Towards Fairness-aware Online Machine Learning from Imbalanced Data Streams

Farnaz Sadeghi

Thesis submitted to the University of Ottawa
In partial fulfillment of the requirements
For the Ph.D. degree in
Computer Science

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Examining Committee

The following served on the Examining Committee for this Doctoral Thesis.

External Member: Jerzy Stefanowski
Professor, School of Computing Science
Poznan University of Technology, Poland

Carleton Member: Wei Shi
Professor, School of Information Technology
Carleton University, Ottawa

Internal Member: Paula Branco
Assistant Professor, School of Electrical Engineering and Computer Science
University of Ottawa

Internal Member: Nathalie Japkowicz
Professor, School of Computer Science
American University, USA

Supervisor: Herna L. Viktor
Professor, School of Electrical Engineering and Computer Science
University of Ottawa

Abstract

Online supervised learning from fast-evolving imbalanced data streams has applications in many areas. That is, the development of techniques that are able to handle highly skewed class distributions (or 'class imbalance') is an important area of research in domains such as manufacturing, the environment, and health. Solutions should be able to analyze large repositories in near real-time and provide accurate models to describe rare classes that may appear infrequently or in bursts while continuously accommodating new instances.

Although numerous online learning methods have been proposed to handle binary class imbalance, solutions suitable for multi-class streams with varying degrees of imbalance in evolving streams have received limited attention. To address this knowledge gap, the first contribution of this thesis introduces the Online Learning from Imbalanced Multi-Class Streams through Dynamic Sampling (DynaQ) algorithm for learning in such multi-class imbalanced settings. Our approach utilizes a queue-based learning method that dynamically creates an instance queue for each class. The number of instances is balanced by maintaining a queue threshold and removing older samples during training. In addition, new and rare classes are dynamically added to the training process as they appear. Our experimental results confirm a noticeable improvement in minority-class detection and classification performance. A comparative evaluation shows that the DynaQ algorithm outperforms the state-of-the-art approaches.

Our second contribution in this thesis focuses on fairness-aware learning from imbalanced streams. Our work is motivated by the observation that the decisions made by online learning algorithms may negatively impact individuals or communities. Indeed, the development of approaches to handle these concerns is an active area of research in the machine learning community. However, most existing methods process the data in offline settings and are not directly suitable for online learning from evolving data streams. Further, these techniques fail to take the effects of class imbalance, on fairness-aware supervised learning into account. In addition, recent fairness-aware online learning supervised learning approaches focus on one sensitive attribute only, which may lead to subgroup discrimination. In a fair classification, the equality of fairness metrics across multiple overlapping groups must be considered simultaneously. In our second contribution, we thus address the combined problem of fairness-aware online learning from imbalanced evolving streams, while considering multiple sensitive attributes. To this end, we introduce the Multi-Sensitive Queue-based Online Fair Learning (MQ-OFL) algorithm, an online fairness-aware approach, which maintains valid and fair models over evolving streams. MQ-OFL changes the training distribution in an online fashion based on both stream imbalance and discriminatory behavior of the model evaluated over the historical stream. We compare our MQ-OFL method with state-of-art studies on real-world datasets and present

comparative insights on the performance.

Our final contribution focuses on explainability and interpretability in fairness-aware online learning. This research is guided by the concerns raised due to the black-box nature of models, concealing internal logic from users. This lack of transparency poses practical and ethical challenges, particularly when these algorithms make decisions in finance, healthcare, and marketing domains. These systems may introduce biases and prejudices during the learning phase by utilizing complex machine learning algorithms and sensitive data. Consequently, decision models trained on such data may make unfair decisions and it is important to realize such issues before deploying the models. To address this issue, we introduce techniques for interpreting the outcomes of fairness-aware online learning. Through a case study predicting income based on features such as ethnicity, biological sex, age, and education level, we demonstrate how our fairness-aware learning process (MQ-OFL) maintains a balance between accuracy and discrimination trade-off using global and local surrogate models.

Acknowledgements

In embarking on this academic journey, I am compelled to extend my heartfelt gratitude to those whose continuous support and guidance have illuminated my path to this thesis's completion.

First and foremost, I express my sincere gratitude to my esteemed supervisor, Professor Herna Viktor. Your invaluable mentorship, expertise, and unending encouragement have not only shaped my academic journey but have also profoundly impacted my personal growth. I am genuinely privileged to have had the opportunity to work under your guidance, and I am profoundly thankful for the knowledge and skills I have acquired through this enriching experience.

I also want to thank my good friends who stood by me during this journey. Foremost among them is Rawad Mcheimech, for his unconditional support, patience, and consistent belief in me throughout this journey. Masoud Hani, your critical thinking and criticisms have been a huge help throughout this process. Neda Arabgol, Bamdad Mousavi, and Ali Simaei, your friendship and support have meant so much to me. I really appreciate everything you have done.

To all of you, thank you for being my source of strength. Your love, support, and resolute belief in me have played an essential role in my academic success. I am truly blessed to have each of you in my life.

Finally, I would like to acknowledge the financial support from the National Science and Engineering Research Council (NSERC) and the University of Ottawa, which has been instrumental in enabling me to pursue my research goals.

Dedication

I dedicate this thesis to the Almighty God, the source of all wisdom and strength, and to my parents, Maghsoud and Mehrinaz, for their unwavering love, encouragement, and sacrifices. Their constant support and belief in my abilities have been instrumental in shaping my academic path. To my sister, thank you for being my eternal source of motivation, and your presence brings immense joy to our family. To my supervisor, whose impact on both academic and personal growth has been significant.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Traditional machine learning algorithms generally consider data in static environments and have assumptions of complete data availability [100]. However, data in real-world environments may have a dynamic behavior, and all data may not be available at the time of learning [71]. Indeed, the concept of data streams is defined as high-speed generated instances of data, gathered from sources such as sensor networks [24], that challenge our computational time, storage space and process limits. It follows that streaming data offer an important source of information that enables us to take crucial decisions in real-time [67].

Data streams usually are affected by the phenomenon of concept drift [24], which means that the underlying distribution of the data changes over the so-called evolving stream. Due to the challenging behavior of data streams, the static machine learning algorithms are not sufficient [37]. This has led researchers to develop techniques that aim to provide timely responses to incoming data. That is, the online learning algorithms that are able to detect and handle concept drift during the stream are an active area of research. These solutions are relevant in various real-world applications [119,160], such as monitoring in bio-medicine, industrial processes, credit card transactions, network analysis, financial data prediction, and traffic control [163]. Online machine learning algorithms are suitable in such scenarios, since they continuously incorporate information into their model, and basically aim for minimal-processing time and space. Due to their ability of continuous large-scale and real-time processing and building adaptive models to changes over streams, they recently attracted much attention in the research community [70, 100].

1.1 Motivations

In many real-world applications, such as medical diagnosis [46], network intrusion detection [7] and spam filtering [109], the distribution between the classes of examples might not be well balanced [129]. Assume a scenario where we want to automatically detect whether an incoming email is a spam or not. Here, we will typically have many more negative examples (i.e., non-spam) than positive examples (i.e., spam). However, most traditional machine learning classification algorithms do not adequately address such skewed distributions. That is, they have difficulty learning from imbalanced class problems since their objective is to maximize the overall accuracy, thus implicitly the class with fewer instances could be ignored or misclassified. In the literature, class imbalance refers to the disproportion among classes, when one class, called a minority class, has significantly fewer examples than another class, called a majority class [100]. This problem can cause learning bias toward the majority class and poor generalization of data, as well as greatly reduce the performance of the minority class. The class imbalance problem is not limited to binary classification and in several real applications, algorithms must deal with more than two classes. In this case, we must extend the imbalanced classification problem to the multi-class scenario and as the number of classes increases, so does the challenges of classifying the instances accurately. While online imbalanced learning on binary classification has been studied over the years, online multi-class imbalanced classification has received limited attention [97, 101]. However, most of the algorithms for binary imbalanced data cannot be directly extended to the multi-class case study, mainly due to the multi-majority and multi-minority examples. For this reason, in our study, we focus on finding specific solutions, rather than adapting standard ones to the multi-class context.

Now that the importance of class imbalance in classification has been acknowledged, a variety of techniques have been developed to address the problem [33]. These approaches fall into the following three categories [59] as data level, algorithm level and hybrid approaches. One of the effective ways to encounter a multi-class imbalance data stream when considering data level solution is employing one of the sampling methods. This family of approaches can also be broken down into three groups: increasing the number of instances in the minority classes (over-sampling), decreasing the number of instances in the majority classes (under-sampling), or a combination of both (hybrid-sampling). The state-of-art multi-class imbalance approaches that are utilizing the sampling methods such as Multi-class Over-sampling Online Bagging (MOOB) [157] and Multi-class Under-sampling Online Bagging (MUOB) directly duplicate instances for minority classes and remove samples from majority classes. However, these two methods do not encounter concept drifts. In another study, SCUT-DS [120] also employed a window-based over-sampling and cluster-based

under-sampling method. Further, data with a severe imbalance ratio will suffer from overfitting the model by creating excessive synthetic instances. In addition, [40, 41] introduced ensemble-based approaches utilizing random feature subsets and resampling to address imbalances. However, current solutions do have some limitations. That is, additional research is required to explore classifier-agnostic approaches that can effectively utilize alternative evaluation metrics and resampling parameters to deal with the issues posed by imbalanced datasets. Furthermore, it is essential to develop class-based drift detection methods to mitigate performance degradation. This thesis addresses these shortcomings.

Moreover, other than dealing with the class imbalance and concept drifts, the potentially unconscious and harmful use of the models constructed against such evolving data is an area of concern [114]. That is, models constructed by online learning algorithms are currently widely being used to assist or even replace human-based decision-making systems, including the allocation of healthcare resources, insurance rate assessment, court judgments, or issuing of mortgage plans. There is growing concern that these algorithms fail to explicitly address issues around fairness and bias. Such failure may lead to unfair discrimination against certain individuals or groups [167]. Research into fairness-aware learning is an active and growing area of research [81, 114, 121, 153]. However, only a few recent works investigate the problem of fairness-aware learning in evolving environments [127]. FAHT [166] and FEAT [165], are based on Hoeffding Tree (HT) [52] algorithm which considers fairness in their decisions. (Note that HTs [52] are incremental decision trees for data stream classification that use Hoeffding's bound [76] as splitting criteria to commence online learning.) These two methods extend new splitting criteria for HT induction that includes the fairness gain in each instance, while it is growing. However, neither technique adequately handle the class imbalance issue. In addition, FABBOO [82] represents another study that extends the online boosting approach which changes the distribution of training data as it arrives. This approach associates the model with stream imbalance, concept drift and discriminatory behavior of the model over data. The final decision is based on long-term consideration of class imbalance and fairness, which is different than other methods. However, it's important to note that the algorithms previously mentioned primarily address discrimination based on individual features and may not comprehensively consider challenges related to numerous sensitive groups. These methods tend to focus on specific forms of bias and discrimination, often associated with features like race or gender. Gerrymandering, on the other hand, is a complex issue that requires specialized approaches for effective resolution.

Finally, while machine learning algorithms have demonstrated great success in analyzing complex patterns [168] and are used in a wide range of applications, including finance, justice, voting, and healthcare, they can be challenging to comprehend due to their lack of intuitiveness, interpretability and explanation of model predictions. The complexity

and lack of transparency in the process used to produce the final output often make machine learning algorithms difficult to interpret [115]. The absence of explanations for the decisions made by models represents a significant shortcoming in critical decision-making processes, particularly in discrimination-aware areas. Therefore, it is crucial that machine learning models possess two primary characteristics: interpretability and explainability. This concern has led to the emergence of explainable machine learning, which advocates for machine learning algorithms that can display their internal processes and explain how they arrived at their decisions. Although studies have been conducted on the interpretability and explainability of machine learning models [1], only a few of them have focused on fairness-aware algorithms [124].

In short, current algorithms only partially handle the issues of concept drift, class imbalance and eliminating discrimination when learning from an evolving data stream. Additionally, there are few studies that refer to interpretability and explainability of these concepts. This thesis introduces novel solutions to address these limitations.

1.2 Research Questions

The following research questions are addressed in this thesis:

- What are the challenges and considerations when extending binary imbalanced data algorithms to the multi-class scenario, particularly in the presence of multiple majority and minority classes?
- What is the effective approach to address the problem of class imbalance in multi-class imbalanced data streams, specifically benefiting from resampling methods, while considering the challenges of concept drift?
- How can fairness and bias issues be explicitly addressed in the construction of models using online learning algorithms, particularly in evolving environments?
- What are the limitations of existing methods in adequately handling class imbalance and more sensitive features challenges, such as gerrymandering?
- What are the specialized approaches required to effectively address complex issues like gerrymandering and ensure fair representation in decision-making systems?
- How can the complexity and lack of transparency in machine learning algorithms be addressed to enhance the interpretability and explainability of model predictions, particularly in discrimination-aware areas?

- What are the specific challenges and considerations when applying interpretability and explainability techniques to fairness-aware algorithms, and how can these techniques help address fairness-related concerns in decision-making processes?

1.3 Objectives

Recall that real-world applications are a source of data streams (including evolving concepts and skewed distributions) which makes online learning difficult [36]. This is especially challenging when the data has multiple class labels. Therefore, our first objective in this thesis is to develop accurate and timely models during online multi-class imbalanced learning from evolving streams. Our purpose is to design and implement algorithms that can maintain an accurate and balanced performance over all classes in the stream, even in the case where the data distributions are highly skewed. That is, we introduce novel techniques for dealing with online imbalanced learning that considers a balanced evaluation of instances of each class. Hence, our solution yields better performances against the minority classes, which are often the most important classes to be learned. To this end, we introduce the Online Learning from Imbalanced Multi-Class Streams through Dynamic Sampling (DynaQ) algorithm, as detailed in Chapter 3. DynaQ is a queue-based learning technique that dynamically balances the training instances based on the most recent examples. Our approach utilizes a batch-based resampling method that creates an instance queue for each class to balance the number of instances. We maintain a queue threshold and remove older samples during training. Additionally, we dynamically oversample minority classes based on one of four rate parameters: recall, F1-score, κ_m, and Euclidean distance. Our learning algorithm consists of an ensemble approach that uses sliding windows and a soft voting schema while incorporating a drift detection mechanism.

Our second contribution focuses on addressing the objective of **fairness-aware online learning in imbalanced streams** through the design and implementation of our Multi-Sensitive Queue-based Online Fair Learning (MQ-OFL) approach. As stated earlier, online machine learning methods have an impact on many aspects of human life. However, these algorithms can provide decisions based on sensitive features such as gender or race and therefore can lead to discrimination. Sensitive features are referring to features that may be given special consideration (such as legal, ethical, social, or personal reasons) based on the application [153]. For this reason, it is crucial to develop algorithms that are not only accurate but also objective and fair. Besides this, since the data streams are not stationary, they have the problem of class distribution skewness and concept drift. Consequently, as stated above, our second objective is to develop a discrimination-free classification algo-

rithm for evolving streams even when the data is imbalanced. That is, we aim to balance both the data distribution of each target class and the defined sensitive features. However, working with only one sensitive attribute could lead to overlapping protected groups (defined based on the sensitive attribute(s)) which would still be under discriminated decisions [88, 89]. In order to overcome this issue, our Multi-Sensitive Queue-based Online Fair Learning (MQ-OFL) method considers multiple sensitive features.

Furthermore, as the field of online supervised learning from data streams continues to expand, one of the major concerns is the lack of transparency of the majority of these algorithms. Many of these systems are black-box models, which means they hide their internal logic from the user. As a result, decision models learning from this data may inherit such biases and make unfair decisions. Therefore, it is crucial to develop techniques to interpret online learning results to ensure fairness and transparency. In this context, this thesis introduces state-of-the-art methods to **explain and interpret the results of a fairness-aware online learning algorithm**. Our approach is illustrated through a case study that predicts a person's income based on features such as ethnicity, biological sex, age, and education level. The results show how the fairness-aware learning process maintains an accuracy and discrimination trade-off while interpreting the learning results using global and local surrogate models.

1.4 Contributions

We summarize the contributions of this thesis below:

- In this research, we introduce the DynaQ approach, which creates models from data in a multi-class imbalanced setting. Our multi-class learning algorithm maintains separate queues for each class, thus utilizing training based on the current data. In our work, we dynamically oversample minority classes based on one of four rate parameters: recall, F1-score, κ_m, and Euclidean distance.

- In our study, we adopt a technique that does not assume a fixed frequency of classes, allowing for the possibility of minority classes becoming majority classes, and vice versa. This implies that each class has the potential to receive a varying number of instances in the data stream, and we ensure a balanced treatment of all classes. To handle imbalanced ratios, we employ maintain classes in separate queues.

- We introduce a drift detection mechanism able to separately detect changes in the individual classes, while simultaneously handling multiple class drifts. The novelty

of this approach is that, for each class, we maintain a queue of instances and flag for drifts when a confidence threshold is reached. Thus, drifts in minority classes with fewer samples are not ignored.

- Furthermore, our algorithm combines batch-incremental and instance-incremental online ensemble learning. That is, initially a batch of data with all classes is presented to the learner and it subsequently proceeds to update the model with new instances as they arrive.

- We point out a new perspective of fairness-aware learning in imbalanced data streams with more than one sensitive attribute. Then, we propose a discrimination-aware pre-processing method to address the trade-off between fairness and accuracy. Additionally, we introduce an approach to pre-process multi-sensitive features that satisfies fairness constraints.

- We investigate explainability and interpretability of fairness-aware methods and employ state-of-the-art solutions to interpret the results thereof. This study shows how the MQ-OFL algorithm addresses the challenges posed by learning from imbalanced datasets containing multiple sensitive features. This contribution highlights the need for interpretable and transparent machine learning algorithms, especially in domains where fairness and ethical considerations are paramount. It emphasizes the importance of developing approaches to interpret the results of online learning and ensure that the resulting models are transparent, interpretable, and fair.

1.5 Thesis Organization

This thesis is organized as follows: In Chapter 2, we detail and review the background in relevant areas, namely data streams, imbalanced learning, and fairness-aware learning. Chapter 3 introduces our Online Learning from Imbalanced Multi-Class Streams through Dynamic Sampling (DynaQ) for dealing with multi-class imbalanced data. We illustrate the results of our proposed algorithm when compared to the state-of-the-art approaches. In Chapter 4, we explore online fairness learning by introducing our Online Fair Queue-based methodology (MQ-OFL). The technique employs queues to allocate, to each sensitive attribute and its associated class label, a balanced and fair presentation during online learning. The chapter includes our detailed experimental comparison with the state-of-the-art approaches. In Chapter 5, we explore the relationship between explainability and fairness-aware online learning algorithms. Our study emphasizes the importance of explaining fairness as a crucial component in the responsible utilization of online machine

learning algorithms and trustworthy decision-making. Furthermore, our exploration highlights the interplay between explainability and fairness-aware models including multiple sensitive features, gerrymandering interpretation, managing the trade-off between fairness and model accuracy, and incorporating incremental online learning concepts. Finally, in Chapter 6, we conclude the thesis and detail our future work.

Chapter 2

Background

This chapter provides a review of related work in the fields of online machine learning, class imbalance, fairness-aware learning, and explainability and interpretability of machine learning algorithms. Section 2.1 gives an overview of online learning and related topics, while Section 2.2 discusses imbalanced environments and methods for dealing with imbalanced datasets, including multi-class imbalanced data. Background concepts and methods related to ensemble learning are presented in Section 2.3. The chapter then moves on to Section 2.4, which introduces fairness-aware learning concepts and algorithms. Section 2.5 details the explainability and interpretability of machine learning models. Finally, Section 2.6 summarizes the background information covered in this chapter.

2.1 Online Learning

Recall that traditional machine learning is performed offline using the batch learning method. In batch learning, data are accumulated over a period of time. The machine learning model is then periodically trained with this accumulated data in batches [144]. Traditional machine learning is the direct opposite of online learning because the model is unable to learn incrementally from a stream of live data. In batch learning [131], the machine learning algorithm updates its parameters only after consuming batches of new data. The fact that models are trained with large batches of accumulated data means that more time and resources such as CPU, memory space, and disk input/output are needed. It also takes longer to push models to production because this can only be done at certain intervals based on the performance of the model after being trained with new data. If a

model trained using batch learning needs to learn about new data, it must be retrained using the new data set [24]. A data stream environment has different requirements from the traditional offline learning setting.

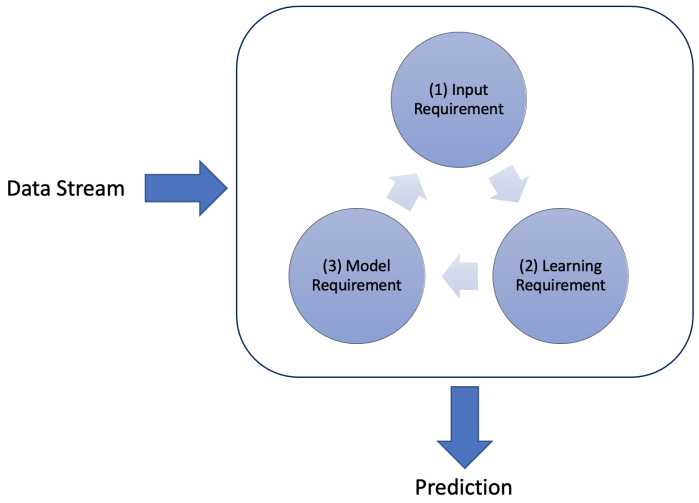


Figure 2.1: The data stream classification cycle

Figure 2.1 illustrates the flow of a data stream classification algorithm: first, the algorithm fetches the next available example from the stream. Then, the algorithm processes the example, updating its learning structures. It does this within the memory bounds defined in the input requirements and as quickly as possible. Finally, the algorithm is ready to accept the next sample. It is also able to predict the class of unseen examples in the prediction part. Evaluating data streams in real-time is a challenging task. Two main approaches arise [71]:

Holdout : The traditional holdout partitions data into training and test sets that requires all data to be available, but in data streams instances arrive one-by-one over time. That is holdout idea is applied in a different way for an online data stream. The online method holdouts a separate batch of data for testing purposes. After the classification model is trained with N number of instances, it is evaluated by that separate batch. The point is the holdout batch has not been used in the training phase. The holdout batch needs to be properly large enough to represent the complexity of the target concept. However, if the holdout set is very large, it would slow down the learning process. Therefore, the size of the holdout set is a critical parameter [24, 69].

Test-then-Train (Prequential) : Unlike the holdout method, here, each instance is used both for training and testing purposes. However, each individual example is used to test the model before it is used for training, and accuracy is incrementally updated while instances are one-by-one processed. This procedure is also known as interleaved test-and-train [25]. The idea of this procedure is to assure that a classification model is always being tested on unseen instances [69].

The next section describes the concept drift in online learning.

2.1.1 Concept Drift

The term concept drift refers to the problem of learning in non-stationary or evolving streams [24] where the distribution of data changes during learning. That is, concept drift is a phenomenon in which the statistical properties of a target domain change over time in an unforeseen way [107]. It was first discussed by [144], who pointed out that noise data may turn to non-noisy information at different times. If the concept drift occurs, the induced pattern of past data may not be relevant to the new data, leading to poor predictions and decision outcomes. Considering D_j as data distribution and $S = S_1, S_2, ..., S_n$ as data samples, concept drift can be formally defined as any scenario where the joint probability $p(x, c_i) = p(c_i)p(x|c_i)$ changes, in which x represents the input features and c_i represents the class label [106]. Generally, concept drifts can be categorized into four (4) groups based on the severity and speed of changes: gradual, incremental, sudden, and recurrent [106].

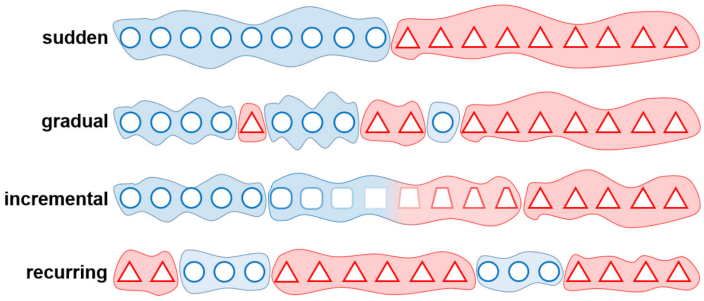


Figure 2.2: Concept drift types [106]

Abrupt/sudden concept drift is drift characterized by a rapid change of the distribution of data [149], where $D_j \neq D_(j + 1)$. A new concept occurs in a short period

of time (e.g. at the beginning of COVID-19 in March 2020, stock prices suddenly changed). To respond to these changes, models need to adapt by forgetting the previous knowledge and learning the new concept.

Gradual concept drift happens slowly, and a new concept gradually replaces an old one; the two concepts go back and forth until the new concept stabilizes. One example is the decline in the number of thriving oil companies and the increase in new tech companies. This drift can be considered as a transition phase where examples in S_{j+1} are generated by a combination of both D_j and D_{j+1} with their varying proportions until the data shift to the new distribution completely [126].

Incremental concept drift has a slower ratio of changes over longer time steps, where the difference between D_j and D_{j+1} in a few time steps is not statistically significant. However, the accumulated changes over a long period of time in data distribution are visible; for example, a stock price gradually and steadily increases [24, 149].

Recurrent/reoccurring concept drift : An old concept may reoccur after some time once or periodically (e.g. changes in winter clothes shopping volumes in the summer versus winter). In this kind of concept drift, a state from a previous distribution may reappear again $D_{j+1} = D_{jm}$, where m is the m_th previous iteration [24, 149].

Concept drift problems exist in many real-world situations and have been recognized as the root cause of decreased effectiveness in, for example, data-based sensor systems and decision support systems [129]. The question of how to provide more reliable predictions and decision facilities has become a crucial issue in evolving and big data environments. Concept drift detectors are used to detect these changes in the distribution of the data [106].

2.1.2 Drift Detection

Drift detection refers to the techniques and algorithms that detect and quantify concept drift via identifying change points or change time intervals. The drift detection methods can be divided into three categories [129]: (a) instance selection or window-based, (b) weight-based, and (c) ensembles of classifiers. Instance selection or window-based approaches [31] employ a related set of previous data to train a classifier. They select instances inside a fixed or dynamic sliding window. The assumption made by these methods is that older samples of data are incompatible with the new data classification, so the model must forget old instances that no longer apply to the incoming data. That is, the training phase uses the last batch of data with the last training instances. The fixed window approach is the

simplest method, and the window size is usually determined by the user. However, we need information on the time scale of the change to determine the window size. In addition, the user might decide on a trade-off between a small window size (reflects the current distribution with fast adaptivity) or a large window size (having more instances in periods of stability with no concept drift may increase accuracy and support better generalization) [71]. The adaptive window technique adjusts the window size to the duration of the drift by maintaining the examples until the concept drift. Other methods select a training set of examples maintained in the memory [129]. A weight-based approach assigns weights to instances based on their importance and deletes the outdated training data. It considers old instances during the time irrelevant and gives more weight to new instances. To this end, choosing an adaptable classifier that continuously updates the instance weights is a necessary step [31]. The ensemble of classifiers combines outputs from different learners to decide on a final classification. The classifiers are evaluated based on different parameters, such as performance. If performance decreases, new classifiers replace the aged and poor-performing classifiers in the ensemble. Commonly, outputs from learners are used in a voting system to classify the instance. Many drift detection methods have been introduced in the literature, but we only described the ones commonly used because of their effectiveness in an online setting. Some of these methods are utilized in this study.

Drift Detection Method (DDM) is one of the most-referenced concept drift detection algorithms [68]. It is a drift detector that works with probability distribution using the online error rate. It defines two levels: the warning and the drift level. Drift is said to have occurred when the error rate reaches the defined warning level at time t_w and the drift level at time t_d. When this occurs, the method resets the learner, starts the training process with data saved between t_w and t_d, and replaces the old learner with the new learner for further prediction tasks. According to the authors, the DDM can operate with online and incremental methods.

Early drift detection method (EDDM) [67] is an extension of the DDM method. EDDM is a drift detector that calculates the average distance between misclassifications rather than the error classifications used in DDM. EDDM has two thresholds: the warning level—instances are kept after this level, and the drift level—the method considers the concept has drifted and builds a new model with data from the warning level. EDDM starts to search for a concept drift after a minimum of 30 errors have occurred. Otherwise, if the similarity of distribution after the warning threshold increases over the warning threshold, the stored examples are removed, and the method returns to the "in-control" level. EDDM improves the drift calculation compared to DDM by using the distance between two correct classifications to improve sensitivity.

This method improves the detection in the presence of gradual concept drift. At the same time, it has a good performance with abrupt concept drift.

Adaptive windowing (ADWIN) [22] is a window-based drift detection algorithm. This method maintains a window of variable size containing bits or real numbers. The method automatically expands the window when no change is apparent and shrinks it when data changes. ADWIN does not require users to define the size of the compared windows in advance; it only needs to specify the total size N of a proper size window W. ADWIN divides the original window into two separate sub-windows w_0 and w_1, with optimal sizes of n_0 and n_1, and checks whether w_0 and w_1 are generated from the same distribution; if not, then the drift is detected at that point, and all data before that time are discarded.

HDDM: The Hoeffding drift detection methods ($HDDM_{A-test}$ and $HDDM_{W-test}$) [136] are based on Heoffding's inequality, which is a probability concept that provides the maximum value difference between estimated amount and the expected amount of an instance [76]. $HDDM_{Atest}$ compares the moving averages of windows to detect drifts, while $HDDM_{Wtest}$ uses the weighted forgetting mechanism to weigh the moving averages. Then, weighted moving averages are compared to detect the drift. For both cases, Hoeffding's inequality [76] is used to set an upper bound to the level of difference between averages. These methods are ideal for detecting abrupt and gradual drifts, respectively.

The Fast Hoeffding Drift Detection Method (FHDDM) [126] detects the drift points using a sliding window and Hoeffding's inequality. When a significant difference between the maximum probability of correct predictions and the most recent probability of correct predictions is observed, FHDDM will detect the drift. The significant difference is defined by the Hoeffding inequality as an upper bound. FHDDM is a very fast drift detector. However, unlike the ADWIN method, the sliding-window sizes need to be specified as a hyperparameter.

2.2 Class Imbalance

The definition of class imbalance technically refers to a data set with an unequal class distribution [92]. However, a data set is considered to be imbalanced when there is a significant disproportion among the number of instances of each class of the data. This poses a difficulty for learning algorithms, as they will be biased towards the majority class [16].

Another important issue in an imbalanced data set is that usually, the underrepresented class is the problem's class of interest from the application point of view. Furthermore, this issue is known to hinder the performance of classifiers due to their accuracy-oriented design, which usually results in the minority class being overlooked. Applications that are known to suffer from this problem are fault diagnosis, anomaly detection, medical diagnosis, e-mail filtering, face recognition, and detection of oil spills [163]. Most of the imbalanced classification literature has been devoted to binary classification problems [43], where one class significantly outnumbers the other (which is, therefore, underrepresented). Some of the multi-class data sets also have skewed-class distributions [100]. We will discuss the related methods in Section 2.2.2. In two-class or binary problems, the minority (underrepresented) class is usually referred to as the positive class, whereas the majority class is considered to be the negative one. Many techniques have been developed to achieve the objective of correctly distinguishing the minority class. These techniques can be categorized into four main groups according to how they deal with the problem.

Data Level: These kinds of approaches, which are also called external levels, aim at rebalancing the class distribution by resampling the data space, in which the training instances are modified in such a way as to produce a more balanced class distribution that allows classifiers to perform in a similar manner to standard classification [59]. This way, the effect caused by the imbalance is decreased with a preprocessing step. Some of these techniques generate new objects for minority groups (over-sampling) and remove examples from majority groups (under-sampling) or combine both until the data set is relatively balanced [93, 146]. This group of methods also consists of solutions for cleaning overlapping instances and removing noisy samples that may negatively affect the learning process [144].

Algorithmic Level: These methods refer to internal modifications by adapting existing classifier learning algorithms. The goal of these algorithms is to provide better accuracy for the minority class, including one-class learning and threshold methods [59]. The most common algorithm-level method, cost-sensitive learning [92], considers higher costs for the misclassification of examples of the minority class with respect to the majority class and, therefore, tries to minimize higher cost errors [71].

Ensemble-based: This family, known as multiple classifier systems, has become a major category of approaches to handling class imbalance. These methods usually consist of a combination of an ensemble learning algorithm [95] and one of the techniques above. An ensemble-based algorithm can be adapted to emphasize the minority class by integrating different resampling techniques [156] or by making base classifiers cost-sensitive. Ensembles are designed to improve the performance of a single classifier

by training several different classifiers and combining their outputs to make the final decision.

2.2.1 Sampling Methods

The first mechanism to address the problem of imbalanced learning is the use of sampling methods at the data level [156]. They consist of modifying a set of imbalanced data using different procedures to provide a balanced or fairly distributed data set to the learning process. In the literature, many studies have employed rebalancing data approaches and have shown that resampling significantly improves the overall performance of the classification compared to a non-preprocessed data set [154]. Resampling techniques can be categorized into three groups: **Under-sampling**, which creates a subset of the original data set by eliminating examples belonging to the majority class with the objective of equalizing the number of examples of each class; **Over-sampling**, which aims to duplicate some instances or create new synthetic instances from existing ones; **Hybrid-sampling**, which combines both sampling approaches. There are many variants of sampling methods introduced for dealing with imbalanced classes [93, 146]. In this section, we only cover the sampling methods that are common or are used in this study.

Random Under-sampling (RUS): This process randomly removes instances from the majority classes until all the classes reach the same number of instances as in the minority class. As a consequence, RUS can discard potentially useful data. Using this approach with ensemble learning (like in RUSBoost [145] and RUSBagging [13]) may overcome this issue because instances that are absent in one iteration may be present during another.

Random Over-sampling (ROS): This method [44] aims to balance the class distribution by randomly duplicating minority class instances. Unlike the RUS, data samples in the minority classes are duplicated/generated until all the classes have the same number of instances as the majority class. This will lead to over-fitting the model with synthetic data.

Synthetic minority over-sampling technique (SMOTE): This is a widely used over-sampling method that generates new minority class samples based on the similarities level between original minority class instances in the feature space [45]. The creation of new minority class instances leads to a reduction in the class imbalance degree compared to the original class ratio. Also, SMOTE elevates the overfitting

concern by the learner in over-sampling techniques because it synthetically creates new instances rather than duplicates existing instances.

SCUT refers to a multi-class imbalanced data sets classification approach using hybrid sampling techniques that combine SMOTE-based over-sampling of the minority classes with cluster-based under-sampling of the majority classes. This method addresses within-class and between-class imbalance ratios in multi-class imbalanced static data without decomposing the data set [4]. The extension of this algorithm that works on the data stream is SCUT-DS [120]. SCUT-DS is a window-based method that balances the incoming instances from all classes while the stream evolves and, like SCUT, combines synthetic minority over-sampling (SMOTE) and cluster-based under-sampling.

The described sampling methods are utilized in the literature and this study because of their effectiveness and performance. In the next section, we detail some of the methods developed for multi-class imbalanced data.

2.2.2 Multi-class Imbalance

Multi-class classification problems (referring to data with more than two classes) are common in real-world scenarios. Some real-world applications are microarray research [46], protein classification, medical diagnosis, activity recognition, target detection, and video mining [163]. All these problems have one thing in common: the distribution of examples among the multiple classes is not homogeneous. When facing multi-majority and multi-minority classes, it is not straightforward to know which class is more important, as was done in the binary case study. Additionally, most of the techniques proposed for binary imbalanced classification are not directly applicable to multiple classes. Some techniques apply traditional binary-class imbalanced approaches to multi-class problems, such as decomposition-based strategies, SMOTE, and SCUT-DS. However, the following methods are designed for online multi-class imbalance problems.

