Density Based Query By Committee - Robust Active Learning Approach for Data Streams

S.M.A Prabashwara , O. Seneweera, G. Dharmarathne
Department of Statistics, University of Colombo, Sri Lanka

***Abstract*—Acquiring precise labels for continuously flowing data streams is resource-intensive and costly. Active learning offers a potential strategy for training precise models while minimizing label requirements with minimal annotation effort. However, adapting active learning to streaming data becomes intricate due to the ever-changing data distributions, known as concept drift. Existing approaches for active learning in data streams predominantly rely on uncertainty sampling and Query by Committee (QBC) due to their simplicity and ease of implementation. This paper introduces a novel and a robust active learning approach tailored for data streams merging key elements from QBC and density-weighted sampling to effectively address the challenges posed by concept drift. Through a comprehensive analysis using benchmark datasets widely used in the literature related to data streams, we demonstrate the superior performance of our proposed method across various data stream scenarios. This includes instances with no concept drift, instances with the presence of concept drift, as well as scenarios involving severe concept drift. In addition, the results reveal that strategies based on uncertainty sampling and its variants exhibit limitations in the presence of concept drift, whereas QBC and its variants prove to be inadequate when faced with significant concept drift. In contrast, our approach, which combines the strengths of QBC and density-weighted sampling using Gower's distance as a similarity measure, exhibits remarkable adaptability to evolving data distributions.**

***Index Terms*—Density-Based Sampling; Query By Committee; Active Learning; Evolving Data**

I. Introduction

Recent technological advancements and the proliferation of automated data collection methodologies have led to an unprecedented data deluge. However, a considerable portion of this data remains unlabeled, creating significant challenges due to the high cost of transforming it into labeled forms, which requires a process necessitating human expertise or oracles. Supervised learning, a popular branch of machine learning, relies heavily on accurate labels, rendering this methodology is potentially ineffective and costly if the expense of labeling exceeds the allocated budget.

Active learning, often referred to as query learning, offers a promising solution to address the challenges associated with high labeling costs. Its foundational hypothesis postulates that if a machine learning algorithm were allowed to select the data from which it learns, it could enhance its performance with less required training. The primary goal of active learning is to reduce the necessary training sample size while maintaining high prediction accuracy, thereby improving data efficiency. It is already being progressively employed within software companies and large-scale research projects, including those at Cite Seer, Google, IBM, and Microsoft. Active learning aims to enhance the efficiency of labeled data by selectively querying the most informative examples. In situations where data streams at high velocities and needs to be processed in real-time or near-real-time environments, labelling every incoming data point becomes impractical due to time constraints, even with a substantial labeling budget.

In high-velocity scenarios where real-time labeling is required, active learning shines. Instead of randomly selecting data points from the stream for labeling, which may or may not be informative, active learning strategically chooses the most informative instances. By doing so, it maximizes the utility of each labeled example, leading to faster and more accurate model training.

Moreover, in high-speed data streaming contexts, active learning provides a more data-efficient approach by intelligently selecting which data points to label, as opposed to relying on random selection. Data streams represent an infinite and fast-flowing source of data. Unlike batch data, data streams are continuous, requiring real-time or near-real-time analytics due to storage constraints and the necessity for timely decision-making. Examples of such streaming data can be found in daily transactions, sensor readings, web data, social media feeds, Internet of Things (IoT) devices, and online monitoring systems.

Addressing the challenges posed by streaming data requires a different approach than that is used with batch data. One of the key challenges is concept drift, which involves the evolution or change in data distribution over time. Additionally, unlike batch data, which allows for multiple data passes, streaming data necessitates one-pass processing (preprocessing of the data must be done at a single step). The high velocity of data streams and the need for significant computational power to process data in real-time make these challenges more complex. Using static, batch-data algorithms in this context often leads to models quickly becoming outdated due to concept drift. Moreover, considering that access is only granted to the data in motion, batch learning algorithms are inadequate for building predictive models for streaming data. This paper aims to explore these issues and potential solutions

Correspondence: S.M.A Prabashwara (E-mail: smashoka123@gmail.com)
Received: 16-06-2024 **Revised:** 12-08-2024 **Accepted:** 09-09-2024
S.M.A Prabashwara, O. Seneweera and G. Dharmarathne are from University of Colombo (smashoka123@gmail.com, oshada.senaweera@gmail.com, sameera@stat.cmb.ac.lk)
DOI: https://doi.org/10.4038/icter.v18i2.7293
The 2025 Special Issue contains the full papers of the abstracts published at the 24th ICTer International Conference.

in depth, providing an active learning query strategy that can be integrated with machine learning models for streaming data.

