concept drift. *Pattern Recognition Letters*, 33(2), 191-198.

[10] Žliobaitė, I. (2013). On evaluation of adaptive models for evolving data streams. In *Proceedings of the 22nd International Conference on Artificial Neural Networks* (pp. 3-15).

[11] Pinho, A. J., & Santos, M. F. (2012). Dynamic Classifier Selection for Drift Detection.

[12] Frías-Blanco, I., del Campo-Ávila, J., & Herrera, F. (2015). Online and Non-Parametric Drift Detection Methods Based on Hoeffding's Bounds.

[13] Domingos, P., & Hulten, G. (2000). Hoeffding Trees: Scalable Decision Trees for Streaming Data.

[14] Gomes, H. M., Borba, G. H., Bifet, A., & Gama, J. (2017). Adaptive Random Forests: Ensemble Methods for Concept Drift.

[15] A. Bifet and R. Gavalda, "ELEC2 Dataset," *OpenML*, Available: `https://www.openml.org/d/151`. [Accessed: 23-Jun-2024].

[16] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "CICIDS2017 Dataset," *Canadian Institute for Cybersecurity (CIC)*, Available: `https://www.unb.ca/cic/datasets/ids-2017.html`. [Accessed: 23-Jun-2024].