OVA: In the one-versus-all approach [61], the authors decompose a multi-class data set into several binary class problems and subsequently train a single classifier for each class, considering the current class as the minority one and the remaining classes as a majority one. The downside of such binarization techniques is that the classes cannot be studied separately.

OVO: One-versus-one [61] is a decomposition-based strategy that first selects a subset from the original data that only contains the instances for each pair of classes, then trains a binary classifier for each pair of classes. This method ignores examples that do not belong to the same pair classes.

MOOB and MUOB: In multi-class over-sampling online bagging (MOOB) [157], the minimum (w_{min}) and maximum (w_{max}) class sizes among all classes is calculated at each time step. Subsequently, the algorithm sets $\lambda = \frac{w_{\max}}{w_j^t}\hat{y}_t$, where w_j^t is the size of class c_k at time t. This approach thus ensures that minority classes have a larger sampling rate. The multi-class under-sampling online boosting (MUOB) algorithm inverses this idea, where λ controls the under-sampling rate. Thus, in MOOB, over-sampling is used to increase the chance of learning minority-class examples based on the occurrence probability of examples belonging to each class, while in MUOB, under-sampling is used to reduce the chance of learning majority-class examples. These algorithms can process multi-classes directly without using class decomposition.

SOUP: Similarity over-sampling and under-sampling preprocessing (SOUP) [84] combines over-sampling with under-sampling to achieve a balanced class distribution in the training set. SOUP resamples all classes to the number of the mean of the biggest minority and the smallest majority class sizes. That is, the algorithm oversamples all the minority classes and undersamples all the majority classes. Additionally, both under- and over-sampling are performed with a weight-based selection of instances to be resampled. The key idea of this method is to assign a weight to each instance based on its similarity score to each class.

In this section, we focused on class imbalanced learning. Next, we turn our attention to ensemble learning.

2.3 Ensemble Learning

Ensemble methods, also known as multiple classifiers or committees, are a promising avenue of research as they are a collection of individual classifiers that can be combined to make predictions for new instances [95]. This approach has been shown to be effective in improving accuracy and simplifying complex learning problems. The motivation for using ensemble methods is based on the seminal work of Robi Polikar [128], which states that no single classifier is appropriate for all tasks. Instead, a pool of classifiers can be used to

solve a problem. The selections of classifiers for the ensemble are crucial, as they should include diverse and accurate models that complement each other. In ensemble learning, diversity refers to the degree of difference or dissimilarity between the individual models in the ensemble [95]. Diverse models tend to make different errors and can complement each other to improve the overall performance of the system. To evaluate the diversity of an ensemble in ensemble learning, various measures can be used. One type of measure is disagreement-based [53], which quantifies the level of disagreement among individual models in the ensemble. Examples of disagreement-based measures are voting entropy, which calculates the uncertainty in the ensemble's prediction based on the distribution of votes across the models. Another type of measure is correlation-based, which measures the degree of correlation or similarity between the individual models in the ensemble. Examples of correlation-based measures include the correlation coefficient, which measures the linear correlation between the predictions of the individual models, and mutual information, which measures the amount of information shared between the predictions of the individual models. The choice of diversity measure depends on the specific problem and the types of models being used in the ensemble. The best choice may vary depending on the data and the ensemble configuration.

2.3.1 Ensemble Combination Rules

In ensemble learning, combination rules refer to the methods used to combine the outputs of individual models in an ensemble. Ensemble learning involves combining multiple machine learning models to improve the overall performance of the system. The combination rule specifies how the outputs of the individual models should be combined to produce a final prediction [143]. Next we list some common ensemble combination strategies:

Majority voting [51] is a simple and popular ensemble learning voting strategy. In binary classification problems, if the number of base classifiers predicting class 1 is greater than the number of classifiers predicting class 0, the final prediction is class 1. Mathematically, the prediction can be defined as follows:

$$Final\_prediction = \arg\max_i count(i) \tag{2.1}$$

Where i is the class label, and $count(i)$ is the number of base classifiers predicting class i.

Weighted voting [57] assigns weights to each base classifier and combines their predictions based on the weights. The weights can be based on the performance of the

base classifiers on the training data or can be manually assigned. Mathematically, the prediction can be defined as follows:

$$Final\_prediction = \arg\max_i \sum_j w_j * p_{ij} \tag{2.2}$$

Where i is the class label, j is the index of the base classifier, w_j is the weight assigned to the j-th base classifier, and p_{ij} is the probability of class i predicted by the j-th base classifier.

Soft voting [50] refers to a combination rule that considers the probabilities or confidence levels of individual models to make the final prediction. Soft voting calculates the average of the predicted probabilities for each class by the individual models and then selects the class with the highest average probability as the final prediction for a given input. Additionally, it can be more robust to noise and outliers in the individual model predictions, as it considers the confidence levels of each model rather than solely their binary decisions. In contrast, hard voting makes the final prediction based on a simple majority vote of individual model predictions, without accounting for their confidence levels. Given an input example x, let M be the set of individual models in the ensemble, and let K be the number of classes. For each class $k = 1, ..., K$, the soft voting rule calculates the probability of class k as:

$$P(y = k|x) = \frac{1}{|M|} \sum_{m \in M} P_m(y = k|x) \tag{2.3}$$

Where $P_m(y = k|x)$ is the predicted probability of class k by model m, and $|M|$ is the total number of models in the ensemble. The final prediction for the input example x is then made by selecting the class with the highest probability:

$$\hat{y} = \arg\max_{k=1,...,K} P(y = k|x) \tag{2.4}$$

In other words, the soft voting rule combines the predicted probabilities of the individual models for each class and selects the class with the highest average probability as the final prediction.

Stacking [159] involves training a meta-classifier on the outputs of the base classifiers. The base classifiers make their predictions, and their outputs are then used as features to train a meta-classifier that makes the final prediction. Mathematically, the

prediction can be defined as follows:

$$Final\_prediction = meta\_classifier([p_1, p_2, ..., p_n]) \tag{2.5}$$

Where p_i is the probability vector predicted by the i-th base classifier, and the meta-classifier is a function that takes the output of the base classifiers as input and makes the final prediction.

Each ensemble learning voting strategy has its strengths and weaknesses, and the choice of strategy depends on the specific problem and data at hand.

2.3.2 Ensemble Learning Methods

There are several types of ensemble-based learning methods. We describe some of them in the following categories:

Bagging and Boosting are methods that involve building multiple models or classifiers using different subsets of the training data, and then combining their predictions. In bagging [35], each model has an equal weight in the final prediction, while in boosting, models are weighted based on their performance. The final prediction in bagging is made by combining the predictions of all the models. Mathematically, the prediction can be defined as follows:

$$Bagging\_Final\_prediction = \arg\max_i \sum_j 1/M * p_{ij} \tag{2.6}$$

Where i is the class label, j is the index of the base classifier, M is the number of base classifiers, and p_ij is the probability of class i predicted by the j-th base classifier. and in boosting [64] the final prediction is made by combining the predictions of all the models, weighted by their performance.

$$Boosting\_Final\_prediction = \arg\max_i \sum_j \alpha_j * p_{ij} \tag{2.7}$$

Where i is the class label, j is the index of the base classifier, α_j is the weight assigned to the jth base classifier based on its performance, and p_{ij} is the probability of class i predicted by the jth base classifier.

Random forests: This is a type of bagging ensemble based on decision trees [75], where multiple decision trees are built on different random subsets of the features and data, and the final prediction is made by combining the predictions of all the trees.

Gradient boosting: [65] uses gradient descent to optimize a loss function, with each model building on the errors of the previous models.

There are many other types of ensemble learning methods, and the choice of method depends on the specific problem and the data we want to learn.

2.3.3 Ensemble Learning from Data Streams

Ensemble learning is an effective way to create data stream classifiers that can adapt to changes in the data distribution. This adaptation can be achieved by modifying the ensemble line-up, such as adding recently trained classifiers and removing outdated ones, or by retraining the ensemble components [73, 95]. There are two main types of ensemble learning, namely chunk-based and online. Chunk-based learning involves dividing the training data into batches and building separate models for each batch. The models are trained on a fixed-size batch, and their predictions are combined to make the final prediction. Chunk-based learning is useful when the entire dataset cannot fit into memory and when the data arrives in chunks [132].

Online ensemble learning, on the other hand, is designed to learn and adapt to new data in real-time. It involves building an ensemble of models that can update and make predictions as new data points arrive. Each incoming data point is used to update the existing models, which then make a prediction on the next data point. Online ensemble learning is beneficial when the data is constantly changing, and the model needs to be updated in real-time [170]. In summary, chunk-based ensemble learning can be useful for processing large datasets in parallel and can handle non-stationary data, while online ensemble learning can adapt to changes in the data over time and can handle data streams. However, both methods have their trade-offs and require careful consideration of the specific problem and data characteristics.

To conclude, ensemble methods can also be highly effective in handling imbalanced data streams, especially when combined with dedicated combination schemes or adaptive chunk-based learning techniques [108]. To enhance their performance in minority classes, weights assigned to each base classifier can be continuously updated based on their current competencies [133]. A hybrid approach that combines resampling of minority instances with a dynamic weighting of base classifiers based on their predictive performance on sliding

windows of minority samples can be particularly effective [40, 41, 161]. Dynamic selection of classifiers and their associated preprocessing techniques can also be a useful tool for handling concept drift, allowing for the exploitation of diversity among base classifiers [171, 172].

2.4 Fairness-aware Learning

Machine learning algorithms have spread in every aspect of our lives through different applications. These algorithms are used in movie recommendations, suggest products to buy, and who to date. They are increasingly used in courts to assess the probability that a defendant recommits a crime or high-stakes scenarios such as allocating loans [121] and hiring decisions [127]. All of these applications have a direct effect on our lives while they are vulnerable to biases that render their decisions "unfair" and can harm our society if not designed and engineered correctly, that is, with consideration to fairness. In the context of decision-making, fairness is the absence of any prejudice or favoritism toward an individual or group based on their inherent or acquired characteristics [167]. Thus, an unfair algorithm is one whose decisions are skewed toward or against a particular group of people. Therefore, it is important for researchers and engineers to be concerned about the downstream applications and their potentially harmful effects when modelling an algorithm or a system.

2.4.1 Basic Concepts and Problem Definition

Assume a set of attributes $\{A_1, \ldots, A_d\}$ with their respective domains $dom(A_i)$ and let class be the class attribute of the classification problem. The data stream D consists of potentially infinite instances over the schema $(A_1, \ldots, A_d, Class)$ arriving over time, with each instance $x_t \in D$ being an element of $dom(A_1) \times \ldots \times dom(A_d)$ and class label $c_t \in dom(Class)$ [167]. We assume an attribute S, referred to as a sensitive attribute with a special value $s \in dom(S)$, which is a sensitive value that defines the discriminated group. Also, we assume that S is a binary attribute: $dom(S) = \{s, \bar{s}\}$. As an example, we use $S =$ "$gender$" as the sensitive attribute and $\bar{s} =$ "$female$" as the sensitive value (with $s =$ "$male$"). We also assume the class is binary with values {0 as rejected, 1 as granted}.

2.4.2 Causes of Unfairness

The literature has indicated several reasons that may lead to unfairness in the machine learning area [153, 167]. These causes can be divided into data-based and algorithm-based roots. Most algorithms require data upon which to be trained. So, it is easy to understand why biases in the data can lead to unfairness in machine learning models. Algorithms can even amplify existing biases in the data. However, even if the data do not contain any biases, the learned model can still produce unfair results. In addition, algorithms themselves can display biased behaviour due to model designs, even if the data are not biased. The outcomes of these biased algorithms can then be fed into real-world systems and affect users' decisions, which will result in more biased data for training future algorithms. We hereby explain some types of biases for unfairness identified in the literature:

Measurement bias: Measurement, or reporting, bias is based on the processes used to choose, utilize, and measure particular features [147]. An example of this type of bias was observed in the recidivism risk prediction tool COMPAS [98], where a person's family and friend history was used as an effective variable to measure that person's level of "riskiness" or "crime." This may cause a more controlled environment or higher arrest rates in minority communities. However, it is not a fair decision to judge that a person is more dangerous because that person comes from minority groups with historically higher arrest rates.

Missed variable bias: When one or more important variables are left out of the model, we have missed/omitted bias [135]. In these cases, a model may be unable to see a variable that drives changes in prediction or performance. One example is a new competitor entering the market with a lower price. A subscriber-retention model built to predict, with relatively high accuracy, the annual percentage rate at which customers will stop subscribing may not use data about new competitors. Because the model cannot account for the entry of a new competitor into the market, it will fail to show the correct reason for losing customers.

Sampling bias arises due to non-random sampling of subgroups [89]. As a consequence of sampling bias, the trends estimated for one population may not generalize to data collected from a new population.

Algorithmic bias is when the bias is not present in the input data and is added by the algorithm [48]. The process of modelling, optimization functions, applying regression

models on the data as a whole or considering subgroups, or use of statistically biased estimators in algorithms can all contribute to biased algorithmic decisions.

Evaluation bias happens during model evaluation. This includes the use of inappropriate and insufficient measures to evaluate applications. Results of this bias have been seen in the evaluation of facial recognition systems that were biased toward skin color and gender [147].

2.4.3 Fairness Definition and Measurements

This section presents some discrimination notions followed by some of the most common measures for algorithmic fairness [153]. With a few exceptions, the vast majority of work to date on fairness in machine learning has focused on the task of offline classification. At a general level, we define two main families of discrimination: disparate treatment and disparate impact [3]. The former means intentionally treating an individual differently based on their membership in a protected class (direct discrimination), while the latter refers to negatively affecting members of a protected class more than others, even if by a seemingly neutral policy (indirect discrimination). We define how to measure fairness in algorithms based on these notions. Generally, fairness definitions can be categorized into three groups. **Individual Fairness** gives similar predictions to similar individuals [127], and **Group Fairness** treats different groups equally [83]. **Subgroup Fairness** aims to achieve the best benefits of both group and individual notions of fairness and uses both concepts to obtain better outcomes [89].

Disparate impact: This measure [102] requires a high ratio between the positive outcome rates of both protected and non-protected groups. This keeps the proportion of the positive predictions similar across groups. For example, if a positive prediction represents acceptance for a job, the condition requires the proportion of accepted applicants to be similar across groups. Formally, this measure is computed as follows:

$$\frac{P(f(x) = y^+ \| \bar{z})}{P(f(x) = y^+ \| z)} \geq 1 - \epsilon \tag{2.8}$$

where z represents the protected attribute (e.g. race or gender), and $\bar{z}$ is the privileged group. $f(x) = y^+$ means that the prediction is positive. A higher value of this measure represents more similar rates across groups and, therefore, more fairness.

Demographic parity: This measure is also known as statistical parity and is calculated as follows [89]:

$$S.P. = P(f(x) = y^+ \| \bar{z}) - P(f(x) = y^+ \| z) \tag{2.9}$$

The result calculates the difference between the probability of a random individual drawn from the protected group being predicted as positive and the probability of a random individual drawn from a non-protected group being predicted as positive.

Equalized odds: This measure computes the difference between the false-positive rates (FPRs) and the true-positive rates (TPRs) of the two groups [74]. Formally, this measure is computed as follows:

$$\mid P(f(x) = y^+ \| \bar{z}, y^-) - P(f(x) = y^+ \| z, y^-) \mid \leq \epsilon \tag{2.10}$$

$$\mid P(f(x) = y^+ \| \bar{z}, y^+) - P(f(x) = y^+ \| z, y^+) \mid \leq \epsilon \tag{2.11}$$

where both the FPRs and TPRs are bounded by ϵ Smaller differences between groups indicate better fairness.

Equal opportunity: One common statistical measure, equal opportunity [153], means that the probability of a person in a positive class being assigned to a positive outcome should be equal for both protected and unprotected (often, female and male) group members. The definition of this metric is "A binary predictor $\bar{Y}$ satisfies equal opportunity with respect to Z as sensitive attribute and Y as class label, if $P(f(x) = y^+ \| \bar{z}) = P(f(x) = y^+ \| z)$" [153]. In other words, the equal opportunity definition states that the protected and unprotected groups should have equal true positive rates. Another algorithm calculates whether the classification rates are equal for these groups (also known as statistical parity), which means the likelihood of a positive outcome should be the same regardless of whether the person is in a protected (for example, female) group.

Individual fairness: This measure means that similar individuals will be treated similarly [58]. Individual fairness may be described as follows:

$$\mid P(f(x)^i = y \| x^i, z^i) - P(f(x)^j = y \| x^j, z^j) \mid \leq \epsilon; sim(i, j) = 0 \tag{2.12}$$

where i and j refer to two individuals from data, z^i denotes the individual's sensitive attributes, and x is their features. $sim(i, j)$ is a measure to calculate the similarity between two people who should be similar. This measure also considers other individual attributes for defining fairness rather than only the sensitive attributes.

2.4.4 Fairness-aware Machine Learning Methods

A variety of methods have been proposed that satisfy some of the fairness definitions or other new definitions depending on the application. As already noted, most of these studies assume offline settings by design and can be typically categorized into three main strategies [121], namely **pre-processing**, **in-processing**, and **post-processing** techniques. The first strategy in this category, preprocessing methods, works under the assumption that the training data should be discrimination-free to learn a fair classifier. If the algorithm is allowed to modify the training data, the designers aim to balance the representation of the different groups in the population [14]. Massaging [83] directly swaps the class labels of selected instances to change data distribution for the sake of balanced representation. The swapped instances are selected using a ranker based on the potential accuracy deterioration to minimize accuracy loss while reducing discrimination. Re-weighting [91] instead of intrusively relabeling the instances assigns different weights to different communities to reduce discrimination. Instances belonging to the unprivileged group will receive higher weights than instances from the privileged group. However, methods in this category are typically not quite as effective as standalone methods unless they are used in conjunction with other methods with sophisticated designs. Further, techniques in this category are typically model-agnostic, and therefore, any classifier is applicable after the preprocessing phase.

The second group, in-processing approaches aim to modify and change state-of-the-art learning algorithms to limit discrimination [167]. Therefore, distinct from the classifier-agnostic strategy, these methods are algorithm-specific. This implies that in-processing can be used during the training of a model, either by incorporating changes into the objective function or imposing a constraint. Specifically, [86] represents one of the seminal in-processing works in which discrimination, reflected by the entropy w.r.t. the sensitive attribute, is incorporated into the splitting criterion for fair tree induction. In [62], the measure of "decision boundary fairness" is leveraged to penalize discrimination in the formulation of a set of convex margin-based classifiers. More recently, [3] proposed the notion of "fair group construction" to emphasize sensitive attributes in the classification process for the sake of improving fairness in prediction outcomes.

The last post-processing category consists of either adjusting the decision boundary of a model or directly changing the prediction labels. It modifies the resulting models by "correcting" the decision regions that lead to redlining for a fair representation of different subgroups in the final decision process. Kamiran et al. [86] carefully relabels selected leaves of a decision tree model to reduce discrimination with the least possible impact on accuracy. The latter approaches pay attention to the outcome of a classifier.

Most of the proposed fairness-aware learning algorithms consider fairness as a static

problem and assume that the data is generated by a single concept without drift. However, as mentioned earlier, in many real-world applications, data is generated in a streaming fashion and might also evolve over time, which further requires the model to be able to adapt to the evolving bias patterns. That is, the proposed techniques should be able to maintain an effective trade-off between accuracy and fairness while considering the removal of prediction biased on the sensitive attributes and the imbalance ratio of targeted classes. We introduce a few state-of-the-art fairness algorithms adapted to an evolving stream.

FAHT: This "fairness-aware" method builds upon the Hoeffding tree (HT) classifier. That is, FAHT extends HT to make fair decisions as the tree is growing [166]. To this end, the authors introduce two new splitting criteria: fair information gain, which jointly considers the gain of an attribute split classification and discrimination, and adaptive Fair Information Gain (FIG), which additionally considers evolving data distribution and fine-grained control of fairness. FAHT aims at optimizing both predictive performance and fairness of splits.

FEAT: This algorithm [165] is similar to FAHT and built on top of the HT classifier. It extends the information gain (IG) criteria by introducing an enhanced fair-splitting criterion that enables fairness-aware learning by adding the ability to detect and adapt to the evolution of underlying distribution. In FEAT, the entropy regarding the sensitive attribute is modified to directly adjust the fairness gain according to the discrimination difference due to the split. Further, the new enhanced splitting evaluation metric counts for local discrimination to maximize the cumulative fairness by assigning equal weights to different discrimination representations.

FABBOO: This approach [82] is based on an online boosting ensemble that changes the training distribution of evolving data, while taking into account class imbalance. The algorithm employs post-processing fairness by monitoring the cumulative fairness measure over the stream, then adjusting the decision boundary on demand.

2.5 Explainability and Interpretability of Machine Learning

Machine learning models are improving rapidly and providing more-accurate solutions to complex problems. However, with increased accuracy comes greater complexity, which makes explaining these models challenging. Additionally, black-box systems may utilize

advanced machine learning models to predict sensitive information. Interpretation involves explaining and presenting concepts in terms that are understandable to humans. Therefore, the interpretability of these systems refers to their ability to provide explanations in understandable terms [55].

The study of machine-learning interpretability approaches is an interdisciplinary field that draws from various domains, including computer science, statistics, psychology, and philosophy [1]. These approaches can be categorized into **pre-modelling explainability**, **interpretable model**, and **post-modelling explainability**.

Pre-modelling explainability uses data-processing techniques to gain insights into the datasets used to train machine learning models [10]. Important methods within pre-modelling explainability include data analysis, data summarization, and data transformation. Data analysis [134] uses techniques like sensitivity analysis to provide an overview of statistical information. Sensitivity analysis evaluates the uncertainty in the outcome of a black-box with respect to different sources of uncertainty in its inputs [78]. This method identifies which input features have the greatest influence on the model output and how the model behaviour changes when the input variables are perturbed or varied. Data summarization [6] attempts to find a minimal subset of the original dataset. Data transformations promote effective interactions between dataset creators and users, reducing issues such as data bias.

A model is considered interpretable if humans can understand it by examining the model summary or parameters. There are various approaches in the interpretable model group, including inherently interpretable models [148]. Inherently **interpretable models** consist of techniques and algorithms designed to be understandable by design. A common strategy used to generate explanations for decisions made by a black-box machine learning model is to build a surrogate model based on these inherently interpretable machine learning algorithms [29]. These surrogate models can be divided into two categories based on a global focus and a local focus. A global surrogate model is an interpretable model trained to approximate the behaviour of the black-box model across its entire input space [26]. By providing abstract-level transparency, globally interpretable models offer a clear understanding of what is happening inside an original model constructed by the algorithm. Local surrogate models are trained to approximate the behaviour of a black-box model for a specific instance or subset of instances. These models are typically simpler and require less training data [148]. Standard algorithms for this approach include linear models, decision trees, and decision rules. Linear regression is a statistical technique that models the relationship between a dependent variable and one or more independent variables. In an interpretable linear regression model, the coefficients of the model's features are easily understandable, making it easier for humans to interpret how each feature affects the prediction [72]. Decision trees are a type of machine learning model that predict outcomes

by recursively dividing the data into smaller subsets based on input feature values. The algorithm selects the input feature that best divides the data into subsets with the most homogeneous classes or the smallest variance in the case of regression. Decision trees are considered to be one of the more interpretable and understandable models [80]. Decision rules are also easily understandable. They describe the conditions and actions for making decisions in a particular domain or problem. These rules can be used for transparent design and to explain the model or outcome. Moreover, there are techniques available to transform a decision tree into a set of rules, making them even more accessible for interpretation [151].

Finally, the common methods for achieving **post-modelling explainability** include textual justification, visualization, and feature importance. These techniques aim to enhance the interpretability of black-box machine learning models by providing detailed explanations and insights into how the model works. Textual justification produces explanations in the form of text for every decision the model makes, so it is easier for humans to understand the model's thought process [138]. Visualization provides visual representations of the model's behaviour, allowing users to explore and understand the relationships between different variables. Visualization techniques are often considered the most effective way to explain the complex inner workings of a model's variables and their interactions [78]. The feature importance approach is simple and effective and provides explanations by determining the relative importance of input features in making predictions or classifications. This method helps us to understand the underlying patterns and relationships in the data [32].

2.5.1 Discussion

The main advantage of pre-modelling explainability is that it can ensure the model is being trained on meaningful and relevant data, which can ultimately lead to better performance and interpretability. Data analysis techniques, such as visualization and statistical analysis, can identify patterns and relationships in the data that may be useful for understanding the model's behaviour. Data summarization techniques, such as feature selection and dimensionality reduction, can simplify the data and make it more manageable for modelling. Data transformation techniques, such as normalization and scaling, can ensure that the data's format is suitable for the model to learn from. However, one potential disadvantage of pre-modelling approaches is that they can require considerable manual effort and expertise for effective implementation, and they may not always apply to all types of data or models. Additionally, they may not fully capture the complexity of real-world data. We need to fully understand the data before using it to make the model [56].

Interpretable model explainability approaches have several benefits, such as being easily comprehensible to humans, providing transparency in decision-making, and being inherently interpretable without requiring additional explanation techniques. These attributes can assist stakeholders in better understanding the model's decision-making process and identifying and addressing issues related to fairness, bias, or discrimination. However, this simplification may lead to reduced accuracy, and the surrogate is usually unable to reproduce the results perfectly. Thus, it is crucial to evaluate the performance of this approach and find the trade-off between the model's performance and explainability. Additionally, by creating a surrogate model we can only derive conclusions about the machine learning model, not the data. [142]. In contrast, post-modelling explainability approaches have the advantage of applicability to any model, regardless of its complexity or design. Yet, this approach also has drawbacks, such as the potential to generate biased explanations and the need for additional computational resources to produce them [99].

To conclude, selecting an appropriate explainability approach in machine learning systems depends on various factors, such as the specific use case and application, available resources and expertise, and the desired level of interpretability and transparency in the system.

2.6 Summary

In this chapter, we provide an overview of the background concepts and related works that form the basis of our research. We begin by defining online machine learning and exploring related concepts. We then delve into previous studies on imbalanced datasets, focusing on multi-class solutions. Given that our research involves fairness considerations, we thoroughly examine discrimination definitions and measures. We also review the concepts of interpretability and explainability in machine learning algorithms.

In the following three chapters, we elaborate on our contributions in the areas of online multi-class imbalanced learning, online fairness-aware learning, and the interpretability and explainability of our discrimination-aware algorithm.

Chapter 3

DynaQ: Online Learning from Imbalanced Multi-Class Streams through Dynamic Sampling

The work presented in this chapter is based on an extension of our initial paper: *Farnaz Sadeghi and Herna Viktor, "Online-MC-Queue: Learning from Imbalanced Multi-Class Streams", the Third International Workshop on Learning with Imbalanced Domains: Theory and Applications (LIDTA 2021), Virtual Event, Proceedings of Machine Learning Research (PMLR), 154, pp. 21-34, 2021*. The extended version of this work has been published as: *Farnaz Sadeghi, Herna Viktor and Parsa Vafaie, "DynaQ: Online Learning from Imbalanced Multi-Class Streams through Dynamic Sampling", Applied Intelligence, Springer Nature, pp. 1-23, (2023).*

Recall from Chapter 1 that one of our main aims in this thesis is to design an online algorithm for multi-class imbalanced data present in evolving streams. In this chapter, we introduce our DynaQ [141] framework that was designed and implemented to accomplish this task. This chapter is organized as follows. Section 3.1 provides an introduction to our work and Section 3.2 highlights related work in online class imbalance, while Section 3.3 details the DynaQ process. This is followed in Section 3.4 with an experimental evaluation, while Section 3.5 concludes the chapter.

3.1 Introduction

In dynamic data streaming environments, online learning algorithms consider incoming examples "on arrival" without the need for persistent storage and multiple scans, while maintaining a model that reflects the current data. This type of learning has applications in various real-world applications, such as network intrusion detection, spam filtering, fault diagnostics in manufacturing, and e-commerce applications [71]. Learning from such streaming data is challenging, especially in the presence of multiple skewed class distributions, also known as "multi-class imbalance", where a large number of majority-class instances may lead to the minority classes being ignored. This problem is aggravated in an online setting because a steady arrival of minority instances cannot be guaranteed, and a minority class may become a majority concept and vice versa [59]. In addition, evolving streams are susceptible to concept drifts, which is the phenomenon of unexpected changes in the underlying data distribution [106].

While previous studies have primarily focused on binary imbalanced data or stationary streams, only a limited number of studies have addressed the challenge of learning from evolving streams with multi-class imbalanced data [5]. For instance, [59] proposed modifications to resampling methods to adjust for multi-class imbalanced data. In addition, [40, 41] introduced ensemble-based approaches utilizing random feature subsets and resampling to address imbalances. However, further investigation is needed to explore classifier-agnostic approaches that can effectively utilize alternative evaluation metrics and resampling parameters to address the challenges of imbalanced datasets. Additionally, it is crucial to develop class-based drift detection methods to prevent performance deterioration.

Only a few studies have addressed the combined problem of learning from such evolving streams that contain a multi-class imbalance, that is, more than two classes. Recent studies [59] have mainly focused on binary imbalanced data, proposed modifications of resampling methods, or worked only with stationary streams. Also, they do not address dynamic class evolution. This section introduces an algorithm for learning in a multi-class imbalance setting. Our dynamic queues (DynaQ) approach extends the online-multi-class-queue (OMCQ) [140] approach that maintains separate queues for each class, without any form of sampling. In addition, our DynaQ algorithm uses oversampling with replacement based on a rate parameter associated with the various classes. That is, we oversample the minority classes based on the recall, F1-score, κ_m and Euclidean distance.

Our DynaQ algorithm combines batch-incremental and instance-incremental ensemble learning. Initially, a batch of data with the classes that have been seen so far is presented to the learner, and it subsequently updates the model with new instances as they arrive. Each

member of the ensemble learns from a sliding batch, and the results are combined using soft voting. Furthermore, we incorporate a class-specific concept drift detection mechanism into DynaQ. Our algorithm can thus dynamically adapt to changes in label arrivals and individual class performances, as monitored by the recalls. Specifically, we do not make any assumptions regarding the frequency of classes, which implies that minority classes may become majority classes, and vice versa. Our experimental results confirm that our DynaQ algorithm is efficient in terms of multi-class concept separation.

The main contributions of this study are:

Novel online queue-based ensemble architecture: DynaQ utilizes a batch-based resampling method that creates an instance queue for each class and implements an ensemble approach that utilizes sliding windows and a soft voting schema.

Self-handling class imbalance: DynaQ dynamically oversampling minority classes based on a rate parameter associated with the various classes.

Class-based concept drift approach: DynaQ incorporates a drift detection mechanism to dynamically adapt to label arrivals and individual class performance changes.

Multi-class imbalanced and concept drift stream: DynaQ highlights the challenges of learning from dynamic data streaming environments while addressing the problem of multi-class imbalance and concept drift in online learning.

3.2 Background and Related Work

In online learning, a data-generating process provides at each time step t a sequence of examples (x_t, y_t) from an unknown probability distribution, where x_t is a vector consisting of qualitative or quantitative f features and $y_t \in Y$ is the class label, where $Y = \{c_1, c_2, \ldots, c_s\}$, and S is the number of classes. An online classifier is built receiving an example x_t at time step t, resulting in a prediction $\hat{y}_t$. In a supervised learning setting, the label y_t is available, and the performance of a learning algorithm is evaluated using a loss function $f(x_t) = l(y_t, \hat{y}_t)$ to find the best predictor for future data at each step [71]. This section focuses on online learning from multi-class imbalanced data, where $S > 2$, from evolving streams.

3.2.1 Online class imbalance learning

In an online class imbalance learning setting, the main goal is to correctly classify minority examples because the minority class is often of the most interest. Recall that existing online learning approaches that address the class imbalance problem maybe categorized into data level, algorithmic and hybrid ensemble-based approaches. In data level techniques, several methods proposed for solving class imbalance problems provide solutions that include resampling and feature selection [9]. Recall from Chapter 2 that data level modifications aim to balance the underlying dataset, making them classifier-agnostic approaches. Resampling is an effective data level approach that proceeds independently of the learning algorithm; this method has been used for binary classification problems in the data stream setting. The major types of resampling are oversampling (increasing the number of minority-class examples), undersampling (reducing the number of majority-class examples), and hybrid sampling [71]. As we introduced in Chapter 2, Synthetic minority over-sampling technique (SMOTE) [60] is an over-sampling technique that creates new instances of the minority class by interpolating between existing instances that are located close to each other. This approach effectively reduces the degree of class imbalance compared to the original majority-to-minority class ratio. Recently, new modifications of the SMOTE algorithm have been introduced that enable it to work compatible with streaming data [17–19]. The technique known as Selection-Based Resampling (SRE) [133], utilizes undersampling to eliminate safe instances from the majority class in an iterative manner, without causing reverse bias towards the minority class. However, undersampling methods may lead to crucial information being overlooked, whereas oversampling may potentially introduce artificial instances that may be deemed unacceptable in real-world *domains. Hybrid ensemble-based methods combine both resampling and algorithmic approaches to manage the class imbalance [30]. As we discussed in Chapter 2, the authors integrated resampling into ensemble algorithms to define the oversampling online bagging (OOB) and undersampling online bagging (UOB) techniques for binary classification. The work extends bagging ensembles following a class-based ensemble approach to dynamically change the learning rate by maintaining a base learner for each class, and dynamically updating the base learners with new data to deal with binary class imbalance. Resampling will be triggered to either increase the chance of training minority-class examples (in OOB) or reduce the chance of training majority-class examples (in UOB).

Online queue-based resampling [112] has also been proposed as a method for binary classification. The main idea of this algorithm is to selectively include a subset of the positive and negative examples in the training set that thus far have appeared in the stream. The examples are selected by maintaining, at any given time t, separate queues based on class labels received from data of equal lengths $L \in \mathbb{Z}^+$, $q_n^t = \{(x_i, y_i)\}_{i=1}^{L}$ and $q_p^t = \{(x_i, y_i)\}_{i=1}^{L}$

that contain the negatives as majority examples and the positives as minority examples, respectively. Once the queues are filled, the classifier is incrementally updated after combining the two queues into one training set [112]. The algorithm employs an interleaved test-then-train evaluation [21] in which each example is used to test the model before the example is appended to the queues for training, thus implementing a sliding window method. Our work extends the notion of queues as presented in [112] to the multi-class scenario by incorporating explicit concept drift detection during learning.

3.2.2 Multi-class imbalanced learning

Recall from Chapter 2, that multi-class classification problems are often considered more challenging than their binary counterparts because multiple classes can increase the data complexity and aggravate the imbalanced distribution [59]. We also categorized the methods into binary decomposition techniques, algorithmic level modifications using misclassification costs, and resampling methods [5, 156]. Binary decomposition algorithms typically combine binarization techniques that transform the original multi-class data into binary subsets [94].

As mentioned in Chapter 2 one of the commonly used binarization strategies is one-versus-one (OVO) [61] decomposition where it first selects a subset from the original data that only contains the instances for each pair of classes and proceeds to train a binary classifier for each pair. In contrast, in the one-versus-all (OVA) approach [61], a multi-class data set is decomposed into several binary class problems and subsequently trains single classifiers or each class by considering a single class versus a combination of all the remaining classes. For instance, in [90], adaptive online one-class Support Vector Machines (SVMs) are used to monitor changes in minority classes over time. In [47] one-class classification is integrated with ensembles using over-sampling and instance selection techniques to balance the class distribution of incoming data batches, which are then used to induce classifier ensembles. A disadvantage of binarization techniques is that the interactions between multiple classes cannot be considered simultaneously.

Recall that algorithmic level approaches adapt the training process to enhance the classifiers' ability to deal with skewed distributions. These methods are often specific to particular learning models, making them more specialized but less flexible than data-level approaches. One of the most common algorithm modifications for addressing class imbalance combines Hoeffding Trees with the Hellinger splitting criteria for imbalanced domains. In GHVFDT [110], Hoeffding Trees are used to construct a decision tree that can handle data streams, while the Hellinger distance is used as a splitting criterion that is less sensitive to class imbalance. Further, [96] introduced an approach that modifies predictions made

by a base classifier to address imbalanced data streams. This algorithm aims to map prior probabilities in the statistics of assigned classes. Few research studies considered multi-class online learning from evolving streams, focusing on resampling technique, combining resampling and online ensemble learning [5, 95]. These approaches aim to learn from data streams in a single pass. One frequently employed approach is the online bagging (OB) algorithm, as mentioned earlier. This algorithm adapts the batch-based bagging method for online settings by using the Poisson distribution to determine the value of k for sampling with replacement. In Chapter 2, we also introduced MOOB and MOUB, which utilize resampling techniques within the OB framework to deal with class imbalance. However, these algorithms rely on certain assumptions, such as the efficiency and lack of bias when sampling based solely on class sizes, that limits there usefulness across numerous domains.

Some approaches employ feature space modification to define what feature space input is used by base classifiers. In [40] diverse feature subspaces of random sizes are used to improve the ensemble's performance. The Kappa Updated Ensemble (KUE) method employs a combination of base classifiers that are updated dynamically based on a κ statistic, which measures the agreement between the base classifiers on a sliding window of recent data. KUE also incorporates an instance weighting scheme that prioritizes recent data over older data. Similarly, [41] introduce the Robust Online Self-Adjusting Ensemble (ROSE), an online ensemble-based algorithm that combines classifiers trained on variable-sized random subsets of features. The ROSE algorithm incorporates undersampling of the majority classes and employs so-called background classifiers initiated once drift is detected. In essence, ROSE is based on self-adjusting bagging and balances the training sets by monitoring the production of the accuracy and κ statistic. Specifically, each window provides a background classifier with a balanced data set of recent instances at the start of model construction. Background classifiers are dynamically added to the ensemble once their performance surpasses a threshold.

The OMCQ framework maintains multiple queues for each class in the multi-class learning setting [140]. This algorithm learns directly from the original data without any resampling and incorporates a drift detection mechanism that can adapt to class sizes. While previous results were promising, it follows that the OMCQ approach may be extended to use dynamic sampling, as will be discussed in Section 3.3. The authors in [150] introduced a new method for dealing with multi-class imbalanced data called improved online ensembles (IOE) for semi-supervised learning. In this technique, instances from the minority and majority classes are sampled based on the classifier's performance, measured in terms of the recall of each class. That is, classes with recalls that are lower than average are oversampled, while classes with higher recalls are undersampled. The sampling in the IOE algorithm is controlled by setting the rate parameter of the Poisson distribution based on the recall score of classes, which controls the number of times each instance is used for

training each base learner in the ensemble learning model. At each time step, the ensemble model is employed so that the N individual classifiers are trained by oversampled or undersampled instances.

In cases where ensembles have limited access to labels, a set of algorithms is also available [150, 164]. One such approach is CALMID [103], which is a robust framework that deals with limited label access, concept drift, and class imbalance by dynamically inducing new base classifiers and weighting the most relevant instances. The method uses a variable threshold uncertainty strategy based on an asymmetric margin threshold matrix to address the problem of a given class being a majority to a given subset of classes while also being a minority to others. A novel sample weight formula is designed to consider the class imbalance ratio of the sample's category and the prediction difficulty.

Next, we discuss our DynaQ framework.

3.3 DynaQ Framework

Our DynaQ framework maintains a queue for each of our multiple classes. Initially, all queues will be empty. As instances arrive, they are added to the appropriate queue as per their true label. When the first queue has been filled, our sampling and training processes commence. Figure 3.1 illustrates how our contributions fit together and operate in one iteration of an interleaved test-then-train loop. For each arriving instance, minority classes are oversampled to balance the different classes. As noted in Section 3.2, the sampling process is based on the rate parameters of the classes, where classes with rates lower than the average are oversampled, while classes with higher rates use the original training instances. That is, our algorithm oversamples minority instances, while majority classes are not undersampled.

The online learning phase of DynaQ thus incorporates four processes: evaluation, dynamic class balancing, ensemble learning, and concept drift detection. Our algorithm creates a queue space in the class balancing module to separate the instances from each class as they arrive within the stream. Subsequently, if the queue for a class is not full or the parameter rate for that class is below a threshold, the queue is updated with an oversampled instance; otherwise, we use the instance only once by inserting it in the related queue. The concept drift detector utilized in this study captures changes in data distributions by adopting the concept of the drift detection method (DDM), as introduced in Chapter 2. It effectively updates the instances in the queues to account for these changes. The online ensemble component updates a single learner per iteration of the interleaved test-then-train loop in a cyclical manner, meaning that each classifier in the ensemble is

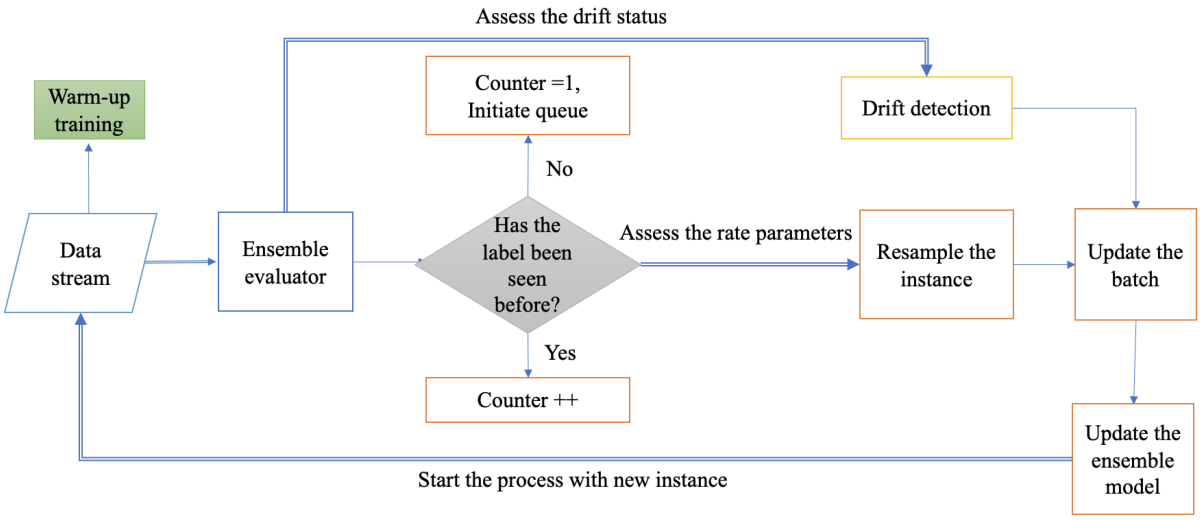


Figure 3.1: High-level overview of DynaQ methodology

trained only once every N loop iterations; N is the number of classifiers in an ensemble. That is, a different learner from the ensemble is trained at the next loop iteration, with the sliding batch including the new instance and overlapping with the previous one.

Specifically, the evaluator is used to predict the class label of arriving instances, using soft voting, and to update the evaluation metrics. In the class balancing module, our algorithm creates a queue space to separate the instances from each class as they arrive within the stream. Subsequently, if the queue for a class is not full or the rate parameter for that class is below a threshold, the queue is updated with the oversampled instance. Otherwise, as indicated above, we use the instance only once by inserting it in the related queue. During online learning, a single learner is trained during a single iteration of the interleaved test-then-train loop in a cyclical manner. A soft voting-based ensemble of classifiers is used during training to incrementally update the model using sliding batches. This implies that each classifier in the ensemble is trained only once every N loop iteration, where N is the number of classifiers in the ensemble. That is, a different learner from the ensemble is trained at the next loop iteration, where the sliding batch includes one new instance. Next, we present the details of our DynaQ algorithm, as also shown in Algorithm 3.1.

3.3.1 Online queue construction

In an offline supervised learning setting, it follows that a data set D is available with input signals x_i and output y_i. The task is to infer a model $M \approx p(y \mid x)$ from such data. In contrast, online learning refers to approaches when the full data set D is not available at the time of learning. Here, examples arrive over time, and the task is to infer a reliable model M_t at time step t based on the newly arriving example (x_t, y_t) and the previous model M_{t-1}.

Incremental online learning is a sub-area of online learning that is additionally bounded by memory resources and the capability of continuous learning with limited data compared to offline learning. In the literature, approaches to incremental learning can generally be categorized as batch incremental and instance incremental [131]. As the name suggests, batch learning methods employ batches of data to form hypotheses about the data. At every time step t, this form of training collects the k newest instances to form a batch. When a batch of data D_t is filled, a model m_t is learned [105]. This process continues, batch by batch. In our work, following [112], we consider a sequence of streaming data $f = \{(x_1, y_1), \ldots, (x_t, y_t)\} \in R^n \times \{1, \ldots, S\}$, where f is the data dimension, and S is the total number of classes. The key idea is to keep a fixed number of examples (queue size denoted by L) for each class in a stream to combine the training set. In other words, each arriving sample (x_t, y_t) at any given time t will be stored in a separate queue of equal length $q^t_{C_S} = L$, where c_s is the class label received with the data. Together, the queues form a sliding batch B_t. This method considers a given stream of data $x_1, x_2, ..., x_t$ and learns from a sequence of batches $b_1, b_2, ..., b_t$, where the batches are updated as instances arrive and update the model.

Figure 3.2 illustrates how $Queue_L$ works when $q = 3$. The upper part shows the examples that arrive at each time step; for example, z^0 and z^4 arrive at $t = 0$ and $t = 4$, respectively. Assume that the data stream contains three classes $Y = \{c_1, c_2, c_3\}$ and that all instances have their own queues. The queues are of equal length $L \in \mathbb{Z}^+$, $q^t_{C_1} = L$, $q^t_{C_2} = L$, and $q^t_{C_3} = L$ and contain the samples of class c_1, class c_2, and class c_3, respectively. After instances have been separated based on their labels, the arriving samples for class c_s are placed at the front of the $queue_{c_s}$. When the queues fill, we combine the full queues in the batch to add to the training set and commence online learning. Here, the sliding batch explicitly employs a forgetting mechanism, where the oldest instance will be removed from the head of the related queue.

As shown in Figure 3.3, B_t is a batch of data, including the full queues of current classes, defined as the number of instances used to update the model. As an initial training step, we construct an initial model with warm-up instances from the data stream. Whenever the

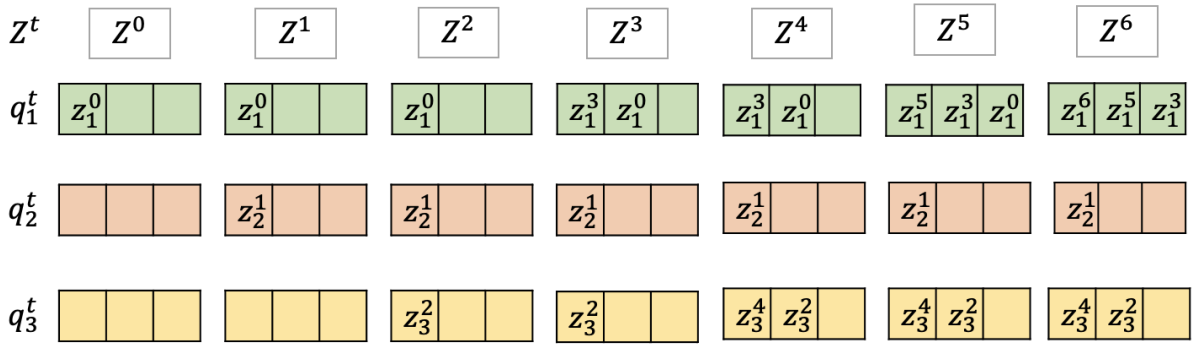


Figure 3.2: Example of $Queue_3$ resampling (adapted from [112])

first queue is full, we proceed to initiate our rebalancing process until we have created a full queue for each class. We include those queues in our batch. Next, the model construction process is initiated, and this batch-incremental process continues until the end of the stream. For each arriving instance (x_t, y_t) at time t, the oldest sample from the queue to which y_t belongs is removed, and the recent sample is added to batch B_t. Whenever the first queues are full, they are used as our first batch, and we then proceed to update the model. Meanwhile, at each point in time, only one learner is updated with the batch that includes the new resampled instance in a circular order. The learner will use batch B_t to update its model; that is, the training process utilizes a balanced set consisting of the most recent data. The algorithm waits until it has enough instances from the classes, including the current minority classes, before updating its model. It follows that the sizes of the individual queues are highly domain-dependent; the size of the queue is set by inspection.

3.3.2 Queue-based sampling

As noted above, our DynaQ methodology employs queue-based sampling of original instances of each class to dynamically construct models against all classes, using an ensemble against a sliding batch. Next, we will explain the process of sampling each instance in the queues. Recall that minority classes suffer from not having enough data to present against majority classes. Following [150], we oversample recent instances of each class based on the rate parameter of the classes while maintaining the majority instances without any form of undersampling. This is done dynamically during learning as the stream evolves. In this way, the learner has access to newer samples and concepts, and we can balance the number of instances for all classes. In addition, the oversampling of minority classes

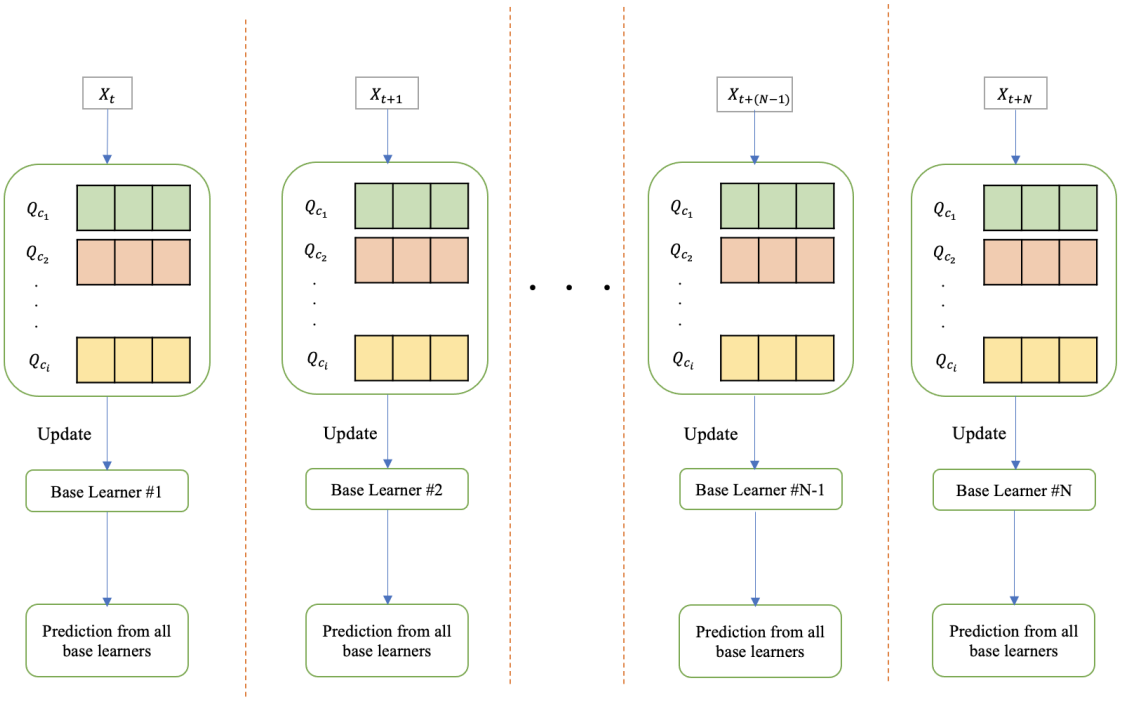


Figure 3.3: Illustration of batch-instance incremental learning process.

implies that the associated queues will fill faster. We employ a sliding batch of S queues, where S is the number of current classes, and N learners are updated one at a time in a periodic order with each arriving instance. The oversampling rate of our DynaQ algorithm is implemented considering four different metrics, as discussed below.

3.3.2.1 Recall-based Sampling

In our first variant, the rate parameter is set based on the recalls of the classes [150]. That is, we employ the recall value to maintain a balance in terms of the samples from the majority and minority classes that we maintain, where we define:

$$recall = \frac{TP}{TP + FN} \tag{3.1}$$

and where true positive (TP) refers to the instances from the actual class that are correctly classified, while false negative (FN) denotes those instances with incorrect predictions. As such, recall measures the ratio of correctly classified instances from the minority class (true

positive rate). When employing the recall scores for the multi-class data stream, we may assess how many of the examples from each class are correctly classified. A model with high recall on each class successfully predicts true labels in the data [157]. If the absolute difference between the recall score for a class and the mean recall is higher than the pre-defined threshold, then the examples for this class will be oversampled using recalls, as follows.

$$k = Poisson(r_{avg\_excluding\_c}/r_c) \tag{3.2}$$

Where, $r_{avg\_excluding\_c}$ is the average recall for the classes excluding class c and r_c is the recall for class c, which is calculated based on prequential evaluation, where each instance will be used to test the model before it is used for training [169], and from this, the recall can be incrementally updated. If the recall for a class is lower than the average recall, then the class will be oversampled. We set the max-value of k equal to the defined queue size, which is determined through inspection.

3.3.2.2 F1-Score-based Sampling

The F-measure [59] refers to the harmonic mean of two metrics, recall, and precision. The F-measure may be weighted depending on the value assigned to α. We used a balanced value, referred to as the F1-score, by setting $\alpha = 1$, which implies that precision and recall are assumed to carry equal weights in the metric.

$$F - measure = \frac{(1 + \alpha^2 \times recall \times precision)}{\alpha^2 \times recall + precision} \tag{3.3}$$

In our second sampling approach, we utilize the F1-score of the classes as a measure to control the number of oversampled instances. That is, if the absolute difference between the F1-score for a class and the mean F1-score is higher than the pre-defined threshold, then the samples for this class will be oversampled using the F1-score.

$$k = Poisson(F1 - score_{avg\_excluding\_c}/F1 - score_c) \tag{3.4}$$

Here, the F1-score refers to the score of each class calculated prequentially and $F1 - score_{avg\_excluding\_c}$ denotes the average F1-score of all classes excluding class c. If the F1-score of one class is lower than the average, it will be oversampled k times.

3.3.2.3 κ_m-based Sampling

In the third case, we utilize the κ_m measure to oversample. Bifet and Morales proposed the κ_m statistic for online learning in [20]; where they confirmed that this measure has advantages over accuracy and the original κ statistic [21]. The main motivation for using the κ_m statistic is when data streams are evolving, and classes are imbalanced, where we have:

$$\kappa_m = \frac{p_0 - p_m}{1 - p_m} \tag{3.5}$$

In equation 3.5, quantity p_0 refers to the current algorithm X's prequential accuracy, while p_m is the prequential accuracy of a majority-class classifier, a baseline learner that predicts the label that occurred most frequently up to now [21]. If classifier X is always correct, we conclude $\kappa_m = 1$. If its predictions are correct as often as those of a majority-class classifier, then $\kappa_m = 0$. Since the κ_m metric measures the performance agreement between the majority class classifier and classifier X, we cannot calculate it for each class. However, it is a measure that sensitively detects changes in the class distribution while automatically compensating for such changes. It may thus be used to recognize where classifier X is underperforming the baseline majority class learner. In this way, we will be able to assess when classifier X would benefit from oversampling the minority classes. Specifically, the ratio of the mean of the κ_m value divided by the last calculated κ_m value is considered as a rate parameter. If the result surpassed a threshold, the class will be oversampled M times following:

$$k = Poisson(MajorClass_{Size}/NewInstanceClass_{Size})) \tag{3.6}$$

In this equation, $MajorClass_{Size}$ refers to the number of instances in the class with the most instances seen so far, and we update each class size by the class label of each arriving instance.

3.3.2.4 Euclidian Distance-based Sampling

The last oversampling version monitors the ratio of the majority class to the class label of newly arrived instances. This parameter as per equation 3.6, defines the number of instances we need to oversample based on a distance criteria [113]. In this case, if k is greater than a defined threshold, we resample the k instances that are **most similar** to the current one. The Euclidean distance similarity score is used to determine the k most

similar instances as:

$$Euclidean\_Distance(x_c, q_c) = sqrt((x_c - q_c)^2) \tag{3.7}$$

Here, x_c refers to the most recently arrived instance with the class label c and q_c is a vector of instances inside the queue of class c. After we sort the results based on distances, the top k instances will be oversampled and inserted into the q_c. In this way, the q_c contains instances that are most similar to the x_c while disregarding the less similar ones.

3.3.3 Ensemble learning

Recall that we utilize a sliding ensemble approach, where the base learners update their models against different batches, corresponding to sliding batches, following [63]. In our algorithm, each of the base classifiers is trained independently [63]. We extended OB [123] so that, instead of training each instance k times from a Poisson distribution, we employ k to oversample the relevant queues. Figure 3.4 illustrates how we use sliding batches to update base learners. That is, for each instance in the data stream, one base learner in N is updated by the sliding batch. If we consider N learners as $N = \{p_1, p_2, ..., p_{N-1}, p_N\}$, where p_1 refers to the first learner, p_1 updates the batch at time t, p_2 updates with the batch at time $t+1$ and the P_N updates with the batch at $t+N$. Subsequently, the rotation restarts from p_1. Note that the prediction at each time step is made over all the base learners. We employed a soft voting process to determine the ensemble prediction [87]. For each arriving instance, soft voting requires each of the learners in the ensemble to produce a confidence score (within the range $[0, 1]$) for their prediction for each class value or to output the probabilities that an instance belongs to a given class label. Consequently, a simple, soft voting classifier without weighting factors, given an instance x_t, calculates the average probability for each class label over the predictions of all classifiers and determines the most probable class C as in Equation 3.8 [87]:

$$\hat{y_t} = \arg\max_{C} \sum_{j=1}^{|N|} P_j(= C \| x_t), y_t \in Y, t \in \{1, 2, ..., t^{'}\} \tag{3.8}$$

The class with the largest average probability is exported as the winner through this process, where $y_t \in Y$, and the notation of N depicts the number of the combined classifiers.

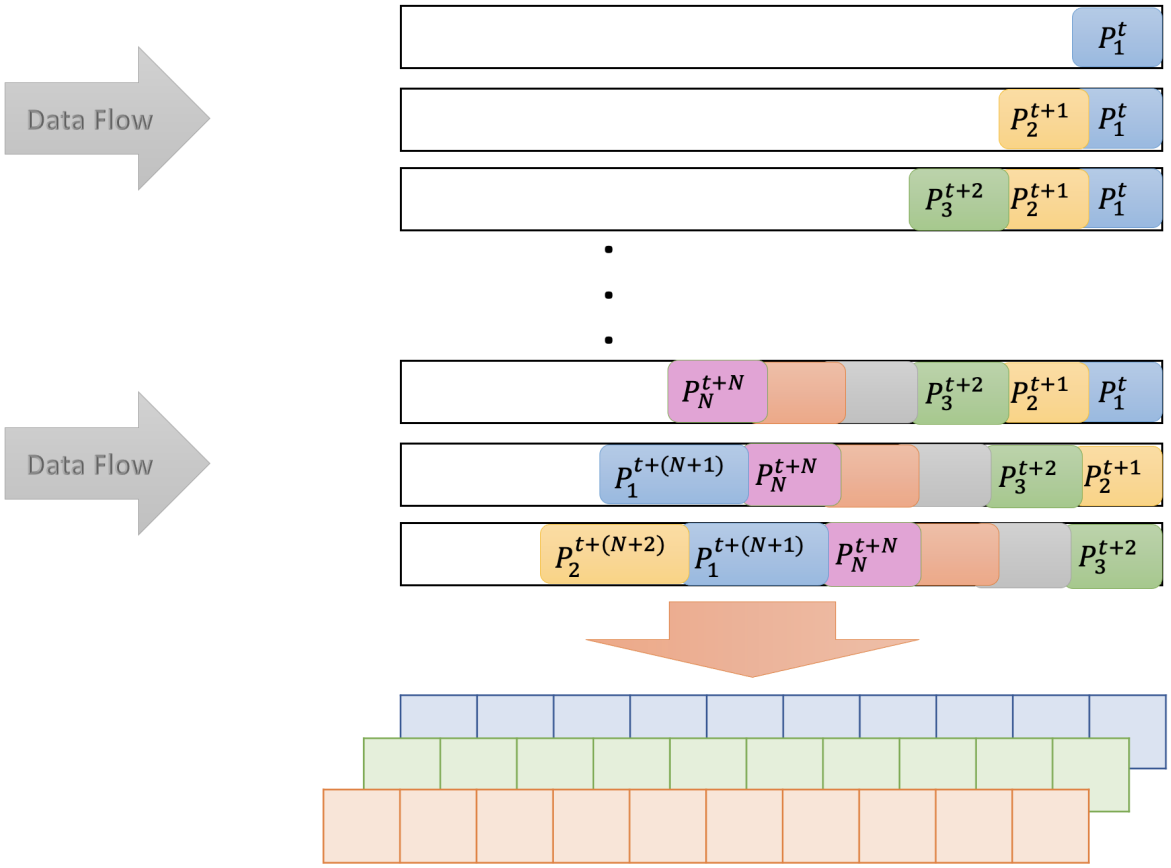


Figure 3.4: Ensemble learning of sliding batch [63]

Our ensemble is designed so that the training set of the N individual classifiers proceeds out of step, using a sliding batch for instance selection. As shown in Figure 3.4, at every time step, we append a new updated batch to the ensemble and train a single classifier p_n on that batch. For the next $N-1$ iterations, we train the remaining $N-1$ classifiers, and so on. From the ensemble point of view, it looks like we are using a sliding window to train with the differences of N time steps. However, from the point of view of each base classifier in the ensemble, we are employing sliding batches to train. Figure 3.4 illustrates this process for N classifiers in the ensemble where the difference for each is one time step, and each classifier will learn based on a batch of the same colour. Starting with p_1 with the blue batch, the next classifier is p_2 with the orange batch of time $t+1$. The next rotation will start after $N+1$ steps, as shown with the next blue batch training p_1 at time $t+(N+1)$. Intuitively, this method ensures diversity in terms of the instances used by

the individual classifiers when casting their votes.

3.3.4 Concept Drift Detection

Recall that we also included a class-based concept drift detector to handle evolving streams. To this end, we adapted the idea of the DDM drift detection algorithm in our framework [155]. The main task of a drift detector is to prompt the learner to update the model after drift occurs. The number of misclassified instances corresponding to each class is used as a drift indicator based on the results so far.

Following [155], we employ two counters for each class, where w_i denotes a warning level, and d_i denotes the drift detection threshold. That is, we continuously update w_i and d_i, and if the number of misclassified instances reaches d_i, then a drift is detected. Subsequently, a new model is induced using the examples stored between w_i and d_i. Practically, this process aids in removing outdated samples and updates the queue with new instances. Our drift detector process is initiated once an instance is misclassified, then continues until it reaches the specified proportion of the queue (denoted by L/n). The rationale behind this approach is to find a trade-off between the ability of the learner to adapt faster while not testing for drift too often to limit the overhead associated with detection. We aim to maintain only the optimal "small subset" of data necessary to accurately flag for drift. Intuitively, if $L = 1$, then the process corresponds to testing for drift as every instance arrives; that is, $n = 1$.

Algorithm 3.1 DynaQ

Define $rate\_parameter(Recall, F1score, \kappa_m, EuclideanDistance)$
while stream.has_more_instances() at each time step t **do**
    x_i^t, y_i^t = get.next_instance();
    y_i^t_predict = Ensemble.Predict(x_i^t); #Test-then-train evaluation
    $W = Update\_W_{c_i}$;
    $Drift\_list_{C_i}$ = [ ];
    $\alpha_{C_i}^t$ = TestForDrift(x_i^t, y_i^t); #Test for concept drifts
    if y^t *in previously seen class* C_i **then** #Add to queue
        Increment $Counter_{C_i}$;
        $Q_{C_i}^t =_{C_i}^{(t-1)} .append(i)$;
        Update($\alpha_{C_i}^t$);
    else
        $Counter_{C_i} = 1$;
        Initialize$_{C_i}^t$,$Q_{C_i}^t.append(i)$;
        Update($\alpha_{C_i}^t$);
    end if
    if $rate\_parameter\_ratio > threshold$ **then**
        $L[C_i] = rate\_parameter\_oversampling$;
    end if
    $k = Poisson(L[C_i])$;
    if $k > queue\_size$ **then**
        $k == queue\_size$;
    end if
    for k times **do**
        $Q_{C_i}^t.append(x_i^t)$;
    end for
    if $(\alpha_{C_i}^t \geq w_{C_i}^t)$ is TRUE **then** #Warning, Start Drift list
        Add instance i to $Drift\_list_{(C_i)}$;
    end if
    if $(\alpha_{C_i}^t \geq d_{C_i}^t)$ is TRUE **then** #Drift detected
        Update $Q_{C_i} = Drift\_list_{C_i}$;
    end if
    $Q_{C_i} = Q_{C_i}(x_h, y_h)$;
    $B_t = \bigcup_{i=2}^{n} Q_{C_i}$;
    $Base\_Learner_p = t\%N$; #Creates a rotatory updates of base learners
    Classifier($Base\_Learner_p$).Incremental.Update(B_t);
end while
Return G_Mean, F_Measure, κ, Model

Figure 3.5 shows our results against the Gas Sensor [152] and LED data stream [24], two of the repositories we used in our experiments where n was set to 2 by inspection. The reader will notice that, as expected, the drift detection threshold has a considerable influence on the predictive performance. In this setting, once a misclassification occurs, we signal a warning for potential drift and start to collect all instances from this point in time into the drift detector queues. Next, we test for drift when we reach (L/n) instances and proceed accordingly. We either detect a drift (and reset the learner) or continue to monitor. If no drift has been detected, but the warning level remains, we proceed to collect and test with the next (L/n) instances. This process continues until the set of examples is equal to our queue size L. When a drift is detected, the learner is reset, and a new model is learned using a training set consisting of all the examples in the drift detection queues maintained since the warning was triggered. It follows that the values of n and L are domain-dependent and should be carefully selected to ensure the accuracy and efficiency of the drift detector. As shown in Figure 3.5, the response to concept drift in $\alpha = (1, L)$ is later than the other two, which decreased the model performance. However, as should be expected, the reaction to the threshold values is different for both data streams. The Gas Sensor data experience more fluctuation and decrement during the stream; the LED stream is more tolerant of late drift detection for $\alpha = (1, L)$. In summary, results of $\alpha = (1, 1/2L)$ outperform the other thresholds for both streams.

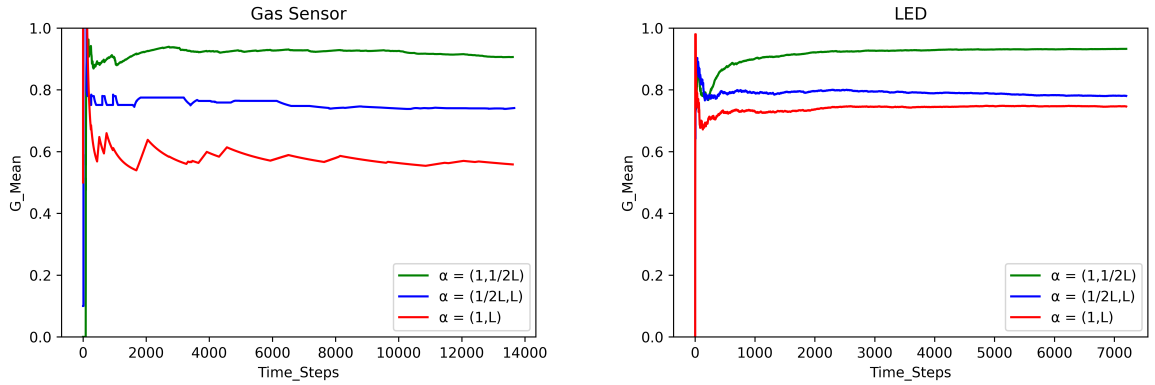


Figure 3.5: G-mean value results with different concept drift thresholds against the Gas Sensor and LED stream.

In summary, in our DynaQ algorithm, when a new instance arrives, the rate parameter is calculated. Subsequently, for minority classes, the new instance is resampled k times and appended to the appropriate queue. As explained above, we use each instance to test the model, then insert the instance into the appropriate queue. The set of queues form a batch and each batch is used by a single learner to update the model. Ensemble learning involves sliding windows and soft voting. Next, we discuss our experimental evaluation.

3.4 Experimental Evaluation

All experiments were conducted on a MacBook Pro with a Dual-Core Intel Core i5 processor, CPU @ 3.1 GHz processor, 8.0 GB RAM on the Mac Catalina Operating System (OS), and the Alliance Canada Cloud with 10 Core CPUs [8]. Our code was implemented using the Scikit-Learn [125] and Scikit-Multiflow [117] packages in Python version 3.8.2. Our competitor methods run on the MOA [24] library and we used MOA to generate synthetic streams. The framework's implementation and all the code for the experiments will be made available on GitHub upon publication. The no-change and majority-class classifiers were used as our baselines. The no-change classifier assumes that the class label of instance x_i would be the same as the last-seen instance x_{i-1}, while the majority-class learner assigns the class seen most often so far to a new instance [117]. Additionally, we employed three baseline classifiers, namely, Hoeffding adaptive tree (HAT) [23], Hoeffding tree (HT) [52], and the self-adjusting memory (SAM) model for the K-nearest neighbour (KNN), denoted by SAMKNN [104], during our ensemble learning. HTs are incremental decision trees for data stream classification that use Hoeffding's bound to commence online learning. HAT is an extension of HT that adaptively learns from data streams that change over time without needing a fixed-size sliding window. SAMKNN is an online implementation of KNN, and we set $k = 7$ by inspection. Following [150], we set the number of base learners in our ensemble to 10.

The estimation technique we use is prequential evaluation, which consists of executing a loop infinitely, where the ensemble first predicts labels for new data (without its label), then updates its model by that data with the correct label [169]. The performance measures we used are the F-measure, geometric mean (G-mean), and κ_m statistic. The F-measure [59] is macro-averaged over the sum of F1-scores over all classes, which assigns equal weights to the existing classes. Additionally, we employed the G-mean [155] value which is the geometric mean of the recall rates of majority and minority classes in the imbalanced data

set. The calculation method is as shown in 3.9.

$$G - mean = \sqrt[n]{Recall_1 \times Recall_2 ... \times Recall_n} \tag{3.9}$$

The G-mean value is higher only when the classification accuracies of the majority sample and the minority sample are high; therefore, the G-mean value can accurately the classification effect of unbalanced data sets. In addition, we utilize the previously introduced κ_m metric, to address the effect of the performance agreement between the majority class classifier and classifier X [117].

We also compare DynaQ with six state-of-the-art online multi-class learning methods, namely, OMCQ, IOE, ROSE, KUE, MOOB, and MUOB. We use inspection and grid-searching for hyper-parameter tuning to determine the optimal parameters for all methods. However, if a particular algorithm specifies certain parameters, we respect that and use them accordingly. As we introduced in 3.2.2, ROSE and KUE are driven by the κ metric. While KUE is a chunk-based general-purpose ensemble for drifting data streams [40], ROSE is an online ensemble that works with imbalanced data streams with dynamic imbalance ratio and concept drift, offering several features designed specifically to deal with these challenges [41]. All ensembles are evaluated using HTs as base learners with the same parameter settings of 10 base classifiers. These two methods employ a sliding window size of 1000 instances.

Recall that OMCQ uses a queue-based sampling strategy to keep each class separated in the queues to maintain a balanced training set. The method employs a class-based DDM with a queue-based recovery process. Following [140], the queue sizes were set by inspection. As noted in Section 3.2.2, Vafaie et al. (2019) introduced an IOE algorithm for handling multi-class imbalanced data. The approach samples with replacement and incorporates DDM-OCI [155]. DDM-OCI tracks the recall rates on the minority classes to actively locate concept drifts for imbalanced data streams. A significant drop in the recall suggests a drift in this class. After drift is detected, the model will be reset and trained based on the data received between the drift warning and drift detection. Following [150], we set the forgetting factor in recall rates to be 0.9 and the threshold of the absolute difference between the class recall and average recall equal to 0.05.

Additionally, as mentioned in Section 3.2.2, [157] introduced the MOOB and MUOB algorithms that oversample or undersample classes based on the probabilities of instances belonging to a class. That is, in MOOB, oversampling is used to increase the possibility of learning minority-class examples based on the occurrence probability of examples belonging to each class. Meanwhile, in MUOB, undersampling is used to reduce the chance of learning majority-class examples. We incorporated the DDM-OCI [155] method with MOOB and MUOB in case of concept drift to conduct a fair comparison.

We conducted four sets of experiments to assess our DynaQ algorithm. First, we studied the impact of the four different rate parameters used in order to conduct minority class oversampling. Second, we considered the impact of queue size on learning. Third, we explored the performance of DynaQ when utilizing different base classifiers in our ensemble. Forth, we considered the impact of concept drift detection on DynaQ. Finally, we contrast our DynaQ with the state of the art.

3.4.1 Data streams

Our experimental study was based on the following multi-class data sets depicted in Table 3.1: historical weather data obtained from Open Data Canada [39], the Shuttle data set from the KEEL repository [49], the LED data stream [24], the radial basis function (RBF) stream [24], the Gas Sensor stream [152], the Human Activity Recognition (HAR) stream [42], the CoverType data stream [28] and the Intel Berkley Research Lab Sensor data stream [111]. The Weather repository contains data from probes located across Canada to detect adverse weather with natural drifts. The Shuttle stream considers three classes and is used to predict when an auto-landing would be preferable to the manual control landing of a spacecraft. The LED data set comprises seven Boolean attributes and ten labels; the goal was to predict the digit displayed on a seven-segment LED display, where each attribute has a 10% noise level. We used a version of LED available through Scikit-Multiflow that includes gradual concept drifts in the stream by simply changing the attribute positions. The RBF generates a fixed number of random centroids, where each center has a random position, a single standard division (SD), a class label, and a weight. The generated RBF data sets have ten numerical attributes and 50 centers with four classes, and a change speed of 0.89 was chosen for the gradual drift in the data. To assess the impact of concept drifts on imbalanced streams, we created two RBF streams with comparable imbalance ratios but distinct concept drift patterns.

The Gas Sensor stream contains 13,610 measurements from 16 chemical sensors utilized in simulations for drift compensation in a discrimination task of six gases at various concentration levels. The HAR data set contains uncalibrated accelerometer data collected from 15 participants performing seven activities. We combined the activity of three participants to create drift in the stream. CoverType is a benchmark data set for evaluating stream classifiers that originates from the UCI repository and contains cartographic attributes for predicting forest CoverType. This data set represents a forest CoverType for 30×30m cells, where each CoverType is represented by one of the seven classes. In this domain, concept drift may appear due to weather and climate change. The Intel lab data is collected from 54 sensors deployed in the Intel Berkeley Research lab between February 28th and

April 5th, 2004. Mica2Dot sensors with weatherboards collected timestamped topology information and humidity, temperature, light, and voltage values once every 31 seconds. Data was collected using the TinyDB in-network query processing system, built on the TinyOS platform. We used the data instances from 15 sensors to produce an imbalanced data stream.

Table 3.1: Data streams and their properties.

Data set	Size	Number of Class	Class Imbalance Ratio	Drift Type
Weather	29,375	4	1:2.5:1.5:13	Abrupt
Shuttle	2167	3	1:5:13	No drift
LED	7205	4	1:1.5:2.7:5.7	Gradual and Noise
RBF1	50,000	4	1:1:1:2	Gradual
RBF2	40,000	5	1:2:4:10:20	Gradual
RBF3	40,000	5	1:2:4:10:20	Abrupt
Gas Sensor	13,610	4	1:1:1:4.4	Abrupt
HAT	35,300	4	1:1:1.7:6.3	Gradual
CoverType	42,000	7	1:1:1:1:1:5:13	Abrupt
Intel Sensor	50000	15	1,1,1,1,1,2,2,3,3,3,6,6,6,10,10	Abrupt

3.4.2 Experimental results

In our first set of experiments, we investigate the use of the four different DynaQ oversampling metrics as introduced in Section 3.3.2, as depicted in 3.2. In this set of experiments, we employ the HT classifier and report the G-mean, F-measure and κ_m evaluation metrics. Table 3.2, shows that the DynaQ variant using recall produced the highest values for four out of seven data streams. When considering the individual metrics, the reader will notice that the results for DynaQ with F1-score are highest for the Weather and HAR streams. In the case of the CoverType data set, the variant of DynaQ using G-mean and κ_m metrics yields the highest results. Future, we consider the statistical significance of the results using the Nemenyi post-hoc test with $\alpha = 0.05$, as depicted in Figure 3.6. Figure 3.6 illustrates that oversampling using recall and F1-score, results in similar performance, with the recall-based sampling ranking first. This result indicates that paying close attention to the true positive rates clearly benefits learning. As a result, we utilize the DynaQ variant with recall-based sampling in our subsequent experiments.

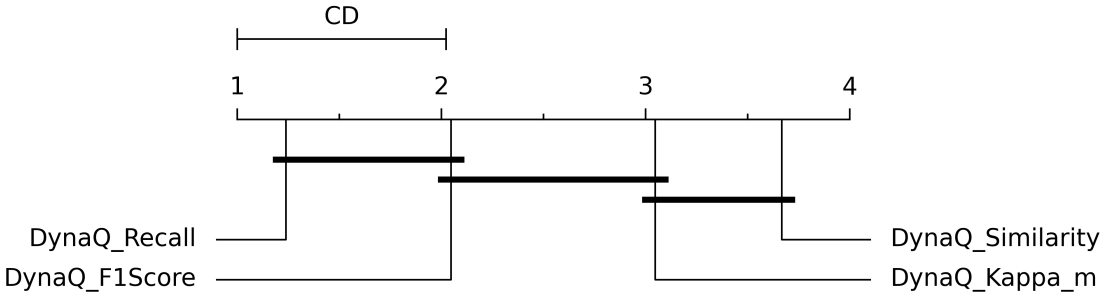


Figure 3.6: Nemenyi graph ranking HT base classifier for various sampling methods.

Second, we investigated the effect of queue size on our DynaQ algorithm to assess how the size of L affects the performance of queue-based learning. Figure 3.7 depicts the behaviour of the proposed method on different sizes of queue $L \in \{1, 10, 20, 30, 50\}$. As expected, the figure shows that the smaller the queue length, the faster the learning speed, although the results might be different with longer queues. For most data sets, this number is highly domain-dependent, and it should be set according to the characteristics of each data stream. In online learning, there is an obvious interplay between accuracy and learning time. Our results indicate that, for our experiments, a queue size of 10 resulted in a good trade-off between accuracy and speed for the Shuttle, Weather and Gas Sensor data sets. A queue size of 20 produced good results for RBF and LED, but CoverType and HAR worked better with a queue size of 50. We subsequently report the results of using a queue size of 50 against all data sets. It is noticeable that the queue size does not depend on the size of the data set.

Table 3.2: Evaluation of different versions of DynaQ against data streams.

Hoefding Tree					
Data	Metric	DynaQ_Recall	DynaQ_F1Score	DynaQ_κ_m	DynaQ_Similarity
Shuttle	G-mean	**0.971**	0.958	0.871	0.828
	F-measure	**0.952**	0.916	0.849	0.805
	κ_m	**0.837**	0.742	0.783	0.582
Weather	G-mean	0.862	**0.863**	0.750	0.711
	F-measure	0.851	**0.861**	0.726	0.732
	κ_m	0.821	0.763	0.734	0.505
LED	G-mean	**0.943**	0.892	0.788	0.781
	F-measure	**0.901**	0.869	0.830	0.727
	κ_m	**0.693**	0.660	0.639	0.555
RBF	G-mean	0.878	**0.894**	0.863	0.801
	F-measure	0.890	**0.901**	0.848	0.790
	κ_m	0.675	**0.698**	0.641	0.650
Gas Sensor	G-mean	**0.883**	0.855	0.775	0.780
	F-measure	**0.833**	0.828	0.782	0.699
	κ_m	**0.822**	0.794	0.706	0.681
HAR	G-mean	**0.904**	0.894	0.875	0.725
	F-measure	**0.801**	0.798	0.708	0.763
	κ_m	**0.701**	0.678	0.602	0.635
CoverType	G-mean	0.915	0.908	**0.923**	0.827
	F-measure	**0.858**	0.844	0.850	0.768
	κ_m	0.711	0.648	**0.751**	0.638
Intel Sensor	G-mean	0.878	**0.894**	0.863	0.801
	F-measure	0.890	**0.901**	0.848	0.790
	κ_m	0.675	**0.698**	0.641	0.650

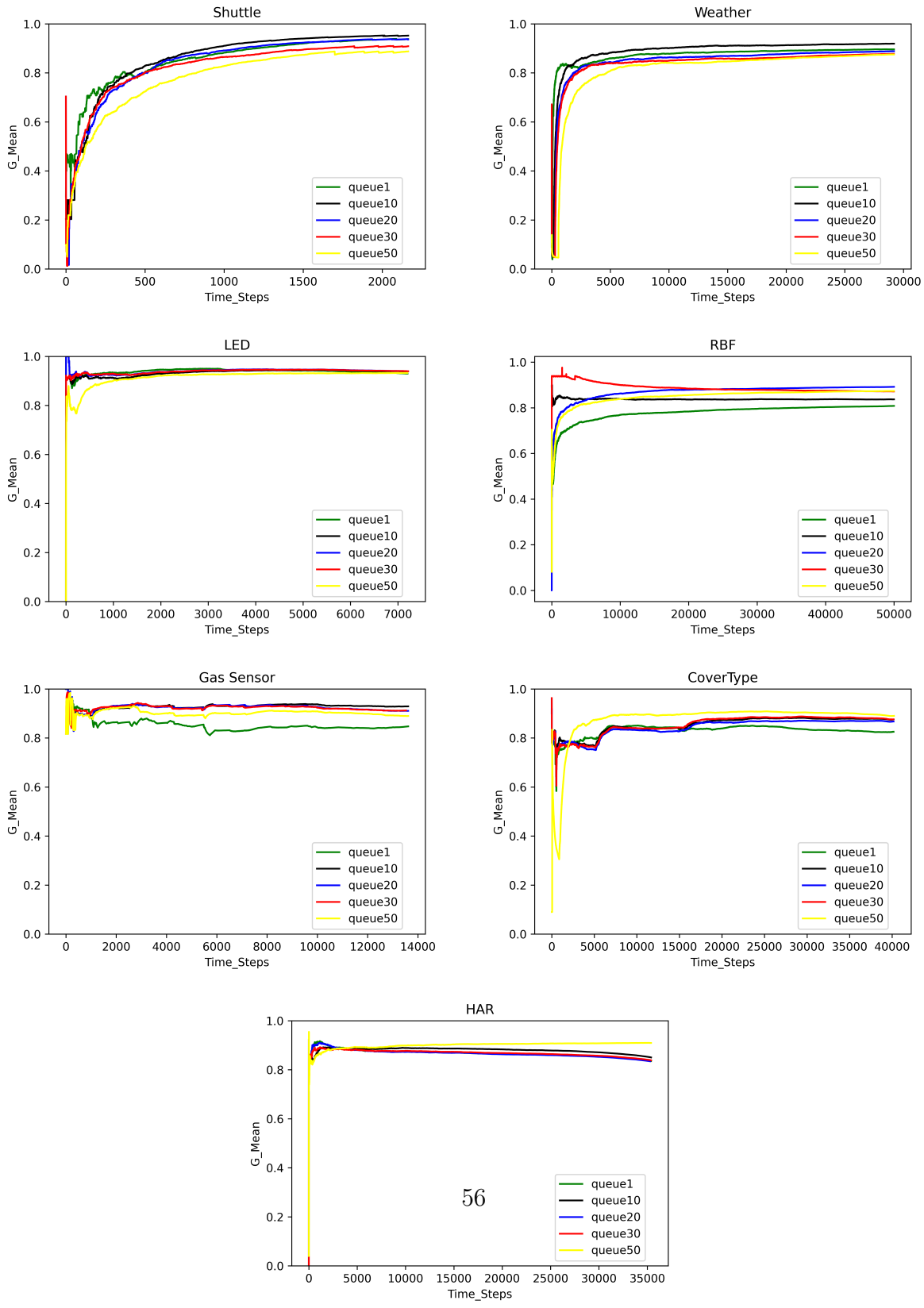


Figure 3.7: Performance comparison against different queue sizes on DynaQ.

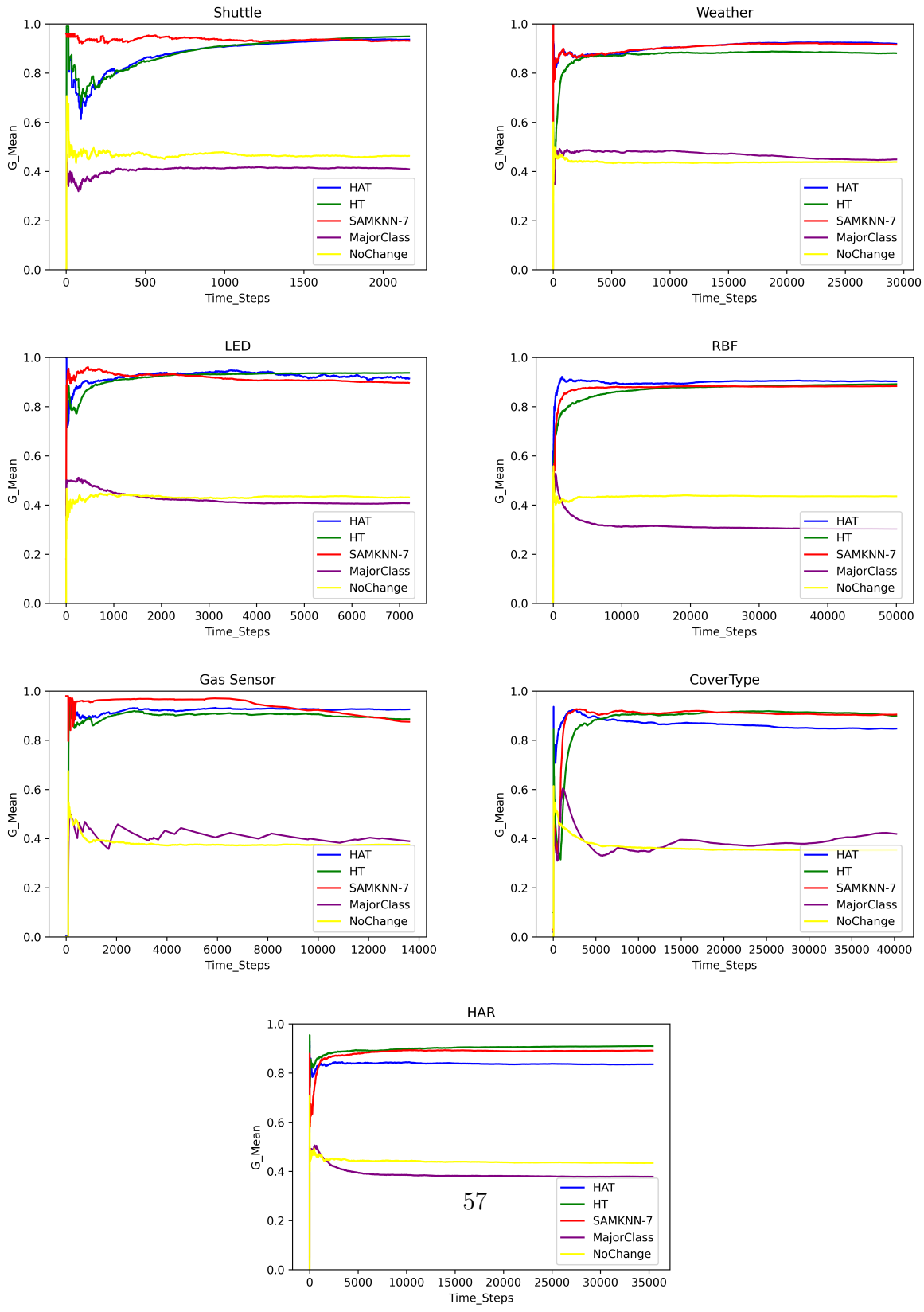


Figure 3.8: Results of DynaQ algorithm for different base learner when compared with baselines.

Next, we turn our attention to the second set of experiments, where we utilise various base learners in our DynaQ algorithm. Figure 3.8 depicts the G-mean results for our data sets when assessing the performance of the DynaQ technique compared to the two baseline algorithms. The results clearly show the benefit of our DynaQ algorithm compared to the majority-class and no-change learners, which could not learn the concepts within our multi-class imbalanced streams. The HT ensemble reaches higher performance (in Shuttle by 1%, in LED by up to 4%, in CoverType by 6%, and in HAR by 7%) than the HAT or SAMKNN ensembles. However, for the Weather, RBF, and Gas Sensor data sets, the HAT ensemble presents up to 4% better results than ensembles based on HT or SAMKNN. Recall that HT and HAT are both incremental tree learners. However, HAT is more adaptable with streaming data because it uses an adaptive sliding window (ADWIN) algorithm as a drift detector error estimator and requires no parameters related to change control. We conclude that those data sets that work slightly better with HAT ensembles may include gradual or abrupt concept drift while streaming. Readers should notice that fast drift detection and the associated recovery process prevented a significant performance drop. Our results also indicate that the baseline no-change and majority-class learners produced low values among all data sets.

In the next set of experiments, we investigate the effect of the queue-based concept drift detection method on the DynaQ learning process. As expected, Figure 3.9 indicates that the G-mean is higher when we include the DDM mechanisms during learning. This result suggests that because the queue-based sampling keeps the recent concepts of the stream, the use of drift detection clearly benefits the learning process. The fluctuation and recovery periods are different for various types of concept drifts. In the related visualization graphs, the reader will notice that the weather data experience abrupt drift while LED data show gradual drift. With drift detection, the performances against these two streams do not drop drastically, but without drift detection, the learner is not able to recover and reach back to higher values after the drift happens. In this case, employing a DDM clearly prevents performance degradation. From Figure 3.9, we find that RBF and HAR, which are susceptible to gradual drifts, do not suffer a drastic reduction in their total predictive accuracies. However, there are clear fluctuations related to gradual concept drifts that are handled by the DDM. The Gas Sensor and CoverType data sets encounter sudden performance decrement and variation among the stream caused by abrupt drifts. However, with the DDM, these changes recovered quickly without harming performance.

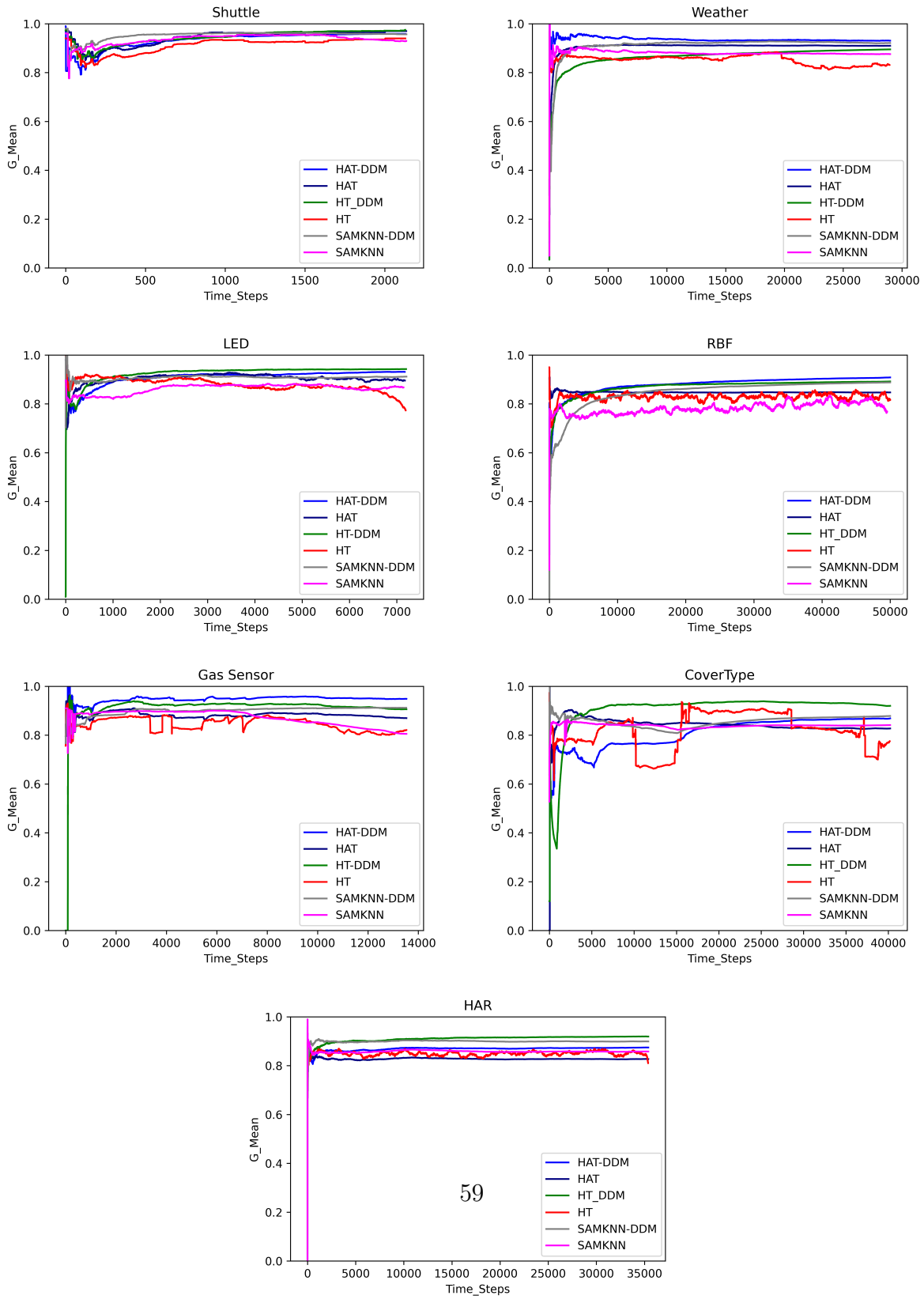


Figure 3.9: Results with and without concept drift detection for DynaQ ensembles.

Table 3.3: HT results against data streams.

Hoefding-Tree								
Data	Metric	DynaQ	OMCQ	IOE	MOOB	MUOB	ROSE	KUE
Shuttle	G-mean	**0.971**	0.951	0.901	0.838	0.800	0.955	0.895
	F-measure	**0.952**	0.942	0.849	0.815	0.764	0.802	0.782
	κ_m	**0.837**	0.742	0.683	0.483	0.472	0.822	0.813
Weather	G-mean	0.862	0.857	0.840	0.791	0.787	**0.871**	0.843
	F-measure	**0.851**	0.831	0.766	0.752	0.749	0.796	0.770
	κ_m	0.821	0.713	0.714	0.605	0.601	**0.824**	0.777
LED	G-mean	**0.943**	0.912	0.782	0.780	0.777	0.940	0.893
	F-measure	**0.901**	0.888	0.890	0.747	0.710	0.867	0.832
	κ_m	**0.693**	0.560	0.659	0.455	0.495	0.664	0.649
RBF	G-mean	**0.878**	0.844	0.843	0.701	0.617	0.823	0.811
	F-measure	**0.890**	0.819	0.828	0.590	0.476	0.729	0.713
	κ_m	**0.675**	0.615	0.601	0.450	0.325	0.624	0.630
RBF2	G-mean	**0.825**	0.784	0.762	0.692	0.688	0.812	0.800
	F-measure	**0.818**	0.769	0.715	0.630	0.621	0.799	0.782
	κ_m	**0.638**	0.598	0.544	0.568	0.487	0.602	0.652
RBF3	G-mean	**0.811**	0.782	0.723	0.800	0.719	0.767	0.738
	F-measure	**0.794**	0.724	0.751	0.732	0.689	0.751	0.722
	κ_m	0.608	0.583	0.512	0.534	0.492	**0.615**	0.565
Gas Sensor	G-mean	**0.883**	0.860	0.815	0.740	0.550	0.710	0.694
	F-measure	**0.833**	0.833	0.760	0.587	0.414	0.683	0.603
	κ_m	**0.822**	0.780	0.786	0.511	0.495	0.792	0.778
HAR	G-mean	**0.904**	0.834	0.872	0.825	0.830	0.730	0.714
	F-measure	**0.801**	0.781	0.778	0.743	0.652	0.716	0.668
	κ_m	**0.701**	0.641	0.666	0.655	0.612	0.671	0.663
CoverType	G-mean	0.915	0.888	0.793	0.627	0.582	**0.919**	0.784
	F-measure	**0.858**	0.844	0.667	0.661	0.387	0.816	0.751
	κ_m	0.711	0.618	0.651	0.438	0.291	**0.726**	0.715
Intel Sensor	G-mean	0.723	0.64	0.58	0.66	0.51	**0.730**	0.624
	F-measure	**0.591**	0.508	0.543	0.509	0.484	0.523	0.518
	κ_m	**0.433**	0.421	0.409	0.375	0.333	0.419	0.420

Finally, we present our results when contrasting different online multi-class approaches. Specifically, Tables 3.3-3.6 present the results of our comparative study contrasting the DynaQ, ROSE, KUE, OMCQ, IOE, MOOB, and MUOB algorithms. Table 3.3 includes results on all methods based on HT as the base classifier. The remainder of the tables present results on base-classifier agnostic methods with three component classifiers (HT, HAT, and SAMKNN) and three evaluation metrics (G-mean, F-measure and $Kappa_m$).

Table 3.3 shows that DynaQ outperforms all approaches over different measures. Although DynaQ and ROSE produce competitive results, DynaQ yields better results in 25 out of 30 cases than ROSE. However, ROSE performs slightly better on the Weather, RBF3, and CoverType streams. This result may be attributed to the abrupt drifts that these three data streams experience, where the background classifiers utilized by the ROSE algorithm facilitate the learning process. Table 3.4 shows that the DynaQ algorithm produced the highest values in terms of G-mean for 18 out of the 21 cases. The results also indicate that the three base learners produce comparable results in terms of G-mean, while no single base learner consistently outperforms the other two in all setting. However, the reader will notice that there are often very large differences in terms of G-mean, e.g., for LED and CoverType, when contrasting the queue-based algorithms with IOE, MOOB and MUOB.

A similar observation holds for the F-measure depicted in Table 3.5, where DynaQ produced the highest results for all data streams and base learners. The design of DynaQ implies that when there is no wait time for each queue to commence training the model, and the queues are resampled with the most recent data based on recall rates. Thus, the algorithm allows for better generalization of the incoming stream. Recall that OMCQ maintains queues that only include original instances. In the case of highly imbalanced streams, the minority queues may still contain instances representing the concept prior to drift, thus leading to a degradation in performance. The IOE technique also provides competitive results with DynaQ, which suggests the benefit of balancing performance based on the recall parameter. However, the results suggest that the undersampling of the majority instances do not benefit learning. The superior results of DynaQ are especially evident for the LED, RBF, Gas Sensor and CoverType data streams, which contained severe abrupt or gradual concept drifts. The MOOB and MUOB algorithms struggled to obtain high values against such evolving streams.

Table 3.4: G-mean results against data streams.

		G-mean				
Data	Classifier	DynaQ	OMCQ	IOE	MOOB	MUOB
Shuttle	HAT	**0.964**	0.950	0.957	0.867	0.826
	HT-Tree	**0.971**	0.951	0.901	0.838	0.800
	SAMKNN-7	0.963	0.968	**0.970**	0.896	0.845
Weather	HAT	**0.905**	0.854	0.825	0.785	0.790
	HT-Tree	**0.862**	0.857	0.840	0.791	0.787
	SAMKNN-7	**0.890**	0.880	0.876	0.839	0.771
LED	HAT	**0.935**	0.921	0.898	0.784	0.789
	HT-Tree	**0.943**	0.912	0.782	0.780	0.777
	SAMKNN-7	0.908	**0.911**	0.856	0.767	0.792
RBF	HAT	**0.886**	0.882	0.806	0.620	0.645
	HT-Tree	**0.878**	0.844	0.843	0.701	0.617
	SAMKNN-7	**0.871**	0.861	0.823	0.777	0.643
Gas Sensor	HAT	**0.920**	0.866	0.901	0.619	0.640
	HT-Tree	**0.883**	0.860	0.815	0.740	0.550
	SAMKNN-7	**0.880**	0.874	0.824	0.764	0.757
HAR	HAT	0.835	**0.839**	0.827	0.806	0.796
	HT-Tree	**0.904**	0.834	0.872	0.825	0.830
	SAMKNN-7	**0.887**	0.883	0.867	0.839	0.767
CoverType	HAT	**0.857**	0.844	0.848	0.835	0.772
	HT-Tree	**0.915**	0.888	0.793	0.627	0.582
	SAMKNN-7	**0.917**	0.909	0.868	0.818	0.800

Table 3.5: F-measure results against data streams.

		F-measure				
Data	Classifier	DynaQ	OMCQ	IOE	MOOB	MUOB
Shuttle	HAT	**0.944**	0.945	0.936	0.833	0.796
	HT-Tree	**0.952**	0.942	0.849	0.815	0.764
	SAMKNN-7	**0.967**	0.964	0.947	0.879	0.723
Weather	HAT	**0.906**	0.838	0.785	0.733	0.750
	HT-Tree	**0.851**	0.831	0.766	0.752	0.749
	SAMKNN-7	**0.874**	0.868	0.745	0.786	0.694
LED	HAT	**0.926**	0.902	0.905	0.756	0.761
	HT-Tree	**0.901**	0.888	0.890	0.747	0.710
	SAMKNN-7	**0.910**	0.893	0.862	0.699	0.754
RBF	HAT	**0.877**	0.854	0.830	0.478	0.494
	HT-Tree	**0.890**	0.819	0.828	0.590	0.476
	SAMKNN-7	**0.858**	0.841	0.836	0.721	0.547
Gas Sensor	HAT	**0.901**	0.826	0.836	0.640	0.544
	HT-Tree	**0.833**	**0.833**	0.760	0.587	0.414
	SAMKNN-7	**0.889**	0.865	0.818	0.710	0.737
HAR	HAT	**0.778**	0.745	0.771	0.674	0.640
	HT-Tree	**0.801**	0.781	0.778	0.743	0.652
	SAMKNN-7	**0.819**	0.811	0.772	0.704	0.655
CoverType	HAT	**0.761**	0.728	0.730	0.715	0.661
	HT-Tree	**0.858**	0.844	0.667	0.661	0.387
	SAMKNN-7	**0.863**	0.846	0.782	0.682	0.641

Regarding the κ_m statistical results presented in Table 3.6, the DynaQ and IOE algorithms yielded comparable values, with DynaQ in the first place and IOE in second place. In the case of the LED and RBF data sets, employing SAMKNN results in κ_m values up to 10% higher than HT and HAT. The reader will notice that recall balancing causes improvement on synthetic streams using SAMKNN while the performance decreases drastically in the HAR and Weather data sets. For the CoverType, Gas Sensor and Weather data sets, even though they experience abrupt drifts that may change the behaviour of major classes, our results are up to 8% higher than other methods. In DynaQ, we do not make any assumptions about separating majority and minority classes. This aids DynaQ to reach promising κ_m performance. In our overall analyses, DynaQ using HT as a base

learner performs better than, or similar to HAT, as a base learner. Also, the comparison between MOOB and MUOB shows that oversampling benefits the method more than undersampling. This observation further reinforces our design choice of using an oversampling approach.

Table 3.6: κ_m results against data streams.

		κ_m				
Data	Classifier	DynaQ	OMCQ	IOE	MOOB	MUOB
Shuttle	HAT	**0.825**	0.715	0.738	0.430	0.391
	HT-Tree	**0.837**	0.742	0.683	0.483	0.472
	SAMKNN-7	**0.819**	0.713	0.760	0.660	0.531
Weather	HAT	**0.784**	0.780	0.746	0.715	0.714
	HT-Tree	**0.821**	0.713	0.714	0.605	0.601
	SAMKNN-7	**0.642**	0.613	0.642	0.340	0.321
LED	HAT	**0.662**	0.616	0.638	0.450	0.454
	HT-Tree	**0.693**	0.560	0.659	0.455	0.495
	SAMKNN-7	**0.789**	0.726	**0.788**	0.392	0.445
RBF	HAT	**0.681**	0.618	0.612	0.460	0.338
	HT-Tree	**0.675**	0.615	0.601	0.450	0.325
	SAMKNN-7	**0.733**	0.710	**0.732**	0.525	0.482
Gas Sensor	HAT	**0.826**	0.756	0.760	0.516	0.500
	HT-Tree	**0.822**	0.780	0.786	0.511	0.495
	SAMKNN-7	**0.816**	0.754	0.795	0.679	0.654
HAR	HAT	**0.685**	0.618	0.654	0.581	0.427
	HT-Tree	**0.701**	0.641	0.666	0.655	0.612
	SAMKNN-7	**0.566**	0.514	0.510	0.483	0.464
CoverType	HAT	**0.684**	0.611	0.627	0.385	0.310
	HT-Tree	**0.711**	0.618	0.651	0.438	0.291
	SAMKNN-7	**0.718**	0.637	0.648	0.580	0.339

Next, we present the results of the Nemenyi posthoc test [85] shown in Figures 3.10–3.13, where β is set to 0.05. This test highlights the contrasts in the algorithms against all data sets, where a lower rank means better predictive performance (G-mean, F-measure and $kappa_m$). In Figure 3.10 DynaQ is ranked first, followed by ROSE, OMCQ, and IOE. While ROSE benefits from combining backup ensembles, DynaQ provides a stable backup

in case of drifts by allocating small-sized queues to each class. For this reason, DynaQ delivers consistent performance and high ranks across all data streams.

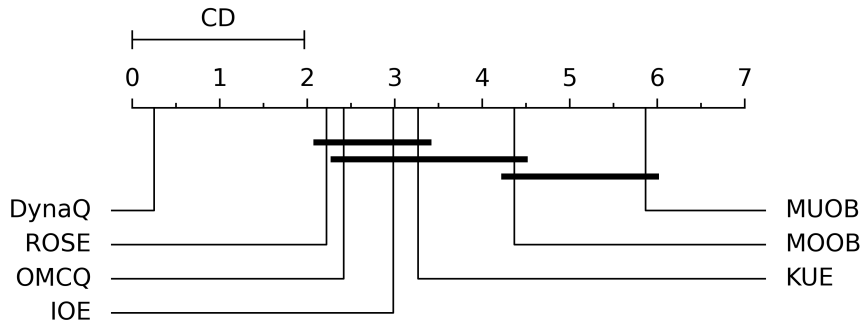


Figure 3.10: Nemenyi graph ranking algorithms based on HT base classifier.

Figures 3.11 and 3.12 show a critical difference between our DynaQ algorithm and the IOE, MOOB and MUOB techniques, for the F-measure and G-mean metrics. The reader will notice that there are no significant statistical differences between the DynaQ and OMCQ methods. Both the DynaQ and OMCQ methods clearly benefit from their underlying queue-based learning processes. However, our DynaQ method ranks first and OMCQ second. This ranking indicates the strength of combining queue-based learning with minority-class oversampling and online ensemble learning.

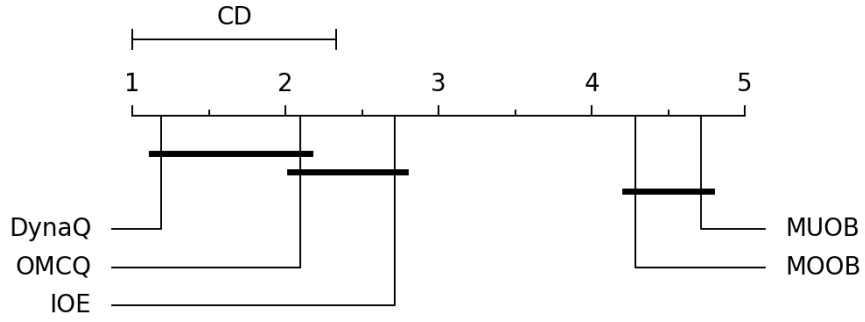


Figure 3.11: Nemenyi graph ranking G-mean results among various algorithms.

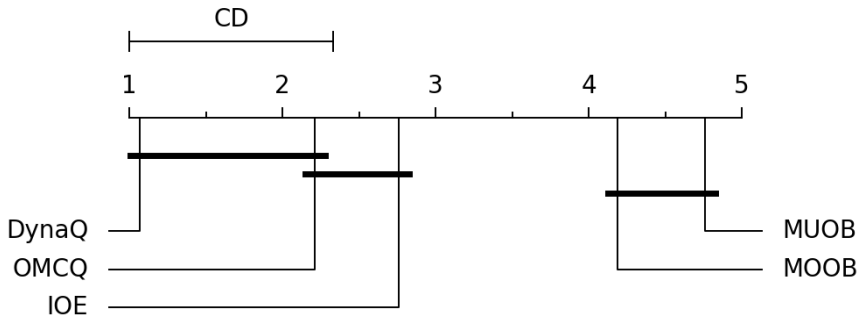


Figure 3.12: Nemenyi graph ranking F-measure results among various algorithms.

Figure 3.13 indicates that DynaQ and IOE present similar κ_m values, with DynaQ ranked first while outperforming OMCQ, MOOB and MOUB. The DynaQ and IOE algorithms both utilise recall rates that aid the learners to handle the change in class labels caused by an evolving and skewed stream. The main difference between these two approaches is that DynaQ employs oversampling, while IOE combines oversampling and undersampling. Since DynaQ is ranked first, one may conclude that undersampling is unnecessary in most setting. The results further indicate the value of balancing recall rates to improve the performance, when focusing on the $kappa_m$ metric, in a multi-class imbalanced settings.

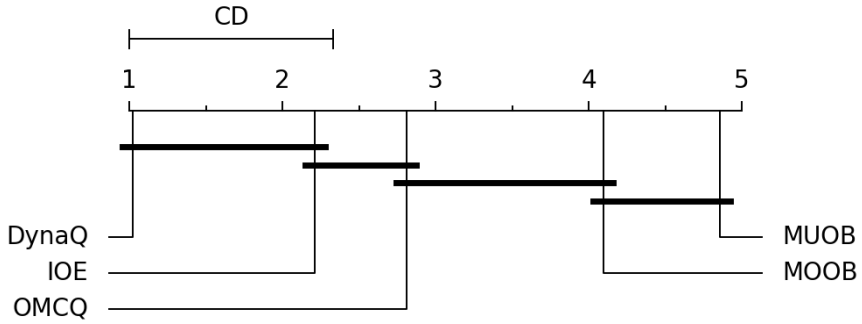


Figure 3.13: Nemenyi graph ranking $Kappa_m$ for various algorithms.

In summary, our experimental evaluations indicate the strength of combining a queue-based approach with dynamic minority class oversampling, concept drift detection, and online ensemble learning.

3.5 Summary

In this chapter, we introduce the DynaQ framework for online multi-class learning. The novelty of our approach lies in the fact that we combined class-based queues, dynamic oversampling of minority classes, online ensembles based on sliding windows, and class-based concept drift detection. An advantage of the DynaQ method is that it operates independently of a base classifier, thus providing a general framework for dealing with evolving multi-class streams. Our experimental results illustrate the value of our contribution when compared to a number of benchmarking datasets.

In the next chapter, we turn our attention to fairness-aware online learning. In addition, we introduce our multi-sensitive approach for adaptive fairness-aware learning from more than one protected group.

Chapter 4

MQ-OFL: Multi-Sensitive Queue-based Online Fair Learning

The work presented in this chapter is based on our paper: *Farnaz Sadeghi and Herna Viktor, "MQ-OFL: Multi-Sensitive Queue-based Online Fair Learning", the 25th International Conference on Discovery Science (DS 2022), Montpellier, France, October 10-12, Lecture Notes in Computer Science (LCNS) 13601, pp. 271-285, 2022.*

In Chapter 3, we introduced the DynaQ approach for multi-class online learning while considering a skewed data stream. In this chapter, we focus our attention on a complementary perspective of machine learning systems, referring to discrimination-free or fairness-aware approaches. To this end, we introduce a framework that is able to deal with imbalanced data, while considering streams that contain two or more sensitive attributes. The novelty of our work is twofold. First, our algorithm is designed for the online learning setting. Second, our algorithm considers more than one group of sensitivity.

This chapter is organized as follows. Section 4.1 provides a brief introduction and Section 4.2 discusses related work, focusing on the main fairness concepts considered in the research. In Section 4.3, we introduce a fairness-aware online algorithm. Section 4.4 details our experimental evaluation while Section 4.5 provides a summary.

4.1 Introduction

As noted in the Introduction and Chapter 2, AI applications have become a necessity to deliver all sorts of decisions. However, unconsciously, these automated data-driven sys-

tems may lead to discrimination against particular groups of people sharing one or more sensitive attributes (e.g., marital status, age, gender or sex, and ethnicity) [114]. As a recent example, Howard et. al. [79] discusses examples of how bias in the real world can breach into AI systems, such as bias in face recognition applications, voice recognition, and search engines. As a result, a number of studies have been proposed to address this concern. A common theme amongst all these prior works is the assumption of fairness as a static problem, which means the inappropriate discriminate correlations (e.g., marital status or age) is implicitly modeled as a constant and static property. Learning from the data stream assumes that new instances arrive continuously and that their properties may change over time due to a phenomenon known as concept drift [106]. Frequently, in a supervised learning setting, such streams are subject to data skew, i.e., class imbalanced, with a disproportion of the number of examples of the different classes [34]. Additionally, the vast majority of the algorithmic fairness literature focused on the simplest case where there is only one sensitive attribute [114]. To the best of our knowledge, this is the first work that jointly considers non-stationary imbalanced data distributions where there are more than one sensitive attribute.

The contribution of this chapter is three-fold. First, we define a new problem of fairness-aware learning in imbalanced data streams with more than one protected group. Subsequently, we propose a discrimination-aware pre-processing method to handle the trade-off between fairness and accuracy. Second, we introduce an approach to pre-process multi-sensitive attributes that satisfies fairness constraints. Thirdly, our experimental evaluation illustrates the capability of the proposed model in online settings and for application-driven fairness-aware learning.

4.2 Background

This section provides an overview of related work and offers definitions for key concepts associated with fairness-aware online learning.

4.2.1 Related Work

Recall from Chapter 2 that a number of research approaches have been proposed to address the problem of bias and discrimination in machine learning systems. As noted in Chapter 2, they may be categorized into three main groups, namely pre-processing approaches, in-processing approaches, and post-processing approaches, based on whether they mitigate bias at the data level, the algorithm design or the output of a model, respectively [114]. The

first strategy works under the assumption that in order to learn a fair classifier, the training data should be discrimination-free [83]. In-processing techniques modify and change learning algorithms to limit discrimination [114]. The last category consists of either adjusting the decision boundary [62] of a model or directly changing the prediction labels.

Fairness in an online setting requires simultaneously taking the evolution of underlying data distribution into consideration. Recall from Chapter 2 that the Fairness-Aware Hoeffding Tree (FAHT) [166], addresses discrimination by incorporating discrimination-awareness into the model induction process. This is accomplished by introducing the fair information gain splitting criterion, which is able to maintain a moderate predictive performance with low discrimination scores over the course of the stream. Another work [165], also introduced in Chapter 2, proposed a Fairness-Enhancing and concept-Adapting Tree (FEAT) with an embedded fair-enhancing splitting criterion. A strength of this approach is the ability of change detection and concept forgetting to handle discriminated and non-stationary data streams. However, these two methods do not consider imbalanced data.

Specifically, none of the current online fairness-aware techniques are able to simultaneously handle more than one sensitive attribute. Our work situates in this highly under-explored research direction by including multi-protected groups to provide fair online decision-making. Next, we discuss definitions and key concepts in the fairness-aware domain.

4.2.2 Fairness Definitions

We assume an attribute S, referred to as a sensitive attribute with a special value $s \in dom(S)$ which is a sensitive value that defines the discriminated group. Also, we assume that S is a binary attribute: $dom(S) = \{s, \bar{s}\}$. As an example, we use $S = "maritalstatus"$ as the sensitive attribute and $\bar{s} = "single"$ as the sensitive value (protected group) with $s = "married"$ (non-protected group). We also consider the class is binary with values {0 as rejected, 1 as granted}.

Fairness definitions fall under different types such as individual, group and subgroups [114]. The individual notion of fairness means "similar individuals" have to be "treated similarly", while the group notion refers to treating different groups equally, and subgroup fairness focuses on the best properties of the group and individual notions of fairness. It chooses a group fairness notion and finds whether this metric satisfies a large collection of subgroups [88, 89]. Here we define Statistical Parity (S.P.) as which is a group notion for fairness. S.P. measures whether the probability of having a positive outcome is the same for both protected and non-protected groups. Also, we can refer to it as a difference in the probability of a random individual drawn from the non-protected group to be predicted as

granted (positive) and the probability of a random individual drawn from the protected group to be predicted as granted:

$$S.P. = P(f(x) = y^+ \mid \bar{z}) - P(f(x) = y^+ \mid z) \tag{4.1}$$

The SP values lie in the $[-1, 1]$ range, with 0 meaning the decision does not depend on the sensitive value (meaning fair), 1 meaning that the protected group is discriminated against, and -1 that the non-protected group is discriminated. The next section introduces the concept of Gerrymandering, where we consider more than one sensitive attribute.

4.2.3 Gerrymandering

The most straightforward setting in fairness is the independent case, with only one sensitive attribute, which can take multiple values, e.g., age only. The presence of multiple sensitive attributes (e.g., ethnicity and age simultaneously) leads to non-equivalent definitions of group fairness. For example, consider a model restricted to S.P. between subgroups defined by ethnicity. Simultaneously, the model can be constrained to S.P. between subgroups defined by gender. We term fairness in this situation independent group fairness. On the other hand, one can consider all subgroups defined by intersections of sensitive attributes (e.g., ethnicity and gender, ethnicity and age, age and gender, and so on), leading to intersectional group fairness. A given algorithm can be independently group fair, e.g., when considering age and gender in isolation, but not intersectionally group fair, e.g., when considering intersections of age and gender groups. For example, [38], showed how facial recognition software had particularly poor performance for black women. This phenomenon, called fairness gerrymandering, has been studied by [88], where the authors specifically focus on ethnicity and gender.

As shown in Figure 4.1 [88], imagine a setting with two binary features, corresponding to ethnicity (say blue and green) and gender (say men and women), both of which are distributed independently and uniformly at random in a population. Consider a classifier that labels an example positive if and only if it corresponds to a blue man or a green woman. Then the classifier will appear to be equitable when one considers either protected attribute alone, in the sense that it labels both men and women as positive 50 percent of the time, and labels both blue and green individuals as positive 50 percent of the time. However, as pointed out by [88] if one considers any conjunction of the two attributes (such as blue women), then it is apparent that the classifier maximally violates the statistical parity fairness constraint. Similar examples for classification are easily constructed. We remark that the issue raised by this toy example is not merely hypothetical. To avoid

this issue, we would like to satisfy a fairness constraint for more than one protected group defined by multiple sensitive attributes.

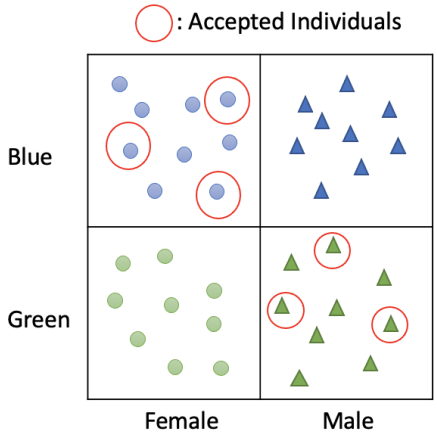


Figure 4.1: Gerrymandering illustration (from [88])

Next, we turn our attention to fairness-aware learning from evolving, imbalanced streams that contain multiple sensitive attributes.

4.2.4 Imbalanced and Drifted Data Streams

Recall that our study assumes that the underlying stream distribution is non-stationary, that is, the characteristics of the stream might change with time leading to changes in the joint distribution so that the decision boundary might change over time for two instances i, j, it might hold that $P_i(x, y) \neq P_j(x, y)$, a phenomenon called concept drift [106]. Numerous algorithms have been developed in order to detect and handle concept drift [122] as discussed in Chapter 2. Recall that Hoeffding's inequality-based Drift Detection Method (HDDM) proposed by [122] employs Hoeffding's inequality [76] to set an upper bound to the level of difference between error rates. That is, using Hoeffding's inequality, triggers a warning level to indicate a drift may have occurred. The threshold used to trigger the warning level is a relaxed condition of the threshold used for the drift level. The data accumulated between the warning level and the drift levels are used as the training set for updating a learning model. We employ HDDM in our work.

Apart from the occurrence of concept drifts, recall that we also assume that the stream

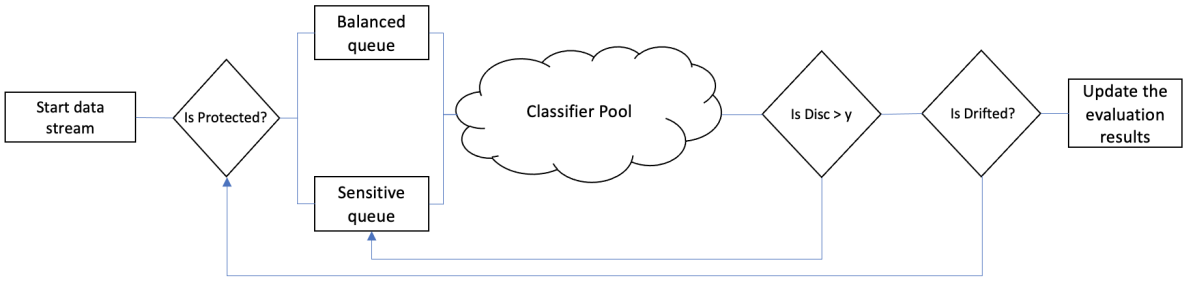


Figure 4.2: High-level overview of Queue-based fairness aware methodology

is imbalanced. In two-class problems, the minority (underrepresented) class is usually referred to as the positive class, whereas the majority class is considered to be the negative one. There have been several proposals for coping with imbalanced data sets [34] where the main goal is to correctly classify minority examples. In this study, we utilize the queue-based DynaQ framework, as introduced in Chapter 3. Recall that in DynaQ, we combine batch-based and instance-based learning, as will be discussed in the next section.

4.3 MQ-OFL Framework

Our MQ-OFL approach, as shown in Figure 4.2, consists of three stages: customized queue construction, prediction and fair online learning. Our method includes a class imbalance monitoring and balancing step, that keeps track of the class ratios over the stream and adjusts the proportion of classes by assigning them to related class label queues. In addition, the queue-based system is provided to train customized classifiers based on each sensitive attribute and the subgroups. Each arriving instance is evaluated both based on the sensitive attributes and the class label while testing for concept drift. In the case where we have concept drift, we employ the previously introduced HDDM algorithm. By employing a fairness-aware post-processing method, the decision boundary is adjusted to ensure that the classifier does not incur discrimination.

4.3.1 Balanced and Fairness-Aware Pre-processing

In our work, we model each individual (person) as being described by a tuple $((x, \bar{x}), y)$, where $x \in X$ denotes a vector of protected attributes, $\bar{x} \in \bar{X}$ denotes a vector of unprotected attributes, and $y \in 0, 1$ denotes a label. We assume that points (X, y) are drawn

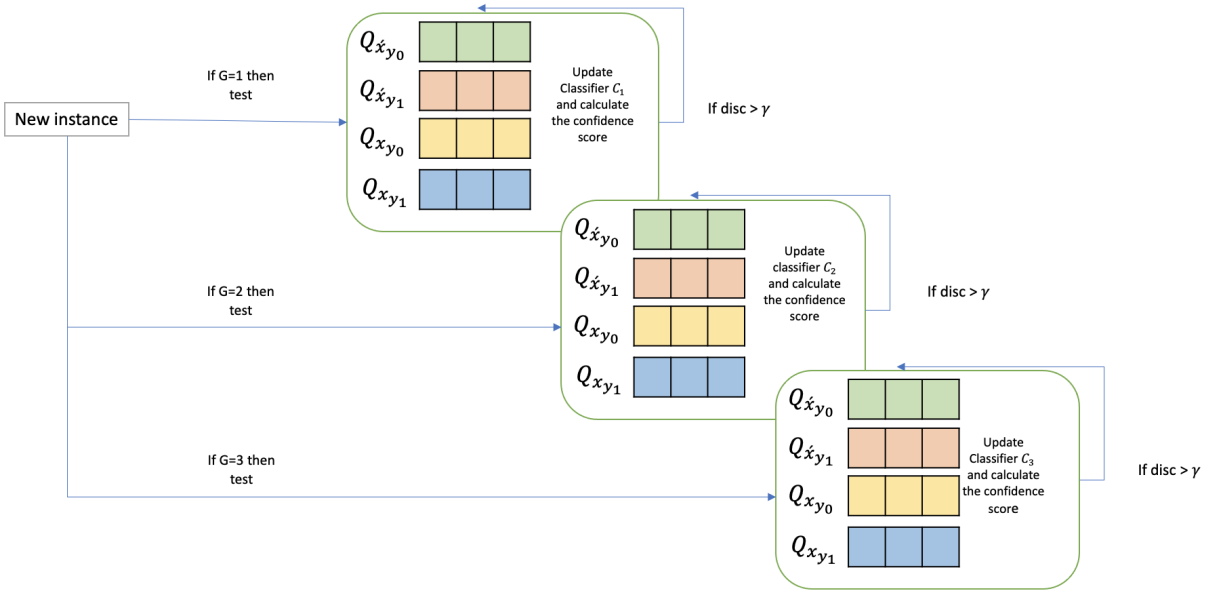


Figure 4.3: Example of updating sensitive queues and forming batches

i.i.d. from an unknown distribution P. Let D be a binary classifier where $D(X) \in 0, 1$ denote the (possibly randomized) decision induced by D on individual (X, y).

Figure 4.3 illustrates how $Queue_Fair$ works when we have two sensitive attributes. Each arriving instance (x_t, y_t) is evaluated based on belonging to a group $G = \{g_1, g_2, g_3\}$ and added to the related queue of equal length $q^t_{C_k} = L$, where c_k is the class label. Together, the queues of the same group form a batch. Each of the training batches will customize a separate classifier.

4.3.2 Classifier Pool

As shown in Figure 4.3, each protected group has its own classifier trained based on the importance of that sensitive attribute. First, a candidate classifier pool is established by training a number of base classifiers with related batches to achieve better accuracy and fairness. Generally, base classifiers can be generated on the sample-level, feature-level, or algorithm-level, in which classification models are produced from various sample subsets, feature subsets, or by learning algorithms, respectively. We used this idea to combine sample- and feature-level techniques to select samples based on features from the original

dataset as a training subset. Second, the classifier with the highest confidence score for arriving instance will choose to predict the label. A confidence score is calculated as an evaluation standard; it shows the probability of the instance being detected correctly by the classifier and it is given as a percentage. The scores are taken on the prediction precision of each classifier for each arriving instance.

4.3.3 Decision Boundary Adjustment

Adjustment of the decision boundary for discrimination elimination has been investigated in the literature [114]. The authors in [62], proposed an approach to achieving fairness by shifting the decision boundary (SDB) for the protected group in the static datasets. They illustrate that SDB may be combined with a member of the family of learning algorithms that produce a measure of confidence in its prediction. It follows that, since we are dealing with streaming, we do not have access to all predictions through the stream to adjust the boundary accurately. Thus, following [82], we estimate the number of instances n_t which are needed in order to mitigate discrimination at time point t by:

$$n_t = \lfloor \sum_{i=1}^{t} 1 \cdot \mathbb{I}[x_i \in z] \cdot \frac{\sum_{i=1}^{t} 1 \cdot \mathbb{I}[f_i(x_i) = y^+ | x_i \in \bar{z}, y^+]}{\sum_{i=1}^{t} 1 \cdot \mathbb{I}[x_i \in \bar{z}, y_i^+]} - \sum_{i=1}^{t} 1 \cdot \mathbb{I}[f_i(x_i) = y^+ \mid x_i \in z, y^+] \rfloor \tag{4.2}$$

We consider a window size of M to keep misclassified instances together with their confidence scores in descending order. In case we have discrimination, the top n_t number in the window will be adapted at the boundary.

Algorithm 4.1 MQ-OFL Methodology

Input: A Discriminated Data Stream D,

while stream.has_more_instances() at each time step t **do**
  x_i^t, y_i^t = get.next_instance()
  y_i^t_predict = Classifier_Highest_Confidence_Score.Predict(x_i^t)
  for $i \in G$ **do**
    if $x_i^t \in g_i$ **then**
      $IncrementCounter_{g_i}$;
      $Q_{g_i}^t = {}_{g_i}^{(t-1)}\ .append(x_i^t)$;
    else
      $IncrementCounter_{C_{other}}$;
      $Q_{C_{other}}^t = {}_{C_{other}}^{(t-1)}\ .append(i)$;
    end if
    if $Q_{g_i} == L$ and $Q_{C_{other}} == L$ **then**
      $Training\_set_{classifier_i} = Q_{g_i} + Q_{c_i}$
      $Classifier_i.Incremental.Update(Training\_set)$
    end if
  end for
  Update classifier pool and confidence scores
  if HDDM detects drift **then**
    Fill the queues with new data
  end if
  if Discrimination level $\geq \alpha$ **then**
    Adjust the boundary
  end ifreturn G_mean, F_Measure,Statistical_Parity, Model
end while

Algorithm 4.1 summarizes our methodology. When online learning starts, at each time step, each instance (x_t, y_t) arrives at time t and then receives the predicted label from the classifier with the highest confidence score from the classifier pool with label set $Y = \{0, 1\}$. When training starts, the queue sizes for all current protected and non-protected groups are assessed; if there are full queues, the classifier is able to update the learning model; i.e., the training process utilizes a balanced set of G and y consisting of the most recent data related to them. It follows that both the batch size p and the sizes of the individual queues are highly domain-dependent; these values are set by inspection. Next, the confidence scores and the Classifier pool are updated, and the HDDM drift detection algorithm is initiated. Finally, we evaluate the discrimination level; if it is higher than a pre-defined α, we employ

boundary adjustment. To this end, we use a sliding window model of a pre-defined size M. In particular, we maintain a sliding window of size M for each sensitive attribute to allow for boundary adjustment for different classifiers based on each discriminated group. Finally, the evaluation metrics update and the learning process continues.

4.4 Experimental Evaluation

In this section, we conduct experiments to evaluate the accuracy and fairness of the MQ-OFL framework. To this end, we first investigate the enhanced discrimination reduction capability of the proposed fair reprocessing. We also show a comprehensive quantitative evaluation to verify the ability of handling class imbalance and adapting to concept drift. All experiments were conducted on a MacBook Pro with a Dual-Core Intel Core i5 processor, CPU @ 3.1 GHz processor, 8.0 GB RAM on the Mac Catalina Operating System (OS), and the *Name Withheld* Cloud with 10 Core CPUs. Our code was implemented using the Scikit-Learn [125] and Scikit-Multiflow [117] packages in Python version 3.8.2. The framework's implementation and all the code for the experiments will be made available on GitHub upon publication.

The Hoeffding Tree (HT) [52] and Hoeffding adaptive tree (HAT) [23] classifiers were used as our base classifier in the model. Recall that HTs are incremental decision trees for data stream classification that use Hoeffding's bound to commence online learning. HAT is an extension of HT that adaptively learns from data streams that change over time without needing a fixed-size sliding window. We evaluate MQ-OFL against two recent state-of-the-art fairness-aware stream classifiers FAHT [166] and FEAT [165]. Recall that the FAHT method solves the discrimination problem by introducing a new splitting criterion, called fair information gain (FIG), that jointly considers the fairness gain and information gain of the introduction of an attribute split. On the other hand, FEAT embedded the fair-enhancing splitting criterion and includes change detection and concept forgetting to handle discriminated and non-stationary data streams.

4.4.1 Datasets

Our experimental study is based on the datasets used in the recent works in this research direction [82, 114, 165]. The following datasets are shown in Table 4.1: *Adult* [11], the $COMPAS$ dataset [98] of criminal recidivism, the $Default$ dataset [162] and the *Bank* dataset [118]. There are $48,843$ instances in the *Adult* dataset and each instance is described by 14 employment and demographic attributes. Following the state-of-the-art, we

conduct experiments on these datasets by setting “gender” and "ethnicity" as the sensitive attributes with female and black being the sensitive value and an annual income of more than 50K as the target class, i.e., the positive classification. The *Bank* dataset comprises $41,188$ samples with 20 features and a binary label, indicating whether clients have subscribed to a term deposit. For this dataset, ages less than 25 and more than 60 years and marital status being single/married are considered sensitive. The $CreditCardDefault$ dataset considers age (same as the *Bank* dataset) and gender as sensitive attributes. We select a subset of the $COMPAS$ dataset previously used for fairness experiments, which comprises $5,320$ samples with five features (age category, gender, ethnicity, priors count, and charge degree) and a binary label indicating whether the defendant re-offended within two years.

The reader should notice that all datasets include more than one sensitive attribute (indicated in Table 4.1) which makes them useful for evaluating our MQ-OFL method. It must be noted that the selected features aim to facilitate fairness experiments comparable to previous approaches, rather than only focusing on high predictive accuracy.

Table 4.1: Characteristics of data streams used in experiments.

Dataset	#Instances	#Attr.	Sen. Attr.	Imb. Ratio	Class Label
Adult Cen	48843	14	Gender/Ethnicity	1 : 3	$\leq 50K$ or $\geq 50K$
Bank	41188	16	Marital Status/Age	1 : 7.5	Subscription (yes/no)
Default	30000	24	Gender/Age	1 : 3.5	Default Payment (yes/no)
COMPAS	13610	4	Gender/Ethnicity	1 : 1.1	Re-offended (yes/no)

4.4.2 Evaluation Metrics

We evaluate whether a classifier D is satisfying statistical fairness constraint based on the previously introduced statistical parity (S.P.). This fairness metric is defined with respect to a set of protected groups G if we have more than one sensitive attribute. Each $g : X \rightarrow \{0, 1\} \in G$ has the semantics that $g(x) = 1$ indicates that an individual with protected features x is in the group g. Definition 4.3 (S.P. Subgroup Fairness) [89]; refers to classifier D, distribution P, collection of group indicators G, and parameter $\gamma \in [0, 1]$. For each $g \in G$, define:

$$\alpha_{SP}(g, P) = Pr_p[g(x) = 1], \beta_{SP}(g, D, P) = \|SP(D) - SP(D, g)\| \tag{4.3}$$

Where $SP(D) = Pr_{P,D}[D(X) = 1]$ and $SP(D, g) = Pr_{P,D}[D(X) = 1 \| g(x) = 1]$ denote

the overall acceptance rate of D and the acceptance rate of D on group g respectively. We conclude that D satisfies γ-statistical parity (SP) Fairness with respect to P and G if for every $g \in G$:

$$\alpha_{SP}(g, P).\beta_{SP}(g, D, P) \leq \gamma. \tag{4.4}$$

We refer to SP(D) as the S.P. base rate. For S.P. fairness, if the algorithm D fails to satisfy the γ-fairness condition, then we conclude that D is γ- unfair with respect to P and G. We call any subgroup g which witnesses this unfairness an γ-unfair certificate for (D, P).

Our learning procedure is supervised and is known as first-test-then-train or prequential evaluation [71]. The performance measures we used are the F-measure, and geometric mean (G-mean). The F-measure [34] refers to the harmonic mean of two metrics, recall and precision. We used a balanced value, which implies that precision and recall are assumed to carry equal weights in the metric. The F-measure is macro-averaged over the sum of F1-scores over all classes, which assigns equal weights to the existing classes.

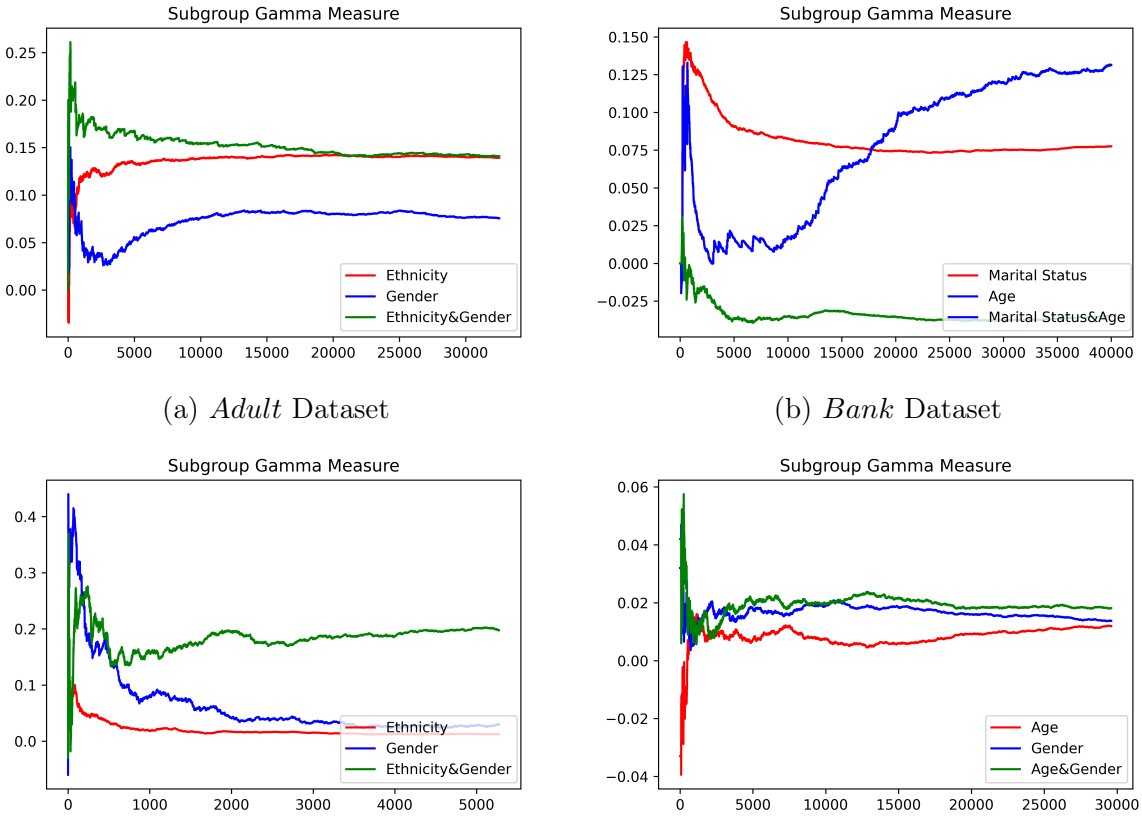
Additionally, we employed the G-mean [71] value which is the geometric mean of the recall rates of majority and minority classes in the imbalanced data set. The G-mean value is higher only when the classification accuracies of the majority sample and the minority sample are high; therefore, the G-mean value can accurately the classification effect of unbalanced data sets.

4.4.3 Experimental Results

First, we investigated the value of S.P. as the indicator of fairness violation and vary the γ measure over time to show the maximum unfairness for each subgroup with gerrymandering. Unfortunately, inter-sectional fairness is not statistically estimable in most cases as most intersections are empty. As a remedy, [88] propose max-violation fairness constraints over $G_{gerrymandering}$, where each group is weighed by group size, defined by:

$$max_{g \in G_{gerrymandering}} \frac{\|g\|}{n} \tag{4.5}$$

Subsequently, following [88], the empty groups are removed, and small groups have relatively low influence unless there is a very large fairness violation. It is also of interest to compare the subgroup fairness achieved by the subgroup customized classifier. We depict the change of γ over the time in Figure 4.4. By referring to Figure 4.4, we notice that the combination of Ethnicity and Gender has the highest value when we consider the Adult and Compass datasets. That is, Ethnicity and Gender is the subgroup with the higher



(a) *Adult* Dataset

(b) *Bank* Dataset

(c) *Compass* Dataset

(d) *Default* Dataset

Figure 4.4: Maximum $\gamma_{gerrymandering}$

fairness violation. On the other hand, the combination of Marital-Status and Age in the Bank dataset exhibits the lowest level of unfairness. Further, in the Default stream, the combination of Marital-Status and Gender in the Default dataset has the highest fairness violation among all subgroups.

Next, we also present the maximum gerrymandering value for all datasets used in the chapter. As shown in Table 4.2, the value of γ_{Bank} is between -0.02 and 0.13 indicating the highest variation among subgroups. The *Default* dataset reached the lowest value of γ equal to 0.011. We find that Adult is $\gamma_{0.05}$-fair, Bank is $\gamma_{0.03}$-fair, COMPAS is $\gamma_{0.013}$-fair and Default is $\gamma_{0.011}$-fair. The results depict that our method is empirically necessary to avoid fairness gerrymandering.

Next, we turn our attention to the second set of experiments, where we investigate

Table 4.2: Gamma measure in each subgroup G

Data	γ_{g_1}	γ_{g_2}	γ_{g_3}
Adult	0.05	0.14	0.16
Bank	0.07	0.131	-0.029
COMPAS	0.013	0.04	0.19
Default	0.011	0.014	0.019

the accuracy-driven and fairness-oriented capabilities of MQ-OFL. Kearns et al. [88], had shown that varying the input α provides an appealing trade-off between accuracy and fairness. We begin by examining the evolution of the accuracy and discrimination of the model.

Based on experimental results we determine the best α and accuracy trade-off. For instance, we show the values of α ranging from 5.0 to 22.5 for *Adult* and between 0 and 2.5 in the *Bank* data set. This implies that the fair customized classifiers aid to control the discrimination propagation and manage to push the discrimination to a low level while maintaining a high prediction capability. Note that the trade-off between accuracy and discrimination can be achieved, by inspection, by adjusting α.

Finally, we present our results when contrasting different fairness approaches. The HAT and HT classifiers were used as our baseline without considering class imbalance and discrimination. Specifically, Table 4.3 presents the results of our comparative study contrasting the MQ-OFL, FAHT, and FEAT algorithms. Table 4.3 presents the results of our comparative study when contrasting these three algorithms. In these results, it is clear that our model is capable of diminishing the discrimination to a lower level while maintaining comparable accuracies across all data sets. In addition, the MQ-OFL algorithm produced the highest values in terms of G-mean and the lowest value for discrimination for data streams. The same observation holds for the F-measure, where again, MQ-OFL produced higher results, especially handling class imbalance helps to improve this measure. When assessing the overall accuracy and discrimination level, the reader will notice that MQ-OFL is able to mitigate unfair outcomes and maintains the highest performance in terms of G-mean and F-measure for all datasets, followed by HAT. In terms of discrimination, MQ-OFL appears fairest when considering two sensitive attributes, across all datasets, with keeping the least discrimination score. This is followed by FEAT, which was designed for enhanced fairness-aware learning with an add-on concept drift adaptation ability to handle non-stationary discriminated data streams. In particular, MQ-OFL achieves a discrimination score of 15.12% and 1.34% at the cost of a slight 1.02% and 2.10% accuracy reduction on the *Adult* and *Bank* dataset, respectively. The results suggest that our

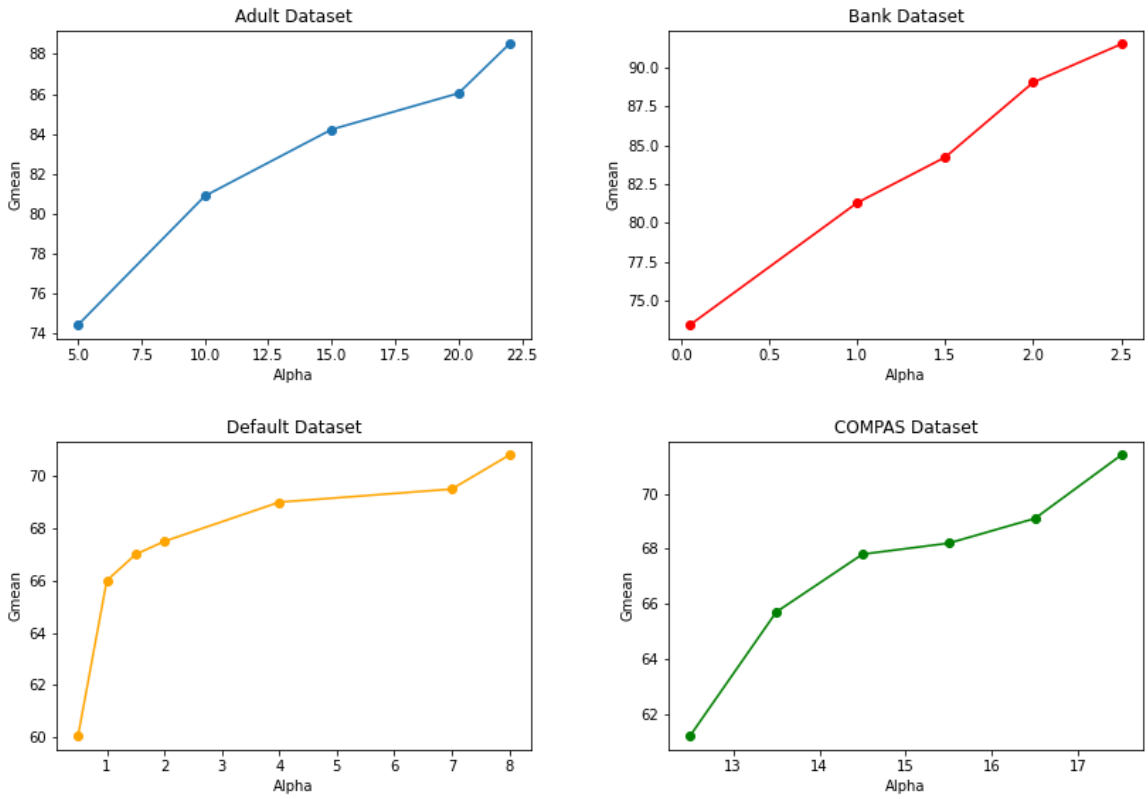


Figure 4.5: Accuracy and discrimination trade-off

MQ-OFL method which seamlessly integrates the fair data merit into classifiers, results into a model that is both accuracy-driven and fairness-driven. MQ-OFL achieves the best discrimination reduction while HT and HAT give the worst fairness results. This is not surprising; due to the exclusively accuracy-oriented tree construction and the intrinsic discrimination bias of the historic data, a lack of fairness trees can be induced during the construction of the HT and HAT. Therefore, although HAT provides a better prediction performance, it may lead to an unfair model.

4.5 Summary

In this chapter, we introduced an approach for fairness-aware stream classification, while addressing class imbalances and concept drifts. Our MQ-OFL algorithm maintained pre-

Table 4.3: Accuracy-vs-discrimination of learning methods.

	Adult			Bank		
Method	Discrimination	G-mean	F-measure	Discrimination	G-mean	F-measure
MQ-OFL	**15.12**	83.05	**80.69**	**1.34**	83.16	**85.11**
FEAT	16.23	81.83	75.69	1.70	79.50	76.95
FAHT	17.40	79.07	71.66	2.65	77.07	73.48
HT	22.60	83.91	72.90	8.10	80.45	71.65
HAT	22.30	**84.07**	76.86	7.80	**85.06**	72.92
	COMPAS			Default		
Method	Discrimination	G-mean	F-measure	Discrimination	G-mean	F-measure
MQ-OFL	**13.50**	65.74	**63.93**	**1.03**	**66.10**	**64.17**
FEAT	15.80	61.62	59.92	1.83	62.58	44.90
FAHT	16.31	60.05	59.84	2.04	53.30	40.81
HT	21.30	65.38	62.18	8.34	65.10	58.23
HAT	19.70	**66.44**	63.03	8.91	66.07	59.08

dictive performance while minimizing discrimination scores in evolving streams. Moreover, our MQ-OFL method facilitated two or more sensitive attributes by customizing the classifier for each protected group. The experimental evaluation showed that our approach outperforms other methods in terms of both predictive performance and fairness preservation across various datasets.

Next, in Chapter 5, we present our work regarding explainability and interpretability in online machine learning.

Chapter 5

An Explorative Study of Explainability and Interpretability in Fairness-aware Online Learning

The work presented in this chapter is based on our paper: *Farnaz Sadeghi and Herna Viktor, "An Explorative Study of Explainability and Interpretability in Fairness-aware Online Learning", 39 pp. (2023)*, that is submitted for publication.

In this chapter, we investigate explainability and interpretability techniques [72], building upon the MQ-OFL algorithm discussed in the previous chapter. We illustrate how these approaches may be used to assess fairness, transparency, and trustworthiness in decision-making contexts that affect both individuals and groups [124].

This chapter is organized as follows. Section 5.1 provides a brief introduction, while Section 5.2 discusses related work. In Section 5.3, we detail the explainability and interpretability methods employed in this work. Section 5.4 presents our experimental evaluation, while Section 5.5 provides a summary.

5.1 Introduction

Decisions made by machine learning systems in areas such as job applications, credit approvals, and autonomous driving are of critical importance for many people [130]. However,

these automated, data-driven systems may lead to unconscious discrimination against particular groups of people who share one or more sensitive attributes. To prevent this, discrimination-aware systems must be trained to prevent discrimination against sensitive features known as "sensitive attributes" or "protected attributes" i.e., attributes that are disproportionately associated with positive or negative outcomes [114]. In this paper, we use these two terms interchangeably. Each application has its own set of protected attributes, such as age, biological sex, ethnicity, and other similar features [114]. To build trust in machine learning solutions and prevent the emergence of an algorithm-based authoritarian society, it is important that automated decisions can be explained and interpreted transparently [79]. Recall that online learning algorithms from data streams process each incoming example "on arrival" without needing persistent storage and multiple scans while maintaining a model that reflects the current data [137]. In the online machine learning setting, it is challenging to develop algorithms that are fair and unbiased toward all groups and that still operate in real-time.

Fairness-aware algorithms for streaming data are limited in the existing literature, and most research is focused on scenarios with only one sensitive attribute [114]. To address the challenge of imbalanced data distributions with multiple sensitive attributes, we have developed the MQ-OFL method in Chapter 4. Recall that MQ-OFL manages imbalanced data using a discrimination-aware pre-processing method. The pre-processing method maintains a batch of recent instances organized into multiple queues based on similar combinations of sensitive attributes and output labels. The MQ-OFL algorithm uses a fairness metric to identify situations where certain sensitive attribute combinations are disproportionately associated with positive or negative outcomes. It employs a decision boundary adjustment approach [62] to improve fairness while maintaining classification accuracy and updating the classifier in real-time.

A drawback of the MQ-OFL algorithm is that it produces a so-called black-box model. Such black-box models are common in machine learning, and are non-transparent models created by machine learning algorithms, whose internals are either unknown to the observer or uninterpretable by humans [72]. Specifically, the MQ-OFL algorithm results in a complex black-box model that uses multiple decision trees trained on different data subsets to make predictions and employs a fairness-based post-hoc method to correct the decision boundary. It therefore becomes a challenge to understand the relationship between features and labels as well as to explain the factors that influence predictions, as online models are updated incrementally over time. In this setting, explainability and interpretability techniques [72] may be used to assess fairness, transparency, and trustworthiness in decision-making contexts that impact individuals and groups [124]. To achieve this, one approach is to build a surrogate model (local/global) [29], which is a simplified and understandable model that mimics the behaviour of a more-complex black-box

model. Global surrogate models provide an overall explanation of the black-box model's behaviour. In contrast, local surrogate models approximate the behaviour of the black-box model within a specific region of the input space. This involves using simpler and more intuitive decision algorithms to build a surrogate model of the black-box model used for decision-making [151].

This chapter addresses the need to interpret the results of the fairness-aware MQ-OFL algorithm through the construction of surrogate models. Specifically, we address two main research questions:

- How does the MQ-OFL algorithm address the challenges posed by learning from imbalanced datasets containing multiple sensitive attributes?
- How can global and local surrogate models aid in understanding the decision-making process of a fairness-aware online learning algorithm and make it more interpretable?

5.2 Explainable Fairness-aware Models

In this section, we provide definition of fairness-aware models and explore related approaches, while also connecting them to the corresponding aspect of interpretability.

In addition to accuracy, interpretability, and fairness are crucial considerations for businesses and researchers when developing machine learning models. The purpose is to explore not only whether decisions made by these systems are unbiased and non-discriminatory but also whether the reasons for those decisions can be easily understood by and communicated to users. The techniques and approaches used to achieve explainable fairness in machine learning include transparency and interpretability methods, fairness metrics, and fairness-aware approaches [88]. Fairness metrics have been defined in the literature [77] to assess discrimination in machine learning models. Fairness can be interpreted differently based on the context or application. Individual fairness, also known as "treatment fairness," requires similar individuals to be treated similarly. In other words, if two individuals have similar characteristics, they should receive similar outcomes or decisions from the machine learning model. This notion of fairness focuses on the treatment of individual instances and does not consider group membership [114]. Group fairness requires that the machine learning model treats different groups of individuals similarly. This notion of fairness focuses on the overall distribution of outcomes or decisions across groups, such as different ethnic or biological sex groups. These measures aim to ensure that different groups have equal representation in the outcomes produced by the machine learning model [89]. In

addition, recent studies have suggested strategies to handle bias and discrimination in machine learning systems. Three main categories are pre-processing approaches, in-processing approaches, and post-processing approaches [114].

Pre-processing approaches aim to reduce bias in the data by modifying the training data to eliminate or minimize discriminatory patterns before feeding it into the learning algorithm. In-processing approaches modify the learning algorithms themselves to reduce discrimination during the model induction process. Post-processing approaches aim to alleviate bias at the output level by modifying the model's predictions after it has been trained.

Interpretability methods, as introduced in Chapter 2, may be applied to trained models allowing the development of fairness-aware machine learning systems. These methods play a crucial role in detecting and mitigating bias during data collection or labeling processes. For example, feature importance can be used to identify fairness failures, such as when a feature has a larger effect than it should, indicating that the algorithm is overly dependent on that feature. A study conducted by Adebayo et al. [2] on a bank's credit limit model found that biological sex is not a highly important demographic feature, which suggests that the model does not rely heavily on biological sex to make credit limit determinations. In this case, biological sex is a sensitive attribute. However, it does not have an effect on the fair outcome of the dataset. In [124], the authors explored the relationship between interpretability and fairness. The study found a direct link between the difference in feature importance values and equality of opportunity after bias was removed from the dataset. That is, interpretability and fairness should be considered together to clarify the hidden correlations between datasets and decisions.

Most current fairness-aware learning approaches concentrate on static datasets, whereas MQ-OFL focuses on an online environment where data is received sequentially and the data distribution can evolve over time. Recall from Section 2.4.4 that FAHT method [166] addresses discrimination by integrating discrimination-awareness into the model induction process. This is achieved by introducing a fair information gain splitting criterion, which maintains moderate predictive performance while minimizing discrimination scores over data streams. Another approach [165] introduces the Fairness-Enhancing and concept-Adapting Tree (FEAT) which embeds a fair-enhancing splitting criterion. This method incorporates change detection and concept forgetting to handle discriminated and non-stationary data streams. However, a drawback of these two approaches is that they do not account for imbalanced data. Furthermore, none of the current online fairness-aware techniques can simultaneously handle more than one sensitive attribute. Our work addresses this inadequately explored research direction by incorporating multi-sensitive fairness, where different sensitive attributes can be taken into account simultaneously.

It follows that the MQ-OFL algorithm exhibits a high level of complexity, i.e., it is a

black box model that requires interpretability and explainability. The online training process involves training different classifiers with different batches at each time step, further decreasing the interpretability. Furthermore, our approach incorporates a fairness measure to adjust the decision boundary at each time step. As a result, multiple decision trees trained on different subsets of data to make predictions need to be interpreted. Additionally, a fairness-based post-hoc method is employed to refine the decision boundary. This complexity presents challenges in understanding the relationship between features and labels, as well as explaining the factors that influence predictions. Due to the incremental updating of online models over time, it becomes difficult to comprehend the inner workings of the model and provide clear explanations for the predictions generated, notably to assess the fairness and unbiased nature of the final predictive model. The next section provides more details on surrogate models and the concepts it employs.

5.3 Surrogate Models

As described in Section 2.4, the interpretable model category utilizes surrogate models as a solution. To effectively employ this category, it is necessary to establish a comprehensive framework that includes both interpretable and black-box models. This is achieved by constructing a surrogate model using more-expressive machine learning algorithms, such as decision rules or decision trees [29]. By using a surrogate model, we can efficiently examine the behaviour of complex models. Surrogate models are model-agnostic and do not require knowledge of the black-box model's operation [29].

5.3.1 Global Surrogate Model

Recall that this approach is an inherently interpretable model that was developed to approximate black-box model predictions. To explain the global output of a model, we need a trained model, knowledge of the algorithm, and data. Figure 5.1 explains the visual process for creating a global surrogate model. Initially, data is fed into the black-box model to generate a prediction. The surrogate model is then trained using input data and predictions from the black-box algorithm's model. Global interpretability has been widely used to extract knowledge from models such as decision trees, where interpretability difficulty depends on the depth and number of terminal nodes, and linear and logistic regression models, where prediction is a weighted sum or a weighted sum transformed by the logistic function [1]. Following Algorithm 5.1, to construct a surrogate model, we first choose a dataset X, which can be either the same one used to train the black-box model (the model

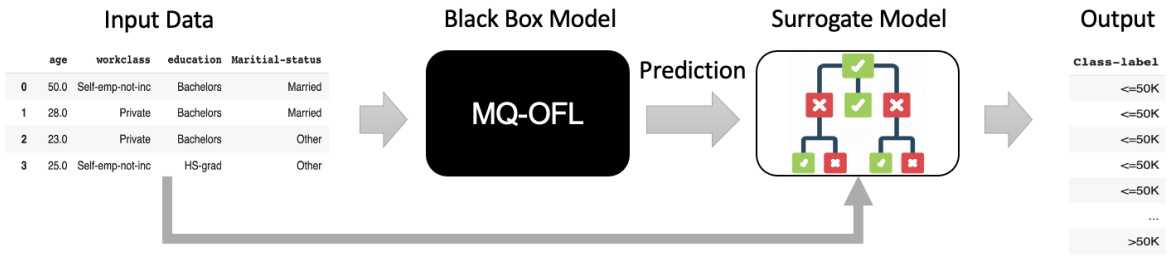


Figure 5.1: Framework of fairness explainability and interpretability through surrogate models.

Algorithm 5.1 Global Surrogate Model Construction

Input: A Discriminated Data Stream D,

1: **while** stream.has_more_instances() at each time step t **do**
2: x_i^t, y_i^t = get.next_instance()
3: y_i^t_predict = MQ-OFL_Predict(x_i^t)
4: Train(MQ-OFL,$\{x_i^t, y_i^t\}$)
5: **end while**
6: Surrogate Model Construction
7: Train_Interpretable_Model(x, y_predict)
8: Draw Interpretable_Model
9: **return** Calculated prediction on each node

trained by MQ-OFL) or a new dataset from the same distribution. Next, we obtain the predictions made by the black-box model for the selected dataset X. Subsequently, we choose an inherently interpretable algorithm, such as a linear model or a decision tree, to use as the global surrogate model to train the interpretable model on dataset X and its predictions. It is crucial to assess how well the trained surrogate model approximates the predictions of the black-box model, as this determines the effectiveness of the interpretation. If the surrogate model cannot replicate the behaviour of the original black-box model, then any interpretation of the surrogate model would not explain the original model. The final step is to interpret the surrogate model.

Global surrogate models are often used to gain an overall understanding of the black-box model's behaviour. In contrast, local surrogate models are used to identify and understand specific aspects of the black-box model's behaviour in a particular region of the input space, as we discuss next.

5.3.2 Local Surrogate Model

Recall from Section 2.4 that local surrogate models can be used to explain the reasoning behind an individual prediction made by a model created by a black-box algorithm. These models are used to approximate the local decision boundary of the black-box algorithm. The advantage is that they can offer a more accurate estimation of the behaviour of the model created by the black-box algorithm within the region of interest [148]. The interpretation of local models can be achieved by designing model architectures that justify why a particular decision was made – for instance, by highlighting the specific characteristics of an individual in comparison to those of a smaller group of individuals (female) but different from other individuals (male).

One such technique, which is used in this research, is Local Interpretable Model-Agnostic Explanations (LIME) [134]. Instead of creating a global surrogate model, LIME trains local surrogate models to explain individual predictions. The objective is to understand why the machine learning model produced a specific prediction. LIME examines the impact on the predictions when variations of the data are introduced into the machine learning model. LIME then generates a new dataset that consists of perturbed instances and the corresponding predictions of the black-box model. LIME trains an interpretable model on this new dataset, weighted by the proximity of the sampled instances to the instance of interest. The learned model should be a good approximation of the black-box's predictions locally, but it does not need to be a good global approximation. This level of accuracy is also referred to as local fidelity. The steps for training this local surrogate model are as follows [115]:

- Select the instance of interest for which an explanation of the black-box prediction is desired.
- Perturb the dataset and obtain the black-box predictions for these new points.
- Weigh the new examples based on their proximity to the instance of interest.
- Train a weighted, interpretable model on the dataset with the variations.
- Explain the prediction by interpreting the local model.

In addition to LIME, we utilized a second, local interpretable technique in this study, which involved training an inherently interpretable algorithm with a subset of the original data. Instead of perturbing new examples that are similar to the instance of interest, we utilize examples from the dataset with features similar to the test examples to train the

interpretable model and predict the class of test instances. This technique results in a local model that makes decisions that are both interpretable and traceable.

In the next section, we provide the details of our experimental evaluation using surrogate models to interpret the models constructed by our MQ-OFL algorithm.

5.4 Experimental Evaluation

In this section, we present a series of experiments to explain and interpret the behaviour of the model created by the MQ-OFL approach against the Adult data stream, which is used to predict a person's income. We demonstrate a comprehensive interpretability process in two distinct sections, focusing on pre-modelling explainability and interpretation through surrogate models. We introduce the dataset we used and examine its properties, such as imbalance and discrimination, to gain insight into the input data. This allows us to observe patterns in the data and apply visualizations to provide prescriptive insights for end users. Next, we utilize surrogate models to approximate the MQ-OFL algorithm-generated model's behaviour. This approach allows us to interpret the model's decision-making process across the entire input dataset and identify the important features that lead us to the decisions and the set of rules the model followed to make fair decisions. Moreover, we provide a detailed breakdown of the model's insights on individual predictions, particularly in cases where the model uses online training. This provides users with an understanding of why the model makes certain decisions and how it arrives at each one. In addition, we present the experimental results when considering both fairness and performance aspects. All experiments were conducted on a MacBook Pro with a Dual-Core Intel Core i5 processor, CPU @ 3.1 GHz, 8.0 GB RAM, running the Mac Ventura Operating System (OS), and the Google Colab Notebook [27]. Our code was implemented using the Scikit-Learn [125], Scikit-Multiflow [117], River [116], LIME [134], and dtreeviz [54] packages in Python version 3.8.2. The framework's implementation and all the code for the experiments will be available on GitHub upon publication.

We conduct three sets of experiments. In the first set of experiments, we construct a global surrogate model to assess the algorithm. Secondly, we use the LIME framework to obtain insights into the local decision flows. Thirdly, we evaluate the impact of the fairness-aware MQ-OFL learning process using a local incremental surrogate model. We utilized two performance measures, namely the F-score and G-mean, to assess our results. The F-score [34] is calculated as the harmonic mean of recall and precision, where equal weight is given to both metrics. This balanced approach ensures that recall and precision contribute equally to the overall metric. The G-mean [71] value is particularly informative

when dealing with unbalanced data, as it increases only when the classification accuracies of both the majority and minority samples are high. Thus, the G-mean accurately reflects the classification performance in unbalanced datasets. In our MQ-OFL model we employed Hoeffding Tree (HT) Classifier [52] as our base classifiers initially trained with the previously introduced. Recall that, HT is an incremental decision tree [80] specifically designed for data stream classification that uses Hoeffding's bound to enable online learning. As previously mentioned, decision trees are a commonly used surrogate model due to their inherent transparency. In our study, we utilized a Hoeffding Tree (HT) classifier as our global surrogate model and local incremental surrogate model. HT is an incremental decision tree [80] specifically designed for data stream classification that uses Hoeffding's bound to enable online learning. The HT algorithm recursively splits the data based on the most-significant attribute that minimizes impurity in the resulting subsets and builds a decision tree that can be easily interpreted. The decision process can be traced back to the root node and provide transparency in the model's decision-making process. The HT work is based on the impurity measures, for which we employed Gini impurity [66] as a criterion for selecting the best split at each node. In other words, the higher the Gini value, the more erratic the tree, and as a result, the less information is conveyed (=impure). Mathematically, the Gini impurity for a set of instances X is calculated as follows:

$$Gini(X) = 1 - \sum_{i=1}^{c} p_i^2 \tag{5.1}$$

where c is the number of classes in the set, and p_i is the proportion of examples in X that belongs to class i. The Gini impurity is minimized when all examples in X belong to the same class, and it is maximized when the examples are evenly distributed across all classes. The result of Equation 5.1 is used to determining the best split at each node by calculating the weighted sum of the impurities of the resulting subsets after the split. The split with the lowest weighted impurity is selected as the best split. We performed hyperparameter tuning on the HT algorithm and selected the best parameters. Specifically, we set the maximum tree depth to 3 and the minimum instances per leaf to 1. For the leaf nodes, we used the majority class prediction mechanism.

In addition, we utilized a Gini-based decision tree (DT) algorithm [80] in the LIME settings. In hyperparameter tuning for the DT method, we established the maximum tree depth to 4 and the minimum number of samples per leaf to 1. Additionally, we set the maximum number of features to 20 and the minimum number of samples per split to 2.

5.4.1 Adult Dataset

In this study, we utilized the Adult dataset, which is commonly used in the literature to assess models' fairness based on their ability to predict income without discrimination based on sensitive demographic features [15]. This section describes the characteristics of the data stream, followed by a report on the distributions of target classes and sensitive variables. We also discuss feature transformation and data optimization.

5.4.1.1 Dataset Description

The Adult dataset [15] consists of 48,843 instances, each instance modelling a person in terms of employment and demographic attributes. This dataset comprises 14 primary features: **Age**: the individual's age; **Sex**: the individual's biological sex; **Education**: the individual's highest level of education, for example, in "Associate-academic" and "Associate-vocational," an associate's degree refers to individuals who have completed a two-year college program leading to an academic degree. This degree combines general university-level courses and foundational work in a specific discipline, such as an Associate of Arts (AA) or an Associate of Science (AS), whereas an associate vocational degree is a two-year college program that provides skills and knowledge related to a specific profession, such as teacher's aide, so the individual is able to work immediately upon graduation. **Occupation**: the general type of the individual's occupation; **Country**: the individual's country of origin; **Ethnicity**; **Marital status**: the individual's marital status. Married-civ-spouse corresponds to a civilian spouse, and Married-AF-spouse is a spouse in the Armed Forces.

Table 5.1: Adult dataset

Features	Amount
Age	Age group below 25 Age group between 25 and 34 Age group between 35 and 44 Age group between 45 and 55 Age group between 55 and 64 Age group equal/over 65
Sex	Female | Male
Education	Preschool| 1st-4th grades | 5th-6th grades 7th-8th grades |9th grade |10th grade |11th grade 12th grade| High School grade |Some-college Associate-academic |Associate-vocational Bachelors | Masters |Doctorate| Prof-school
Occupation	Administrative-Clerks | Armed-Forces | Craft-repair Executive-managerial | Farming-fishing Handlers-cleaners | Machine-operation-inspector Other-service | Private-house-service |Sales Professional-specialty |Protective-service Technical-support |Transport-moving
Country	United-States | Cambodia| England| Puerto-Rico Germany| United States Minor Outlying Islands| India Greece| South Korea| China| Cuba| Iran| Honduras Philippines| Italy| Poland| Jamaica| Vietnam Mexico| Portugal| Ireland| France| Dominican-Republic Laos| Ecuador| Taiwan| Haiti| Columbia | Japan Hungary| Guatemala| Nicaragua| Scotland| Thailand Yugoslavia| El-Salvador| Trinadad-Tobago| Peru Hong Kon| Holland-Netherlands | Canada
Ethnicity	White| Black
Marital Status	Married-civ-spouse| Never-married| Divorced| Separated Widowed| Married-spouse-absent| Married-AF-spouse
Workclass	Federal-government | Local-government Private | Self-employed-income Self-employed-not-income |State-government Without-payment
Relationship	Husband | Not-in-family | Other-relative Own-child | Unmarried | Wife
Capital-gain	Equal/greater than zero ($>= 0$)
Capital-loss	Equal/greater than zero ($>= 0$)
Work-hours-per-week	Equal/greater than zero ($>= 0$)
Income	Equal/less than fifty thousand dollars per year ($=< 50K$) Greater than fifty thousand dollars per year ($> 50K$)

Workclass is a general term representing the individual's employment status; for example, "Self-employed-income" means that the individual owns their business and earns income from it. **Relationship** represents what relation the individual has relative to others; for example, the individual could be a husband. Each entry only has one relationship attribute. "Own-child" means that the individual has a child. **Capital-gain** refers to the amount of profit the individual earned from the sale of an asset, such as stocks, bonds, or real estate, and it is measured in US dollars. This attribute is an integer value greater than or equal to 0 and can provide valuable information in predicting the individual's income level or financial status. **Capital-loss** refers to the loss incurred from the individual's sale of an asset. It is an integer greater than or equal to 0. **Hours-per-week** are the hours the individual has reported working continuously per week and **Income** denotes whether the individual makes more than $50,000$ annually.

In the Adult dataset, the capital gain/loss attribute represents the amount of capital gain or loss reported on the individual's tax return for the given year. It is measured in US dollars. This attribute can be useful in predicting the individual's income level or financial status.

5.4.2 Pre-modelling Explainability

Pre-modelling explainability refers to various data pre-processing and feature exploration methods used to obtain an overview of any dataset and pre-process it before the training process [1]. In data analysis, we investigate data to identify the crucial features required for machine learning models to make decisions. Since the MQ-OFL model works on imbalanced datasets that contain multiple sensitive features, we need to analyze these characteristics in our dataset.

5.4.2.1 Multi-feature Sensitive Imbalanced Data

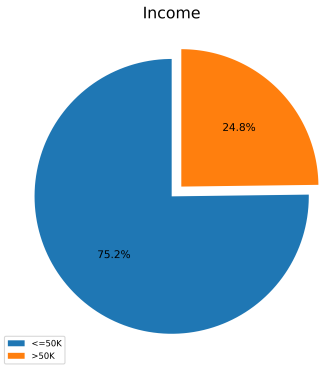
Following the fairness-aware literature, our model takes into account two sensitive features: "Sex," with the females as the disadvantaged group and males as the advantaged group, and "Ethnicity," with Black as the disadvantaged group and White as the advantaged group. The goal of our prediction task is to determine whether a person earns over $50K per year, which is considered a positive classification. In this dataset, the balance ratio of the class labels is 75% with $33,898$ individuals earning less than 50K, against 25% with $11,202$ individuals earning over that amount, which reflects imbalanced class labels.

To investigate the skew of sensitive attributes in our data, we define protected groups by creating binary variables of those sensitive attributes. We assign a value of 1 to the

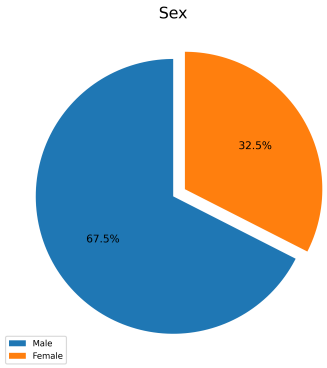
privileged group and 0 to the unprivileged group. Typically, the unprivileged group has been subject to historical injustices within the data and is more likely to face unfair decisions from a biased model. Therefore, we examine whether the dataset is skewed in terms of the frequency of sensitive attributes. Figure 5.2 provides a breakdown of the population by ethnicity and sex. The pie chart Figure 5.2(b) shows that 86% of our population is White, while Figure 5.2(c) indicates that 68% of the population is male. That is, both of the sensitive attributes are skewed.

One issue with an imbalanced dataset is that machine learning model parameters can become skewed toward the majority (negative) class and lead to biased decisions for unprivileged populations. In other words, the sensitive attributes, which already exist in skewed groups, may be even less likely to be predicted as positive classes. Our MQ-OFL model aims to maximize accuracy across the entire population, but doing so may result in decisions that favour the male/White population. Consequently, there may be discrimination against the female/Black population. However, employing a queue-based method by MQ-OFL can overcome this issue by maintaining equal instances from each group. Recall that we dynamically maintain a batch of recent instances and consider the frequency of sensitive attributes and their class labels to manage the imbalanced ratio of sensitive attributes and their class labels.

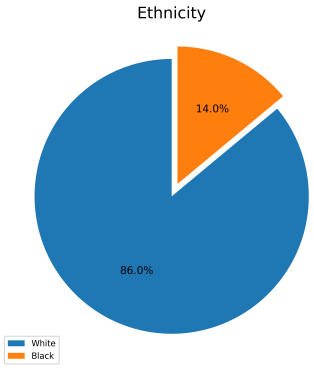
The metric that helps to assess the effects of discriminated data on machine learning model decisions is the prevalence concept. Prevalence refers to the proportion of the positive class to the overall class based on our target class in the dataset. In the Adult dataset, the overall prevalence is 24.8%, which means that roughly $1/4$ of the people in our dataset earn above $50K$. We can also apply this concept to sensitive attributes. The prevalence of defined unprotected (1) and protected (0) groups can be seen in Table 5.2. Notably, the prevalence is much higher for privileged groups. Males are nearly three times more likely to earn above $50K$ than females.



(a) Target class.



(b) Sensitive attribute biological Sex.



(c) Sensitive attribute Ethnicity.

Figure 5.2: Distribution of class label and two sensitive attributes.

Table 5.2: Prevalence of protected features.

Features	1	0
Sex	84.2%	15.8%
Ethnicity	90%	10%

These results worsen when we focus on the intersection of protected groups, such as ethnicity and biological sex. Based on the gerrymandering issue discussed in Section 4.2.3

and despite an algorithm's fairness for one protected attribute in isolation, it may not be fair when considering the intersections of multiple protected groups. The results are shown in Table 5.3. The top-left corner represents the prevalence for individuals who belong to both privileged groups (i.e., male and White), and the bottom-right corner represents the prevalence for individuals who belong to neither privileged group (i.e., female and Black). These results indicate that White males are more than four times as likely to earn over $50K$ compared to non-White females.

Table 5.3: Prevalence of intersection of protected features.

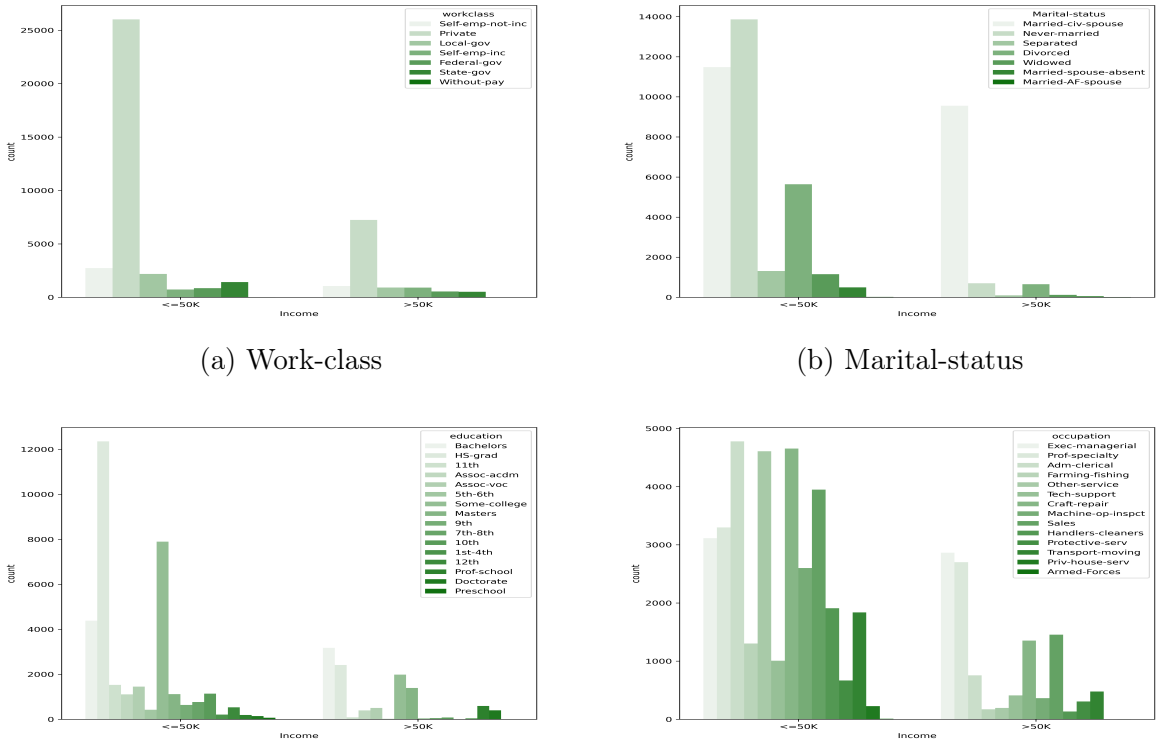
		Ethnicity	
		White	Black
Sex	Male	77%	7.2%
	Female	13%	2.8%

The significance of addressing gerrymandering in data modelling is highlighted in Table 5.3. Recall from Section 4 that the MQ-OFL methodology effectively addresses this issue by utilizing various classifiers to account for intersectional groups. Specifically, a classifier is employed to learn each group separately, and the results are integrated into the final decision.

5.4.2.2 Exploring the Features

This section explores the characteristics of the features and their frequencies based on the class label. The Adult dataset consists of a combination of numeric and categorical attributes. We converted the "Sex" and "Ethnicity" attributes to binary format, because they only have two categories in this specific data stream. Figure 5.3 shows the categorical features that have been collected. Furthermore, the categorical data in this dataset do not have any meaningful order, and we employed one-hot encoding in our data stream.

Due to the high number of countries in the "Country" feature, it was not possible to display it in the image. Therefore, we combined most individuals into two categories: "United States" and "Other." Additionally, we removed the "Relationship" attribute as it was redundant with the "Marital-status" attribute. In the "Marital-status" attribute, we merged "Separated," "Widowed," "Married-spouse-absent," and "Married-AF-spouse" into a single category, "Other." We also grouped the "Age" and "Hours-per-week" attributes.



(a) Work-class

(b) Marital-status

(c) Education

(d) Occupation

Figure 5.3: Distribution of categorical features over each category grouped by target class.

5.4.3 Global Surrogate Model

In this section, we describe our results when utilizing an HT as a global surrogate model to interpret the black-box model generated by our MQ-OFL algorithm. The HT provides us with valuable insight into the model by constructing a decision tree that highlights the most important features, with the higher-level branches indicating their significance. The rules regarding the range of these features provides a trace of the decision-making process. Therefore, our use of HT enables us to better understanding the model's behaviour and decision-making process. Recall that training a surrogate model is a model-agnostic approach that does not require any knowledge about the inner workings of the MQ-OFL algorithm, requiring only access to data and the prediction function [29]. To create a surrogate model for the model M created by MQ-OFL trained on dataset X with target

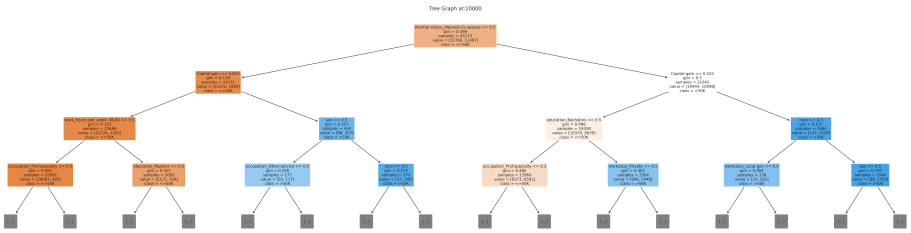
y, we have followed these steps [115]:

- Generated predictions for the training dataset X using the model M created by MQ-OFL to obtain $\hat{y}$.
- Created a new dataset by replacing the target column y in X with the predicted values $\hat{y}$.
- Selected HT as an interpretable model to be the surrogate model.
- Trained the HT interpretable model on the new dataset with the predicted values $\hat{y}$ as the target variable.
- The trained model is the surrogate and can be used to gain insights into how the MQ-OFL algorithm makes its predictions.

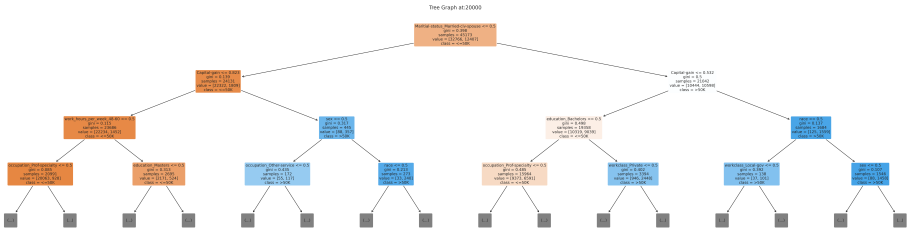
We generated HTs at intervals of $T = \{500, 5,000 \text{ and } 10,000\}$ to assess whether any changes occurred in the construction of the surrogate model. However, as the Adult data stream is stationary and does not exhibit concept drift, the importance value of attributes selected for the split remains in the same order for the first three depths. However, this order may change at deeper levels as the decision becomes more specific. At the illustrated depth, we observe an increase in the number of instances in each subtree as the tree grows. We set $T = 10,000$ to display the trees in this section. Figure 5.4 depicts these trees.

Recall that the HT here employs a Gini impurity value of 5.1 to split the subtrees. The trees in Figure 5.4 begin by splitting on the "Marital-status" feature, followed by splitting on "Capital gain" on both sides. However, the subsequent features used for splitting differ between the left and right subtrees. On the left subtree, the features "Works-hours-per-week" and "Sex" are used for splitting, while on the right subtree, "Education" and "Ethnicity" are used. The first split on "Marital Status" divides the instances into 32,766 cases with income "$<= 50K$" and 12,407 cases with income "$> 50K$". Although this split incorrectly assigns 1,205 cases to the negative class based on this feature, "Marital Status" is still chosen as the best attribute for the first split among all other attributes. Figure 5.5 shows the final tree. We will explain and analyze the behaviour of our fairness model through this tree.

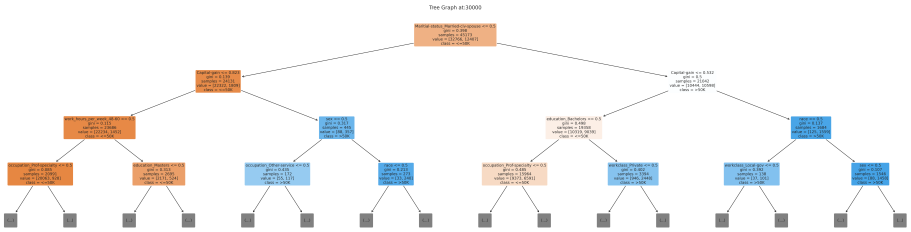
Figure 5.4: Trees of each 10,000 time steps in Adult data



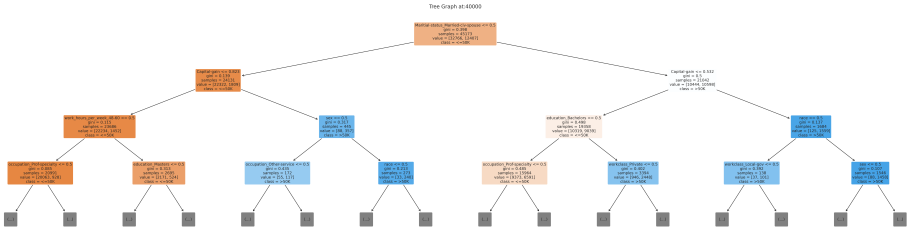
(a) Time step $10,000$ dataset



(b) Time step $20,000$ dataset



(c) Time step $30,000$ dataset



(d) Time step $40,000$ dataset

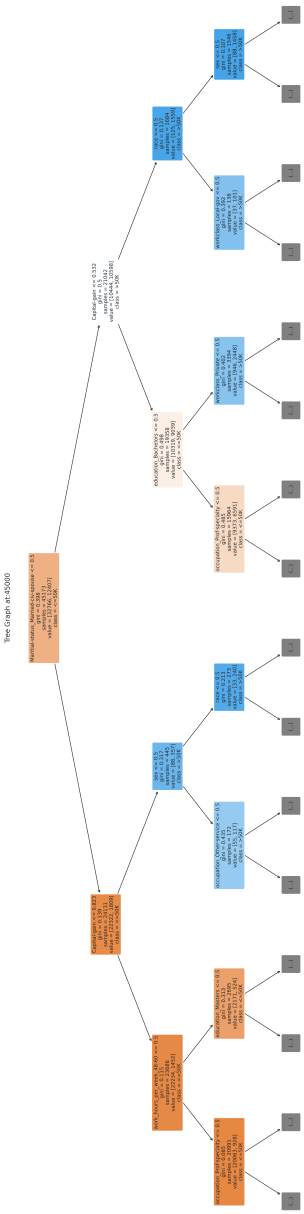


Figure 5.5: Final decision tree.

In Figure 5.5, the feature being used at each decision level and the corresponding Gini values for the condition are displayed. The tree begins with "Marital Status" as it has the lowest impurity among the other features, with a Gini value of 0.398. Since one-hot encoding was used on categorical features, each category of "Marital Status" was assigned to a binary vector, where 1 represents that category and 0s represents the other categories. For example, if the individual is not married to a civilian spouse, the value of the binary vector for that category will be 0 (which is less than 0.5) and the individual will move to the left subtree; otherwise they will move to the right subtree.

Moreover, each node of the tree displayed in Figure 5.5 provides more information. The "samples" line indicates the number of instances in the current node. In the first node, there is the "samples = 45,173" tree or subtree. Digging into the depth branches of the tree reduces the number of instances in each node as the tree splits the data. The "class" shows the classification result of the node, where the colour orange represents class "$<= 50K$" and blue represents class "$> 50K$". The intensity of the colour corresponds to the number of instances in that class in the specific node. The "value" line displays the number of instances in each class at that node. For example, in the first node, "value = [32766, 12407]" indicates that 32,766 instances belong to class "$<= 50K$" and 12,407 instances belong to class "$> 50K$". Since the majority belongs to class "$<= 50K$" at this node, the colour is orange.

Another example is the first right subtree of Figure 5.5. The feature "Capital-gain" has a Gini value of 0.5, indicating that the decision tree can divide instances almost equally among both class labels. Specifically, this node assigns $10,444$ instances with a "capital-gain" less than 0.532 to the left subtree, where the class label is "$<= 50K$", and assigns $10,598$ instances to the right subtree, where the class label is "$> 50K$". This division confirms that both classes have a similar number of instances in this node, which is why the node is white. Darker colors indicate higher purity in the node/leaf, and the goal is to have a Gini score of 0, which means that the instances in each leaf belong to a single class.

5.4.4 Local Surrogate Model

In these two sets of experiments, we examine a subset of selected instances in the data and analyze their features and how they impact the model's prediction. The instances are selected from protected groups that may have a biased prediction. By understanding how each feature impacts the prediction of the model created by MQ-OFL, we gain insight into its decision-making process and identify potential biases or inaccuracies.

In the first set of experiments, we utilized the previously introduced LIME model to extract explanations for individual predictions. This local model perturbs random instances

near the selected instance and trains an interpretable model locally on this new dataset. Recall that we employed a DT as our interpretable algorithm in the LIME model.

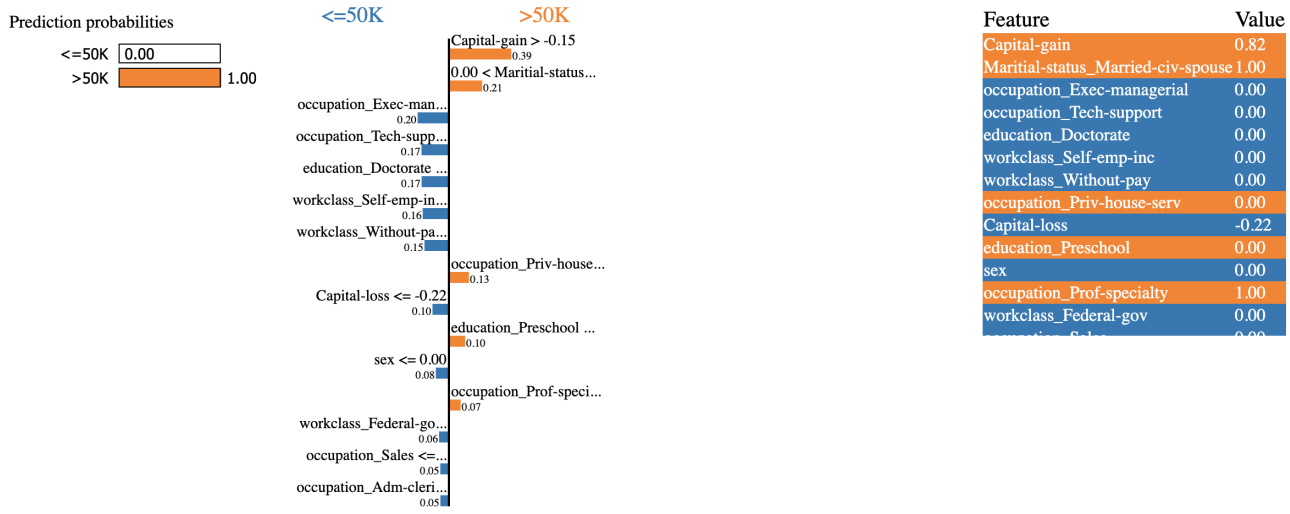


Figure 5.6: LIME: Explaining the MQ-OFL algorithm's individual predictions to determine if an individual earns more or less than 50K per year. The bar chart represents the importance given to the most-relevant features. The colour indicates to which class each feature contributes (orange for "$> 50K$", blue for "$<= 50K$").

First, we consider an unseen individual that is actually earning more than "50K" and is predicted to do so. In Figure 5.6, we note that the DT model has predicted this individual to have a 100% probability of class "$> 50K$". In the middle, we observe that LIME can explain this prediction using each feature of the data point of interest, in order of importance. According to LIME, the example's value of "> 0.15" for "Capital gain" makes them more likely to earn more than "50K." Because the individual is married, they are more likely to earn more than "$50K$". On the other hand, because the example is female, they are more likely to assign to target "$<= 50K$"; however, because our method treats both groups fairly, it still predicted the opposite. Considering all the individual's features (on the right panel), the example was predicted to earn more than "$50K$". These three observations fit our intuition and our knowledge of each individual. Knowing this, we are more confident that the model makes predictions correctly.

The second set of experiments regarding local surrogate models used the HT incremental learning algorithm trained against a balanced subset of instances and as created by the MQ-OFL algorithm. This approach differs from LIME; instead of perturbing random

examples, we select a batch when all queues are full (i.e., we have the same number of examples for all classes). The new dataset is used to train the HT algorithm, and we use HTs to evaluate the model's performance using the original data instances. In the online learning context, we partially utilize HT's training capability and test it with incoming instances to explain the decision-making process of our black-box model for an individual. We visualize the resulting partial tree to gain insights into the path of the decisions being made. We followed the steps outlined in Section 5.3.2 to construct the surrogate model and generated predictions on a training dataset of size 500 using our MQ-OFL model. Finally, we replaced the training dataset's target variables with the predicted values.

Table 5.4: Protected test example 1.

Features	Encoded Value
Age	age_group_45-54
WorkClass	workclass_Self-employed
Education	education_Masters
Marital-status	Marital-status_Other
Occupation	occupation_Exec-managerial
Ethnicity	White
Sex	Female
Capital-gain	0.112
Capital-loss	0
Hours-per-week	work_hours_per_week_48-60
Country	country_United-States
Income	$> 50K$

Employing the HT as our surrogate model, we trained it with an incremental learning setting with each arriving instance from the training dataset. Local surrogate models are useful for identifying and understanding specific aspects of the black-box model's behaviour in a particular region of the input space, so the goal of this experiment was to investigate whether a random test example of upcoming instances from protected groups would be classified correctly. Figure 5.7 illustrates the trained tree.

Subsequently, we chose one instance from each of the protected groups and traced the path from the starting node to the decision leaf. Table 5.4 details the first example from the protected group (Sex=Female).

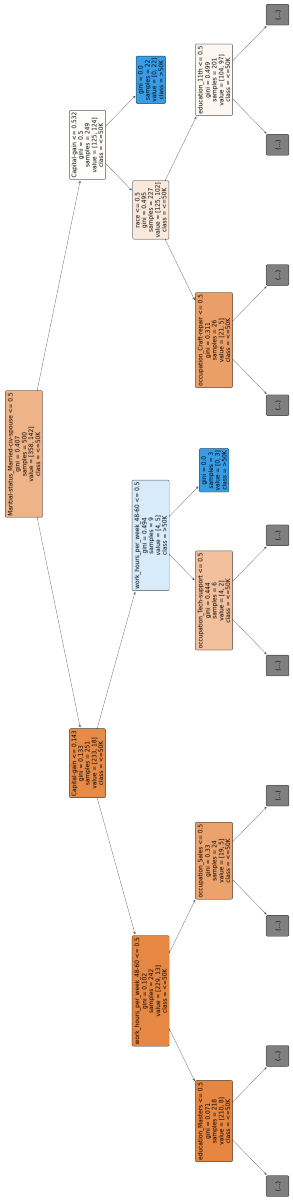


Figure 5.7: HT surrogate model.

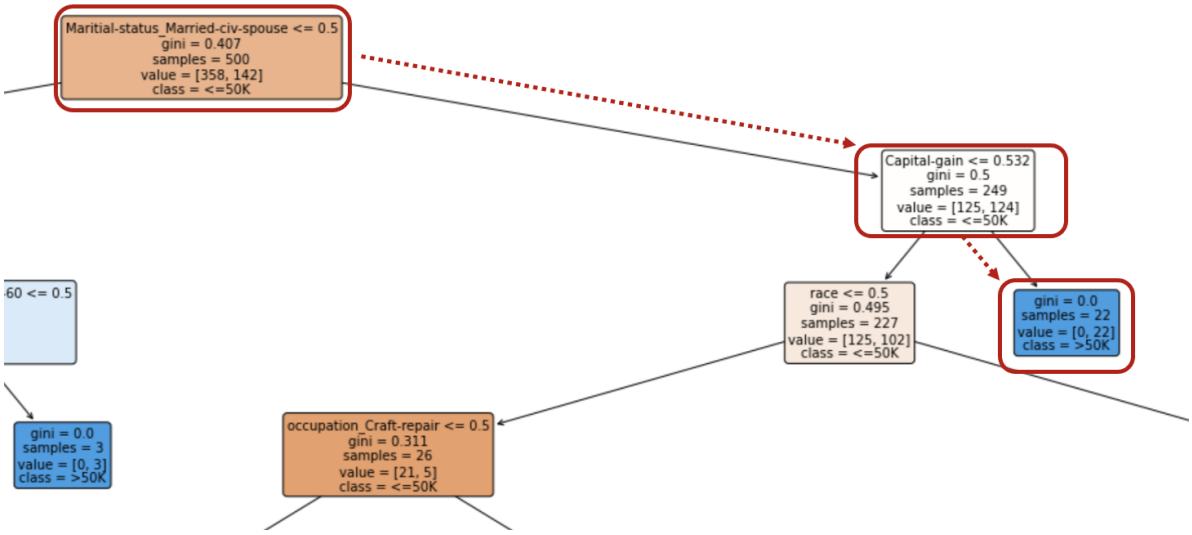


Figure 5.8: First example decision path through HT surrogate model.

In Figure 5.8, the decision-tree path starts at the left subtree because the individual's marital status is less than 0.5, which means that they are not married to a civilian spouse. The next node in the left subtree is based on the "Capital-gain" condition. Since the individual's capital gain is 0.112, which is less than the condition value, we follow the left subtree again. The next condition is the number of hours worked per week, which falls between 46 and 60 hours, so we move to the right subtree. However, because the individual is not working in "sales" in the next step, the condition fails, and we move back to the left subtree. The individual's age falls in the 45 to 54 age group, and the condition passes, so we move to the right subtree. In this case, the instances in the node are equally divided between the two target classes, so the node is represented by a white colour. Based on the feature details, the individual has a "Master's" degree, so we move to the right node, and the predicted label is over $50K$, which matches the ground truth, and we have a successfully classified instance.

Table 5.5: Protected test example 2.

Features	Encoded Value
Age	age_group_35-44
WorkClass	workclass_Private
Education	Education_High-school graduate
Marital-status	Marital_Married-civ spouse
Occupation	occupation_Other-service
Ethnicity	Black
Sex	Female
Capital-gain	0.876
Capital-loss	0
Hours-per-week	work_hours_per_week_32-48
Country	country_United-States
Income	$> 50K$

The next test example belongs to the intersection of both protected groups with "Sex=female" and "Ethnicity=Black," and the details are shown in Table 5.5. Following the decision path in Figure 5.9, the first node directs the decision to the right subtree, because the individual is married to a civilian spouse (its assigned vector is 1). Based on the "Capital-gain" condition, the individual has a capital gain of 0.876, which is greater than the defined condition value. Therefore, we move to the right subtree again, where we have the highest purity with a Gini value of 0, and all 22 instances belong to the class "$> 50K$". This result matches the dataset's ground truth label.

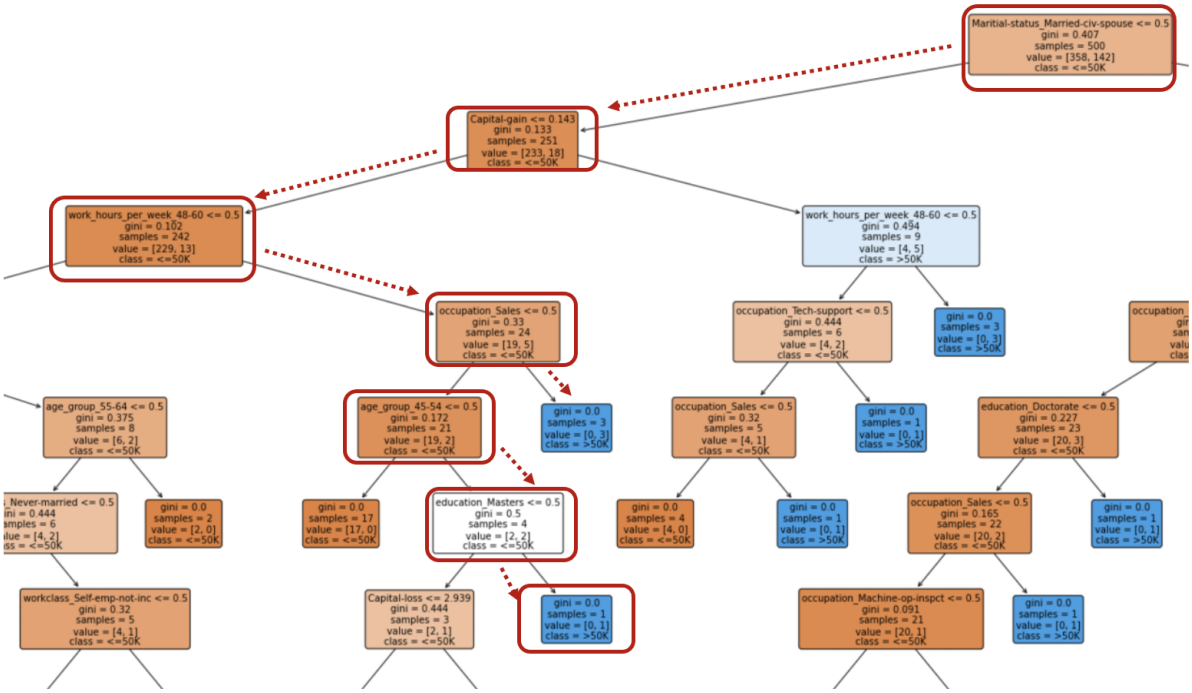


Figure 5.9: Second example decision path through HT surrogate model.

5.4.5 Discussion

One of the advantages of the interpretable analysis of the MQ-OFL online learning algorithm presented here is that it provides a trustworthy machine learning solution when applied to the Adult data stream. Specifically, the MQ-OFL algorithm explores its decisions using pre-modelling explainability and interpretation through surrogate models. Understanding the characteristics of each individual and their sensitive attributes is crucial to create a fairness-aware model. The pre-modelling experiment helps us to understand the challenges posed by discriminated, imbalanced datasets. We showed that the Adult dataset is imbalanced on the income-class label, which affects the classification performance of the positive class.

Additionally, the prevalence concept, which refers to the combination of income and sensitive attributes, highlights that imbalanced data can lead to unfair and biased decision-making. When a minority class (positive) is combined with unprivileged groups of individuals (female or Black), certain groups are disproportionately affected by the black-box model's decisions. For example, if a loan application model is trained on the Adult dataset

and combines individuals with income less than, or equal to, 50K with individuals belonging to a certain ethnicity or biological sex, it may result in biased decisions that unfairly disadvantage that particular group.

Recall that the MQ-OFL algorithm aims to maximize accuracy across the entire dataset while maintaining equal instances from each group, using a queue-based method to overcome discriminated and imbalanced data. We used surrogate models to investigate how our black-box model makes decisions under these circumstances. The global and local surrogate models were created using HT and LIME models for analysis. We utilized an online setting to construct the HTs (global and local), and employed the Gini impurity criterion to highlight the significant features based on the predictions of the black-box model at the different levels of the tree. Additionally, we employed the LIME model to focus on crucial features for predicting an individual with sensitive attributes. The chosen individual was most affected by "Capital gain" followed by "Marital status." However, the rank of the features was different in the local surrogate model created by HT. One of the chosen individuals from the protected groups had "Marital status" and "Capital gain" features on top of the tree, sequentially. The other one was predicted by "Marital status" followed by "Capital gain," "Work hours," "Occupation," "Age," and "Education." However, both models showed that even in severely biased cases, our MQ-OFL algorithm could have fair outcomes, which are explained through the surrogate models.

In conclusion, we state that local and global surrogate models can provide better insights into our complex fairness-aware MQ-OFL algorithm by making it more interpretable.

5.4.6 Accuracy versus Fairness Trade-off

Recall that the Adult dataset is class imbalanced and discriminated. The class imbalance issue arises from the dominant proportion of individuals with a yearly income of below $50K$, causing the HT to learn this class more accurately. Furthermore, there are two sensitive attributes, namely sex and ethnicity, that require discrimination-free decision-making.

The results obtained from the HT built with MQ-OFL algorithm at each time step, as shown in Table 5.6, indicate an improvement in accuracy from 84% at time 5000 to 91% in the final tree. The F1-Score, which provides a better representation of the minority class, remained consistent at 80.69%. The G-mean score also increased from 78% to 81%. It is important to consider that the reliability of this model may vary depending on the nature of the dataset. In this particular case, our fairness algorithm successfully reduced the discrimination level to 15.3% compared to the baseline HT model with a discrimination level of 22.6%. Therefore, the model's fairness performance is 7.3% higher than the baseline. One noteworthy aspect of our methodology is its transferability feature, which means that

Table 5.6: Accuracy-vs-discrimination of learning methods.

Time steps	Accuracy	G-mean	F-measure	Discrimination
5000	0.84	0.78	0.83	15.3
10000	0.86	0.78	0.82	15.4
15000	0.87	0.80	0.80	15.6
20000	0.86	0.81	0.81	15.4
25000	0.88	0.81	0.82	16.1
30000	0.89	0.82	0.82	15.8
35000	0.90	0.82	0.80	15.3
40000	0.90	0.83	0.82	15.3
Final	0.91	0.81	0.80	15.3

the method is not limited to a single dataset but can be applied to other case studies as well [139]. Additionally, we demonstrated that our fairness model, utilizing both local and global surrogate models, exhibits reliability and internal consistency in generating accurate and fair decision paths. In the next section, we provide comparative results with other methods.

5.4.7 Accuracy and Fairness of Algorithms

In this section, we present the results of our investigation into different fairness approaches. Recall that, we utilize the HT classifier as our baseline, without taking into account class imbalance and discrimination. Table 5.7 provides a comparison of the MQ-OFL, FAHT, and FEAT algorithms, together with the HT baseline method. The table showcases the results of our comparative analysis of these three algorithms. It is evident from these results that our model is capable of reducing discrimination to a lower level while maintaining comparable levels of accuracy. The same observation holds true for the F-measure, where MQ-OFL consistently yields higher results, particularly in managing class imbalance, leading to improvements in this measure. When evaluating accuracy and discrimination, it becomes apparent that MQ-OFL effectively mitigates unfair outcomes and maintains the highest performance in terms of G-mean and F-measure. MQ-OFL demonstrates the fairest results when considering two sensitive attributes, exhibiting the least discrimination score. Following MQ-OFL, FEAT performs well and is specifically designed for enhanced fairness-aware learning, with an added capability to adapt to concept drift in non-stationary discriminated data streams. Specifically, on the Adult dataset, MQ-OFL achieves a discrimination score of 15.12% and 1.34%, with a slight reduction in accuracy of 1.02% and

2.10%, respectively. These results suggest that our MQ-OFL method, which seamlessly integrates the merit of fair data into classifiers, yields a model that is driven by both accuracy and fairness considerations.

MQ-OFL achieves the most significant reduction in discrimination, while HT yields the poorest fairness results. This outcome is not surprising, as the exclusive focus on accuracy-oriented tree construction and the inherent discrimination bias of historical data may lead to the omission of fairness considerations during the construction of HT, even if the G-Mean results are slightly higher. Interested readers are referred to [139] for an extensive comparison of these algorithms against multiple streams.

Table 5.7: Accuracy-vs-discrimination of learning methods [139].

Method	Discrimination	G-mean	F-measure
MQ-OFL	**15.12**	83.05	**80.69**
FEAT	16.23	81.83	75.69
FAHT	17.40	79.07	71.66
HT	22.60	**83.91**	72.90

5.5 Summary

The field of explainable and interpretable machine learning is a developing area of study that aims to create machine learning models that are transparent, interpretable, and accountable to humans. Recent research has shown that making fair decisions requires a comprehensive understanding of the algorithms and the models they are creating, and machine learning explanations can help pinpoint variables that may contribute to unfair outcomes. This chapter explores the correlation between explainability and fairness-aware online learning algorithms. In this study, we utilized a specific dataset as a case study to explore the aspects of explainability and interpretability within the MQ-OFL method. We acknowledge that, while the steps and techniques employed in our analysis can be applied to other datasets that exhibit discrimination, it is crucial to acknowledge that each dataset may require a unique approach to explain and interpret its own attributes effectively. We emphasize the importance of explaining fairness as a crucial aspect of responsible machine learning algorithm use and trustworthy decision-making.

Next, Chapter 6 offers our conclusions and a discussion of future work.

Chapter 6

Conclusions and Future Work

Multi-class imbalance in online learning from evolving streams poses an important challenge in numerous domains. Specifically, novel algorithms are needed to face multi-majority or multi-minority classes, notably where the skew may evolve over time. Further, recent advances in machine learning applications have produced capabilities for decision-making tasks, such as allocating resources to people, risk assessment of individual's behavior or other applications that directly affect our everyday lives. However, the resultant models constructed by these techniques can also potentially discriminate and thus lead to unfair decisions. This issue gains more complexity when the data are evolving with skewed distribution, and where there are multiple sensitive attributes. This thesis proposal addresses the combined problem of online machine learning from evolving multi-class skewed distributions while aiming to be fairness-aware against multiple sensitive groups.

6.1 Contributions

The contributions of this thesis are as follows:

6.1.1 Online Multi-class Imbalanced Learning

We addressed the challenge of online learning from evolving multi-class imbalanced data streams, which are susceptible to concept drifts, through the introduction of the DynaQ algorithm. An advantage of our DynaQ method is that it operates independently of a base classifier, thus providing a general framework for dealing with evolving multi-class streams.

The DynaQ algorithm maintains queues for each of the classes and thus implicitly balances the data, while befitting proper resampling rates. In this way, all classes are equally represented during online learning. Concept drift is handled by considering individual classes, which implies that drifts in minority classes are detected as soon as enough instances have been collected. Our experimental results showed the benefits of our approach, and we determined that the DynaQ method is a promising candidate for further experimentation.

6.1.2 Online Fairness-aware Imbalanced Learning

Furthermore, we introduced the MQ-OFL approach for fairness-aware learning from evolving streams with the class imbalance and concept drifts. We illustrated that our technique was able to maintain predictive performance with low discrimination scores over the course of evolving streams. Moreover, our MQ-OFL method facilitated two or more sensitive attributes by customizing a classifier for each protected group. Our experimental evaluation showed that our approach outperforms other methods in a variety of datasets with respect to both predictive performance and fairness preservation. That is, our class-imbalance-oriented approach effectively learns both sensitive attributes while achieving good predictive performance for both minority and majority classes.

6.1.3 Explainability and Interpretability in Fairness-aware Online Learning

The final contribution focused on explainable and interpretable machine learning algorithms. This emerging and rapidly developing field seeks to create machine learning models that are transparent, interpretable, and accountable to humans. The investigations demonstrated that making fair decisions requires a thorough contextual understanding, and machine learning explanations can aid in identifying potential features that drive unfair outcomes. Our approach generated predictions from black-box models and created a transparent model on these predictions and the original features. Our research explored the relationship between explainability and fairness-aware online learning algorithms. We have identified the importance of explaining fairness as a crucial component in the responsible use of machine learning algorithms and trustworthy decision-making. Our investigation also highlighted the challenges related to the relationship between explainability and fairness-aware models including the number of sensitive attributes, gerrymandering

interpretation, the trade-off between fairness and model accuracy, and incremental online learning concepts. Next, we detail our future research directions.

6.2 Future Work

6.2.1 Extended Multi-class Imbalance Online Learning

To date, numerous studies have been conducted on imbalanced learning [44]. However, only a limited number of these studies [5, 100] have addressed solutions suitable for online multi-class streams that experience varying degrees of imbalance and concept drift. In such evolving streams, minority classes may become the majority and vice versa, while various degrees of concept drift occurs. In Chapter 3, we present the DynaQ algorithm, which focuses on mitigating the impact of concept drift by maintaining a queue of recent instances for each class. This allows us to retrain the model after a drift occurs. In our future work, we plan to investigate the expansion of our algorithm to handle extended scenarios where different types of concept drifts (virtual or real) simultaneously affect imbalance ratios [142, 158]. Furthermore, we intend to incorporate comprehensive analyses, including assessments of memory and time consumption, into our online incremental learning approach.

6.2.2 Multi-class Fairness-aware Learning

While there have been many definitions of, and approaches to, fairness in the literature, more study is needed [127]. Notably, ensuring fairness on evolving streams is still open to a number of research opportunities. The study in Chapter 4 proposed the MQ-OFL approach for fairness-aware stream classification. MQ-OFL is able to maintain a trade-off between performance with low discrimination scores over the course of imbalanced streams. Interesting next steps include extending our work to address multi-class online classification with evolving concepts. It would also be worthwhile to investigate additional fairness metrics to further explore the behavior of the subgroups over an evolving stream.

6.2.3 Fairness-aware Metrics and Algorithms

Recall that there are three categories (pre-processing, post-processing, and in-processing) for fairness-aware algorithm design. Pre-processing mechanisms can be advantageous since they are not dependent on a specific classification algorithm and can be applied with any

classification algorithm. However, they may harm model explainability [12] of the results since there is a variety in the level of accuracy obtained with each classifier. Similar to pre-processing mechanisms, post-processing methods are also classifier-agnostic. However, post-processing approaches could obtain unfair results if we blindly apply them at the end of the learning process, since they deliberately damage accuracy for some individuals in favor of others (mostly the members in unprotected groups). Specifically, post-processing algorithms may change the outcomes of two individuals who are similar across all features except for the group to which they belong to keep the threshold balanced [14].

In-processing methods involve directly addressing the trade-off between accuracy and fairness during the processing phase. However, these techniques are closely tied to the selected algorithm itself. As a result, the choice of fairness-aware method depends on various factors, such as the availability of ground truth data, the stage at which fairness-aware methods are incorporated, the specific fairness parameter chosen, and the classification algorithm designed, which may vary across different applications [44]. In our future work, we aim to examine the interplay between these trade-offs. Additionally, we intend to develop a framework that maintains multiple fairness metrics, allowing for the development of fairness mechanisms that preserve accuracy and fairness over streams.

References

[1] Amina Adadi and Mohammed Berrada. Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). *IEEE access*, 6:52138–52160, 2018.

[2] Julius Adebayo and Lalana Kagal. Iterative orthogonal feature projection for diagnosing bias in black-box models. *arXiv preprint arXiv:1611.04967*, 2016.

[3] Sina Aghaei, Mohammad Javad Azizi, and Phebe Vayanos. Learning optimal and fair decision trees for non-discriminative decision-making. *In Proceedings of the AAAIConference on Artificial Intelligence*, 33:1418–1426, 2019.

[4] Astha Agrawal, Herna L Viktor, and Eric Paquet. SCUT: Multi-class imbalanced data classification using SMOTE and cluster-based undersampling. In *2015 7Th international joint conference on knowledge discovery, knowledge engineering and knowledge management (IC3k)*, volume 1, pages 226–234. IEEE, 2015.

[5] Gabriel Aguiar, Bartosz Krawczyk, and Alberto Cano. A survey on learning from imbalanced data streams: taxonomy, challenges, empirical study, and reproducible experimental framework. *arXiv preprint arXiv:2204.03719*, 2022.

[6] Mohiuddin Ahmed. Data summarization: a survey. *Knowledge and Information Systems*, 58(2):249–273, 2019.

[7] Rahbar Ahsan, Wei Shi, and Jean-Pierre Corriveau. Network intrusion detection using machine learning approaches: Addressing data imbalance. *IET Cyber-Physical Systems: Theory & Applications*, 7(1):30–39, 2022.

[8] Alliance Canada Compute. Available Resources, 2022. Last access on November 2022, https://alliancecan.ca.

[9] Ehsan Aminian, Rita P Ribeiro, and João Gama. Chebyshev approaches for imbalanced data streams regression models. *Data Mining and Knowledge Discovery*, 35:2389–2466, 2021.

[10] Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, Siham Tabik, Alberto Barbado, Salvador García, Sergio Gil-López, Daniel Molina, Richard Benjamins, et al. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information fusion*, 58:82–115, 2020.

[11] Arthur Asuncion and David Newman. Census income data set, 2007. UCI Machine Learning Repository. Retrieved from https://archive-beta.ics.uci.edu/ml/datasets/adult.

[12] Hubert Baniecki, Wojciech Kretowicz, Piotr Piatyszek, Jakub Wisniewski, and Przemyslaw Biecek. Dalex: Responsible machine learning with interactive explainability and fairness in Python. *The Journal of Machine Learning Research*, 22(1):9759–9765, 2021.

[13] Ricardo Barandela, Rosa Maria Valdovinos, and José Salvador Sánchez. New applications of ensembles of classifiers. *Pattern Analysis & Applications*, 6:245–256, 2003.

[14] Solon Barocas, Moritz Hardt, and Arvind Narayanan. Fairness in machine learning. *Nips tutorial*, 1:2017, 2017.

[15] Barry Becker and Ronny Kohavi. Adult. UCI Machine Learning Repository, 1996. DOI: https://doi.org/10.24432/C5XW20.

[16] Colin Bellinger, Shiven Sharma, Nathalie Japkowicz, and Osmar R Zaïane. Framework for extreme imbalance classification: SWIM—sampling with the majority class. *Knowledge and Information Systems*, 62:841–866, 2020.

[17] Alessio Bernardo and Emanuele Della Valle. SMOTE-OB: Combining SMOTE and Online Bagging for Continuous Rebalancing of Evolving Data Streams. In *2021 IEEE International Conference on Big Data (Big Data)*, pages 5033–5042. IEEE, 2021.

[18] Alessio Bernardo and Emanuele Della Valle. VFC-SMOTE: very fast continuous synthetic minority oversampling for evolving data streams. *Data Mining and Knowledge Discovery*, 35(6):2679–2713, 2021.

[19] Alessio Bernardo and Emanuele Della Valle. An extensive study of c-smote, a continuous synthetic minority oversampling technique for evolving data streams. *Expert Systems with Applications*, 196:116630, 2022.

[20] Albert Bifet, Gianmarco de Francisci Morales, Jesse Read, Geoff Holmes, and Bernhard Pfahringer. Efficient online evaluation of big data stream classifiers. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 59–68, 2015.

[21] Albert Bifet and Eibe Frank. Sentiment knowledge discovery in twitter streaming data. In *International conference on discovery science*, pages 1–15. Springer, 2010.

[22] Albert Bifet and Ricard Gavalda. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM international conference on data mining*, pages 443–448. SIAM, 2007.

[23] Albert Bifet and Ricard Gavalda. Adaptive learning from evolving data streams. In *Advances in Intelligent Data Analysis VIII: 8th International Symposium on Intelligent Data Analysis, IDA 2009, Lyon, France, August 31-September 2, 2009. Proceedings 8*, pages 249–260. Springer, 2009.

[24] Albert Bifet, Ricard Gavalda, Geoffrey Holmes, and Bernhard Pfahringer. *Machine learning for data streams: with practical examples in MOA*. MIT press, 2023.

[25] Albert Bifet and Richard Kirkby. Data stream mining a practical approach.(2009). *Citeseer: The University of Waikato*, pages 68–69, 2009.

[26] Malthe K Bisbo and Bjørk Hammer. Efficient global structure optimization with a machine-learned surrogate model. *Physical review letters*, 124(8):086102, 2020.

[27] Ekaba Bisong et al. *Building machine learning and deep learning models on Google cloud platform*. Springer, 2019.

[28] Jock Blackard. Covertype. UCI Machine Learning Repository, 1998. DOI: https://doi.org/10.24432/C50K5N.

[29] Alberto Blanco-Justicia and Josep Domingo-Ferrer. Machine learning explainability through comprehensible decision trees. In *Machine Learning and Knowledge Extraction: Third IFIP TC 5, TC 12, WG 8.4, WG 8.9, WG 12.9 International Cross-Domain Conference, CD-MAKE 2019, Canterbury, UK, August 26–29, 2019, Proceedings 3*, pages 15–26. Springer, 2019.

[30] Barbara Bobowska, Jakub Klikowski, and Michał Woźniak. Imbalanced data stream classification using hybrid data preprocessing. In *Machine Learning and Knowledge Discovery in Databases: International Workshops of ECML PKDD 2019, Würzburg, Germany, September 16–20, 2019, Proceedings, Part II*, pages 402–413. Springer, 2020.

[31] Mohamed-Rafik Bouguelia, Yolande Belaïd, and Abdel Belaïd. An adaptive streaming active learning strategy based on instance weighting. *Pattern Recognition Letters*, 70:38–44, 2016.

[32] Clara Bove, Jonathan Aigrain, Marie-Jeanne Lesot, Charles Tijus, and Marcin Detyniecki. Contextualization and exploration of local feature importance explanations to improve understanding and satisfaction of non-expert users. In *27th international conference on intelligent user interfaces*, pages 807–819, 2022.

[33] Paula Branco. Exploring the impact of resampling methods for malware detection. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 3961–3968. IEEE, 2020.

[34] Paula Branco, Luís Torgo, and Rita P Ribeiro. A survey of predictive modeling on imbalanced domains. *ACM computing surveys (CSUR)*, 49(2):1–50, 2016.

[35] Leo Breiman. Bagging predictors. *Machine learning*, 24:123–140, 1996.

[36] Dariusz Brzezinski, Leandro L Minku, Tomasz Pewinski, Jerzy Stefanowski, and Artur Szumaczuk. The impact of data difficulty factors on classification of imbalanced and concept drifting data streams. *Knowledge and Information Systems*, 63(6):1429–1469, 2021.

[37] Dariusz Brzezinski, Jerzy Stefanowski, Robert Susmaga, and Izabela Szczech. On the dynamics of classification measures for imbalanced and streaming data. *IEEE transactions on neural networks and learning systems*, 31(8):2868–2878, 2019.

[38] Joy Buolamwini and Timnit Gebru. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR, 2018.

[39] Canada Government. Historic climate data from environment and climate change Canada, 2020. https://climate.weather.gc.ca/historical_data/search_historic_data_e.html.

[40] Alberto Cano and Bartosz Krawczyk. Kappa updated ensemble for drifting data stream mining. *Machine Learning*, 109:175–218, 2020.

[41] Alberto Cano and Bartosz Krawczyk. ROSE: Robust online self-adjusting ensemble for continual learning on imbalanced drifting data streams. *Machine Learning*, 111(7):2561–2599, 2022.

[42] Pierluigi Casale, Oriol Pujol, and Petia Radeva. Personalization and user verification in wearable systems using biometric walking patterns. *Personal and Ubiquitous Computing*, 16:563–580, 2012.

[43] Vitor Cerqueira, Luis Torgo, Paula Branco, and Colin Bellinger. Automated imbalanced classification via layered learning. *Machine Learning*, 112(6):2083–2104, 2023.

[44] Nitesh V Chawla. Data mining for imbalanced datasets: An overview. *Data mining and knowledge discovery handbook*, pages 875–886, 2010.

[45] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. SMOTE: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[46] You-Shyang Chen. An empirical study of a hybrid imbalanced-class DT-RST classification procedure to elucidate therapeutic effects in uremia patients. *Medical & biological engineering & computing*, 54(6):983–1001, 2016.

[47] Ireneusz Czarnowski. Weighted Ensemble with one-class Classification and Oversampling and Instance selection (WECOI): An approach for learning from imbalanced data streams. *Journal of Computational Science*, 61:101614, 2022.

[48] David Danks and Alex John London. Algorithmic Bias in Autonomous Systems. In *Ijcai*, volume 17, pages 4691–4697, 2017.

[49] J Derrac, S Garcia, L Sanchez, and F Herrera. Keel data-mining software tool: Data set repository, integration of algorithms and experimental analysis framework. *J. Mult. Valued Log. Soft Comput*, 17:255–287, 2015.

[50] Thomas G Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems: First International Workshop, MCS 2000 Cagliari, Italy, June 21–23, 2000 Proceedings 1*, pages 1–15. Springer, 2000.

[51] Alican Dogan and Derya Birant. A weighted majority voting ensemble approach for classification. In *2019 4th International Conference on Computer Science and Engineering (UBMK)*, pages 1–6. IEEE, 2019.

[52] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 71–80, 2000.

[53] Xibin Dong, Zhiwen Yu, Wenming Cao, Yifan Shi, and Qianli Ma. A survey on ensemble learning. *Frontiers of Computer Science*, 14:241–258, 2020.

[54] Dtreez Authors. dtreez: Decision Trees and Random Forests in Python, 2021. https://github.com/david-cortes/dtreez.

[55] Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77, 2019.

[56] Rudresh Dwivedi, Devam Dave, Het Naik, Smiti Singhal, Rana Omer, Pankesh Patel, Bin Qian, Zhenyu Wen, Tejal Shah, Graham Morgan, et al. Explainable AI (XAI): Core ideas, techniques, and solutions. *ACM Computing Surveys*, 55(9):1–33, 2023.

[57] Rob Eisinga, Tom Heskes, Ben Pelzer, and Manfred Te Grotenhuis. Exact p-values for pairwise comparison of Friedman rank sums, with application to comparing classifiers. *BMC bioinformatics*, 18(1):1–18, 2017.

[58] Boli Fang, Miao Jiang, Pei-yi Cheng, Jerry Shen, and Yi Fang. Achieving Outcome Fairness in Machine Learning Models for Social Decision Problems. In *IJCAI*, pages 444–450, 2020.

[59] Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C Prati, Bartosz Krawczyk, Francisco Herrera, Alberto Fernández, Salvador García, Mikel Galar, Ronaldo C Prati, et al. Learning from imbalanced data streams. *Learning from imbalanced data sets*, pages 279–303, 2018.

[60] Alberto Fernández, Salvador Garcia, Francisco Herrera, and Nitesh V Chawla. SMOTE for learning from imbalanced data: progress and challenges, marking the 15-year anniversary. *Journal of artificial intelligence research*, 61:863–905, 2018.

[61] Alberto Fernández, Victoria López, Mikel Galar, MaríA José Del Jesus, and Francisco Herrera. Analysing the classification of imbalanced data-sets with multiple classes: Binarization techniques and ad-hoc approaches. *Knowledge-based systems*, 42:97–110, 2013.

[62] Benjamin Fish, Jeremy Kun, and Ádám D Lelkes. A confidence-based approach for balancing fairness and accuracy. In *Proceedings of the 2016 SIAM international conference on data mining*, pages 144–152. SIAM, 2016.

[63] Sean LA Floyd and Herna L Viktor. Soft voting windowing ensembles for learning from partially labelled streams. In *New Frontiers in Mining Complex Patterns: 8th International Workshop, NFMCP 2019, Held in Conjunction with ECML-PKDD 2019, Würzburg, Germany, September 16, 2019, Revised Selected Papers 8*, pages 85–99. Springer, 2020.

[64] Yoav Freund. Boosting a weak learning algorithm by majority. *Information and computation*, 121(2):256–285, 1995.

[65] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[66] Jerome H Friedman and Bogdan E Popescu. Predictive learning via rule ensembles. 2008.

[67] Joao Gama. *Knowledge discovery from data streams*. CRC Press, 2010.

[68] Joao Gama, Pedro Medas, Gladys Castillo, and Pedro Rodrigues. Learning with drift detection. In *Advances in Artificial Intelligence–SBIA 2004: 17th Brazilian Symposium on Artificial Intelligence, Sao Luis, Maranhao, Brazil, September 29-Ocotber 1, 2004. Proceedings 17*, pages 286–295. Springer, 2004.

[69] Joao Gama, Raquel Sebastiao, and Pedro Pereira Rodrigues. Issues in evaluation of stream learning algorithms. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 329–338, 2009.

[70] Jean-Gabriel Gaudreault, Paula Branco, and João Gama. An analysis of performance metrics for imbalanced classification. In *International Conference on Discovery Science*, pages 67–77. Springer, 2021.

[71] Heitor Murilo Gomes, Jesse Read, Albert Bifet, Jean Paul Barddal, and João Gama. Machine learning for streaming data: state of the art, challenges, and opportunities. *ACM SIGKDD Explorations Newsletter*, 21(2):6–22, 2019.

[72] Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Franco Turini, Fosca Giannotti, and Dino Pedreschi. A survey of methods for explaining black box models. *ACM computing surveys (CSUR)*, 51(5):1–42, 2018.

[73] Hongyu Guo and Herna L Viktor. Learning from imbalanced data sets with boosting and data generation: the databoost-im approach. *ACM Sigkdd Explorations Newsletter*, 6(1):30–39, 2004.

[74] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29, 2016.

[75] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[76] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *The collected works of Wassily Hoeffding*, pages 409–426, 1994.

[77] Andreas Holzinger, Randy Goebel, Ruth Fong, Taesup Moon, Klaus-Robert Müller, and Wojciech Samek. *XXAI-BEYOND EXPLAINABLE AI: International Workshop, Held in Conjunction*, volume 13200. Springer Nature, 2022.

[78] Giles Hooker. Discovering additive structure in black box functions. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 575–580, 2004.

[79] Ayanna Howard and Jason Borenstein. The ugly truth about ourselves and our robot creations: the problem of bias and social inequity. *Science and engineering ethics*, 24:1521–1536, 2018.

[80] Johan Huysmans, Karel Dejaeger, Christophe Mues, Jan Vanthienen, and Bart Baesens. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. *Decision Support Systems*, 51(1):141–154, 2011.

[81] Vasileios Iosifidis and Eirini Ntoutsi. Adafair: Cumulative fairness adaptive boosting. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 781–790, 2019.

[82] Vasileios Iosifidis and Eirini Ntoutsi. Online Fairness-Aware Learning Under Class Imbalance. In *International Conference on Discovery Science*, pages 159–174. Springer, 2020.

[83] Vasileios Iosifidis, Thi Ngoc Han Tran, and Eirini Ntoutsi. Fairness-enhancing interventions in stream classification. In *Database and Expert Systems Applications: 30th International Conference, DEXA 2019, Linz, Austria, August 26–29, 2019, Proceedings, Part I 30*, pages 261–276. Springer, 2019.

[84] Małgorzata Janicka, Mateusz Lango, and Jerzy Stefanowski. Using information on class interrelations to improve classification of multiclass imbalanced data: a new resampling algorithm. *International Journal of Applied Mathematics and Computer Science*, 29(4):769–781, 2019.

[85] Nathalie Japkowicz and Mohak Shah. *Evaluating learning algorithms: a classification perspective*. Cambridge University Press, 2011.

[86] Faisal Kamiran, Toon Calders, and Mykola Pechenizkiy. Discrimination aware decision tree learning. In *2010 IEEE international conference on data mining*, pages 869–874. IEEE, 2010.

[87] Stamatis Karlos, Georgios Kostopoulos, and Sotiris Kotsiantis. A soft-voting ensemble based co-training scheme using static selection for binary classification problems. *Algorithms*, 13(1):26, 2020.

[88] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. Preventing fairness gerrymandering: Auditing and learning for subgroup fairness. In *International conference on machine learning*, pages 2564–2572. PMLR, 2018.

[89] Michael Kearns, Seth Neel, Aaron Roth, and Zhiwei Steven Wu. An empirical study of rich subgroup fairness for machine learning. In *Proceedings of the conference on fairness, accountability, and transparency*, pages 100–109, 2019.

[90] Jakub Klikowski and Michał Woźniak. Employing one-class SVM classifier ensemble for imbalanced data stream classification. In *Computational Science–ICCS 2020: 20th International Conference, Amsterdam, The Netherlands, June 3–5, 2020, Proceedings, Part IV 20*, pages 117–127. Springer, 2020.

[91] Emmanouil Krasanakis, Eleftherios Spyromitros-Xioufis, Symeon Papadopoulos, and Yiannis Kompatsiaris. Adaptive sensitive reweighting to mitigate bias in fairness-aware classification. In *Proceedings of the 2018 world wide web conference*, pages 853–862, 2018.

[92] Bartosz Krawczyk. Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, 5(4):221–232, 2016.

[93] Bartosz Krawczyk, Colin Bellinger, Roberto Corizzo, and Nathalie Japkowicz. Undersampling with Support Vectors for Multi-Class Imbalanced Data Classification. In *2021 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2021.

[94] Bartosz Krawczyk, Mikel Galar, Michał Woźniak, Humberto Bustince, and Francisco Herrera. Dynamic ensemble selection for multi-class classification with one-class classifiers. *Pattern Recognition*, 83:34–51, 2018.

[95] Bartosz Krawczyk, Leandro L Minku, João Gama, Jerzy Stefanowski, and Michał Woźniak. Ensemble learning for data stream analysis: A survey. *Information Fusion*, 37:132–156, 2017.

[96] Paweł Ksieniewicz. The prior probability in the batch classification of imbalanced data streams. *Neurocomputing*, 452:309–316, 2021.

[97] Mateusz Lango and Jerzy Stefanowski. What makes multi-class imbalanced problems difficult? An experimental study. *Expert Systems with Applications*, 199:116962, 2022.

[98] J Larson, S Mattu, L Kirchner, and J Angwin. Propublica COMPAS risk assessment data set (2016).

[99] Thibault Laugel, Xavier Renard, Marie-Jeanne Lesot, Christophe Marsala, and Marcin Detyniecki. Defining locality for surrogates in post-hoc interpretablity. *arXiv preprint arXiv:1806.07498*, 2018.

[100] Joffrey L Leevy, Taghi M Khoshgoftaar, Richard A Bauder, and Naeem Seliya. A survey on addressing high-class imbalance in big data. *Journal of Big Data*, 5(1):1–30, 2018.

[101] Agnieszka Lipska and Jerzy Stefanowski. The Influence of Multiple Classes on Learning Online Classifiers from Imbalanced and Concept Drifting Data Streams. *arXiv preprint arXiv:2210.08359*, 2022.

[102] Zachary C Lipton, Alexandra Chouldechova, and Julian McAuley. Does mitigating ml's disparate impact require disparate treatment. *arXiv preprint arXiv:1711.07076*, 2017.

[103] Weike Liu, Hang Zhang, Zhaoyun Ding, Qingbao Liu, and Cheng Zhu. A comprehensive active learning method for multiclass imbalanced data streams with concept drift. *Knowledge-Based Systems*, 215:106778, 2021.

[104] Viktor Losing, Barbara Hammer, and Heiko Wersing. Self-adjusting memory: How to deal with diverse drift types. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17)*, pages 4899–4903, 2017.

[105] Viktor Losing, Barbara Hammer, and Heiko Wersing. Incremental on-line learning: A review and comparison of state of the art algorithms. *Neurocomputing*, 275:1261–1274, 2018.

[106] Jie Lu, Anjin Liu, Fan Dong, Feng Gu, Joao Gama, and Guangquan Zhang. Learning under concept drift: A review. *IEEE transactions on knowledge and data engineering*, 31(12):2346–2363, 2018.

[107] Ning Lu, Guangquan Zhang, and Jie Lu. Concept drift detection via competence models. *Artificial Intelligence*, 209:11–28, 2014.

[108] Yang Lu, Yiu-Ming Cheung, and Yuan Yan Tang. Adaptive chunk-based dynamic weighted majority for imbalanced data streams with concept drift. *IEEE Transactions on Neural Networks and Learning Systems*, 31(8):2764–2778, 2019.

[109] Chaochao Luo, Zhiyuan Tan, Geyong Min, Jie Gan, Wei Shi, and Zhihong Tian. A novel web attack detection system for internet of things via ensemble classification. *IEEE Transactions on Industrial Informatics*, 17(8):5810–5818, 2020.

[110] Robert J Lyon, JM Brooke, Joshua D Knowles, and Benjamin W Stappers. Hellinger distance trees for imbalanced streams. In *2014 22nd International Conference on Pattern Recognition*, pages 1969–1974. IEEE, 2014.

[111] Samuel Madden. Intel Berkeley Research lab, 2004. Last access May 2023.

[112] Kleanthis Malialis, Christos Panayiotou, and Marios M Polycarpou. Queue-based resampling for online class imbalance learning. In *Artificial Neural Networks and Machine Learning–ICANN 2018: 27th International Conference on Artificial Neural Networks, Rhodes, Greece, October 4-7, 2018, Proceedings, Part I 27*, pages 498–507. Springer, 2018.

[113] Michel Marie and DEZA DEZA. *Encyclopedia of Distances.* Springer, 2018.

[114] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM computing surveys (CSUR)*, 54(6):1–35, 2021.

[115] Christoph Molnar. Interpretable machine learning. 2020.

[116] Jacob Montiel, Max Halford, Saulo Martiello Mastelini, Geoffrey Bolmier, Raphael Sourty, Robin Vaysse, Adil Zouitine, Heitor Murilo Gomes, Jesse Read, Talel Abdessalem, et al. River: machine learning for streaming data in python. *The Journal of Machine Learning Research*, 22(1):4945–4952, 2021.

[117] Jacob Montiel, Jesse Read, Albert Bifet, and Talel Abdessalem. Scikit-multiflow: A multi-output streaming framework. *The Journal of Machine Learning Research*, 19(1):2915–2914, 2018.

[118] Sérgio Moro, Paulo Cortez, and Paulo Rita. Bank marketing data set. *URL https://archive. ics. uci. edu/ml/datasets/Bank+ Marketing*, 2014.

[119] Paul K Mvula, Paula Branco, Guy-Vincent Jourdan, and Herna L Viktor. COVID-19 malicious domain names classification. *Expert Systems with Applications*, 204:117553, 2022.

[120] Olubukola M Olaitan and Herna L Viktor. SCUT-DS: Learning from multi-class imbalanced canadian weather data. In *Foundations of Intelligent Systems: 24th International Symposium, ISMIS 2018, Limassol, Cyprus, October 29–31, 2018, Proceedings 24*, pages 291–301. Springer, 2018.

[121] Luca Oneto and Silvia Chiappa. Fairness in machine learning. In *Recent trends in learning from data: Tutorials from the inns big data and deep learning conference (innsbddl2019)*, pages 155–196. Springer, 2020.

[122] Agustín Ortíz Díaz, José del Campo-Ávila, Gonzalo Ramos-Jiménez, Isvani Frías Blanco, Yailé Caballero Mota, Antonio Mustelier Hechavarría, Rafael Morales-Bueno, et al. Fast adapting ensemble: A new algorithm for mining data streams with concept drift. *The Scientific World Journal*, 2015, 2015.

[123] Nikunj C Oza and Stuart J Russell. Online bagging and boosting. In *International Workshop on Artificial Intelligence and Statistics*, pages 229–236. PMLR, 2001.

[124] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

[125] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[126] Ali Pesaranghader and Herna L Viktor. Fast hoeffding drift detection method for evolving data streams. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part II 16*, pages 96–111. Springer, 2016.

[127] Dana Pessach and Erez Shmueli. A review on fairness in machine learning. *ACM Computing Surveys (CSUR)*, 55(3):1–44, 2022.

[128] Robi Polikar. Ensemble based systems in decision making. *IEEE Circuits and systems magazine*, 6(3):21–45, 2006.

[129] S Priya and R Annie Uthra. Comprehensive analysis for class imbalance data with concept drift using ensemble based classification. *Journal of Ambient Intelligence and Humanized Computing*, 12:4943–4956, 2021.

[130] N Ranasinghe, A Ramanan, S Fernando, PN Hameed, D Herath, T Malepathirana, P Suganthan, M Niranjan, and S Halgamuge. Interpretability and accessibility of machine learning in selected food processing, agriculture and health applications. *arXiv preprint arXiv:2211.16699*, 2022.

[131] Jesse Read, Albert Bifet, Bernhard Pfahringer, and Geoff Holmes. Batch-incremental versus instance-incremental learning in dynamic and evolving data. In *Advances in Intelligent Data Analysis XI: 11th International Symposium, IDA 2012, Helsinki, Finland, October 25-27, 2012. Proceedings 11*, pages 313–323. Springer, 2012.

[132] Siqi Ren, Bo Liao, Wen Zhu, Zeng Li, Wei Liu, and Keqin Li. The gradual resampling ensemble for mining imbalanced data streams with concept drift. *Neurocomputing*, 286:150–166, 2018.

[133] Siqi Ren, Wen Zhu, Bo Liao, Zeng Li, Peng Wang, Keqin Li, Min Chen, and Zejun Li. Selection-based resampling ensemble algorithm for nonstationary imbalanced stream data learning. *Knowledge-Based Systems*, 163:705–722, 2019.

[134] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " Why should i trust you?" Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[135] Stephanie K Riegg. Causal inference and omitted variable bias in financial aid research: Assessing solutions. *The Review of Higher Education*, 31(3):329–354, 2008.

[136] Gordon J Ross, Niall M Adams, Dimitris K Tasoulis, and David J Hand. Exponentially weighted moving average charts for detecting concept drift. *Pattern recognition letters*, 33(2):191–198, 2012.

[137] Leszek Rutkowski, Maciej Jaworski, Piotr Duda, Leszek Rutkowski, Maciej Jaworski, and Piotr Duda. Basic concepts of data stream mining. *Stream data mining: algorithms and their probabilistic properties*, pages 13–33, 2020.

[138] Patrik Sabol, Peter Sinčák, Pitoyo Hartono, Pavel Kočan, Zuzana Benetinová, Alžbeta Blichárová, L'udmila Verbóová, Erika Štammová, Antónia Sabolová-Fabianová, and Anna Jašková. Explainable classifier for improving the accountability in decision-making for colorectal cancer diagnosis from histopathological images. *Journal of biomedical informatics*, 109:103523, 2020.

[139] Farnaz Sadeghi and Herna Viktor. MQ-OFL: Multi-sensitive Queue-based Online Fair Learning. In *25th International Conference on Discovery Science (DS 2022)*, volume 13601, pages 271–285. Montpellier, France, October 10-12, Lecture Notes in Computer Science (LCNS), 2022.

[140] Farnaz Sadeghi and Herna L Viktor. Online-MC-queue: Learning from imbalanced multi-class streams. In *Third International Workshop on Learning with Imbalanced Domains: Theory and Applications (LIDTA 2021)*, volume 154, pages 21–34. Virtual Event, Proceedings of Machine Learning Research Journal (PMLR), 2021.

[141] Farnaz Sadeghi, Herna L Viktor, and Parsa Vafaie. DynaQ: online learning from imbalanced multi-class streams through dynamic sampling. *Applied Intelligence, Springer Nature*, pages 1–23, 2023. https://doi.org/10.1007/s10489-023-04886-w.

[142] Waddah Saeed and Christian Omlin. Explainable AI (XAI): A systematic meta-survey of current challenges and future opportunities. *Knowledge-Based Systems*, 263:110273, 2023.

[143] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[144] Jeffrey C Schlimmer and Richard H Granger. Incremental learning from noisy data. *Machine learning*, 1:317–354, 1986.

[145] Chris Seiffert, Taghi M Khoshgoftaar, Jason Van Hulse, and Amri Napolitano. RUSBoost: A hybrid approach to alleviating class imbalance. *IEEE transactions on systems, man, and cybernetics-part A: systems and humans*, 40(1):185–197, 2009.

[146] Shiven Sharma, Colin Bellinger, Bartosz Krawczyk, Osmar Zaiane, and Nathalie Japkowicz. Synthetic Oversampling with the Majority Class: A New Perspective on Handling Extreme Imbalance. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 447–456, 2018.

[147] Harini Suresh and John Guttag. A framework for understanding sources of harm throughout the machine learning life cycle. In *Equity and access in algorithms, mechanisms, and optimization*, pages 1–9. 2021.

[148] Andrea Torcianti and Stephan Matzka. Explainable artificial intelligence for predictive maintenance applications using a local surrogate model. In *2021 4th International Conference on Artificial Intelligence for Industries (AI4I)*, pages 86–88. IEEE, 2021.

[149] Alexey Tsymbal. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin*, 106(2):58, 2004.

[150] Parsa Vafaie, Herna Viktor, and Wojtek Michalowski. Multi-class imbalanced semi-supervised learning from streams through online ensembles. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pages 867–874. IEEE, 2020.

[151] Jasper van der Waa, Elisabeth Nieuwburg, Anita Cremers, and Mark Neerincx. Evaluating XAI: A comparison of rule-based and example-based explanations. *Artificial Intelligence*, 291:103404, 2021.

[152] Alexander Vergara, Shankar Vembu, Tuba Ayhan, Margaret A Ryan, Margie L Homer, and Ramón Huerta. Chemical gas sensor drift compensation using classifier ensembles. *Sensors and Actuators B: Chemical*, 166:320–329, 2012.

[153] Sahil Verma and Julia Rubin. Fairness definitions explained. In *Proceedings of the international workshop on software fairness*, pages 1–7, 2018.

[154] Herna L Viktor and Hongyu Guo. Multiple classifier prediction improvements against imbalanced datasets through added synthetic examples. In *Structural, Syntactic, and Statistical Pattern Recognition: Joint IAPR International Workshops, SSPR 2004 and SPR 2004, Lisbon, Portugal, August 18-20, 2004. Proceedings*, pages 974–982. Springer, 2004.

[155] Shuo Wang, Leandro L Minku, Davide Ghezzi, Daniele Caltabiano, Peter Tino, and Xin Yao. Concept drift detection for online class imbalance learning. In *The 2013*

International Joint Conference on Neural Networks (IJCNN), pages 1–10. IEEE, 2013.

[156] Shuo Wang, Leandro L Minku, and Xin Yao. Resampling-based ensemble methods for online class imbalance learning. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1356–1368, 2014.

[157] Shuo Wang, Leandro L Minku, and Xin Yao. Dealing with Multiple Classes in Online Class Imbalance Learning. In *IJCAI*, pages 2118–2124, 2016.

[158] Shuo Wang, Leandro L Minku, and Xin Yao. A systematic study of online class imbalance learning with concept drift. *IEEE transactions on neural networks and learning systems*, 29(10):4802–4821, 2018.

[159] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.

[160] Dongliang Xu, Zhihong Tian, Rufeng Lai, Xiangtao Kong, Zhiyuan Tan, and Wei Shi. Deep learning based emotion analysis of microblog texts. *Information Fusion*, 64:1–11, 2020.

[161] Zhang Yan, Du Hongle, Ke Gang, Zhang Lin, and Yeh-Cheng Chen. Dynamic weighted selective ensemble learning algorithm for imbalanced data streams. *The Journal of Supercomputing*, pages 1–26, 2022.

[162] I-Cheng Yeh and Che-hui Lien. The comparisons of data mining techniques for the predictive accuracy of probability of default of credit card clients. *Expert systems with applications*, 36(2):2473–2480, 2009.

[163] Chongsheng Zhang, Jingjun Bi, Shixin Xu, Enislay Ramentol, Gaojuan Fan, Baojun Qiao, and Hamido Fujita. Multi-imbalance: An open-source software for multi-class imbalance learning. *Knowledge-Based Systems*, 174:137–143, 2019.

[164] Hang Zhang, Weike Liu, and Qingbao Liu. Reinforcement online active learning ensemble for drifting imbalanced data streams. *IEEE Transactions on Knowledge and Data Engineering*, 34(8):3971–3983, 2020.

[165] Wenbin Zhang and Albert Bifet. FEAT: A Fairness-Enhancing and concept-Adapting decision Tree classifier. In *Discovery Science: 23rd International Conference, DS 2020, Thessaloniki, Greece, October 19–21, 2020, Proceedings 23*, pages 175–189. Springer, 2020.

[166] Wenbin Zhang and Eirini Ntoutsi. FAHT: An Adaptive Fairness-aware Decision Tree Classifie. *arXiv preprint arXiv:1907.07237*, 2019.

[167] Wenbin Zhang, Mingli Zhang, Ji Zhang, Zhen Liu, Zhiyuan Chen, Jianwu Wang, Edward Raff, and Enza Messina. Flexible and adaptive fairness-aware learning in non-stationary data streams. In *2020 IEEE 32nd International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 399–406. IEEE, 2020.

[168] Jianlong Zhou, Fang Chen, and Andreas Holzinger. Towards explainability for AI fairness. In *International Workshop on Extending Explainable AI Beyond Deep Models and Classifiers*, pages 375–386. Springer, 2020.

[169] Indrė Žliobaitė, Albert Bifet, Jesse Read, Bernhard Pfahringer, and Geoff Holmes. Evaluation methods and decision theory for classification of streaming data with temporal dependence. *Machine Learning*, 98:455–482, 2015.

[170] Paweł Zyblewski, Paweł Ksieniewicz, and Michał Woźniak. Classifier selection for highly imbalanced data streams with minority driven ensemble. In *Artificial Intelligence and Soft Computing: 18th International Conference, ICAISC 2019, Zakopane, Poland, June 16–20, 2019, Proceedings, Part I 18*, pages 626–635. Springer, 2019.

[171] Paweł Zyblewski, Robert Sabourin, and Michał Woźniak. Preprocessed dynamic classifier ensemble selection for highly imbalanced drifted data streams. *Information Fusion*, 66:138–154, 2021.

[172] Paweł Zyblewski and Michał Woźniak. Dynamic ensemble selection for imbalanced data stream classification with limited label access. In *Artificial Intelligence and Soft Computing: 20th International Conference, ICAISC 2021, Virtual Event, June 21–23, 2021, Proceedings, Part II 20*, pages 217–226. Springer, 2021.