II. Related work

Active learning concepts are widely applied in the supervised learning setting; in addition, these concepts are employed in unsupervised learning and function optimization [1]. Wenbin (2013) [2] states that active learning has been extensively studied for classification problems, and only minimal work is done on active learning for regression compared to classification. For classification and regression problems, several widely used algorithms are stated by [3]. In addition, the study [4] provides glimpses of integrating active learning along with deep learning models. All these active learning strategies are categorized into three categories, namely, heterogeneity-based, performance-based, and representativeness-based, depending on their underlying theories [5].

Query-By Committee (QBC) algorithm [6] is found to be the first active learning query strategy as per the literature. The QBC strategy generates a committee of models and selects the most unlabelled examples about which committee members disagree. QBC has been used with probabilistic models, specifically with the naïve Bayes model for text classification [7]. In early studies it has been claimed that this algorithm does not perform well in text classification when integrated with methods such as support vector machines. However, the results obtained from the studies by Settles and Craven in 2008 [8] contradict the previous findings and show that even committees of small sizes tend to work well in practice for most real-world applications [6] [3] [9].

Uncertainty sampling is another active learning query strategy introduced in the paper [10]. It is one of the most widely used approaches where the learner will query the actual label of the instance where the classifier is most uncertain. This active learning strategy is extensively utilized since it is simple to apply with probabilistic models [2].Besides the above active learning query strategies in the literature, two other widely used active learning strategies in classification are defined as expected model change and density-based methods. Moreover, it is identified that the expected model change approach functions well in reality but might be computationally expensive if the feature space is substantial. However, this strategy is not limited to classification; it can also be used in the context of regression [3]. Density-based methods are another active learning strategy suggested by [3]. This strategy takes the representativeness and informativeness of observation when querying which observations are to be labeled. A variety of similar active learning query strategies existed before developing this strategy. For example, a density-based QBC approach for text classification has been carried out by McCallum and Nigam [7], and Nguyen and Smeulders (2004) [11] proposed a strategy that uses nearest neighbors as a similarity measure to evaluate the representativeness of the instances.

Much of the active learning strategies in literature is focused on batch data setting. However, there are several studies that focus on introducing active learning strategies for streaming data. A comprehensive study has been carried out by [12] on the active learning classification of streaming data. The study states that the accuracy does not strongly depend on the number of labelled objects, and 20% is enough to achieve a very similar accuracy as the full supervised approach. Moreover, it is reported that the machine learning model's performance affects the performance of the active learning query strategy. The KNN and Naïve Bayes algorithms performed considerably well compared to that of the rule classifier and the perceptron used. The study further expresses that setting a higher threshold for querying will negatively impact the classifier's stability as the model's standard deviation increases. An even more comprehensive study compared to the above study has been carried out by [13]. The study is completely based on uncertainty sampling query strategy for streaming data where the performance of the query strategy is evaluated using four different variants, which include random strategy, fixed uncertainty, variable uncertainty, and randomized variable uncertainty. In this case, the random strategy acts as the baseline model for comparing the performance of active learning. Fixed uncertainty query instances depend on a fixed threshold throughout the stream, while the other two techniques will have a variable threshold that changes according to the incoming stream. In addition, the randomized uncertainty strategy will query instances randomly from the stream. The results suggest that the variable uncertainty performs well in many cases but only in situations where drift is not strongly expressed. If a more significant drift is expressed, then randomized uncertainty prevails over the other techniques.

In the literature on AL for data streams, there are only a handful of studies that focus on AL approaches when the data stream is evolving. A study by Lughofer (2012) [14] proposes an AL algorithm based on concepts of conflict and ignorance. Conflict models the extent to which incoming data point lies in conflict regions of two or more classes. This reflects the level of certainty of the classifier about the prediction of that data point. Ignorance models the an incoming data point's distance from the training samples seen so far. This reflects the actual variability of the version space. This approach was initially defined for classification tasks. However, later on, Lughofer and Pratama (2018) [15] extended this approach to evolving regressing models also. Another AL approach for evolving data streams is proposed in [16]. This approach uses a dynamic threshold based on the variable uncertainty strategy introduced in [13] with a recurrent fuzzy classifier. However, there is a gap in the literature regarding the integration of ignorance and conflict models [15] and density-based sampling [3] with a variable threshold strategy. While approaches utilizing variable thresholds exist, they have not been combined with the strengths of ignorance and conflict models for active learning on evolving data streams. This presents an opportunity for further research to explore the potential benefits of such a combined approach. In addition to these AL approaches which focus on evolving streams, a technique called stream-based active learning (SAL) has been proposed by [17], which can handle both the drift and evolution of the stream. SAL aims to optimize the expected future error, using non-parametric Bayesian Modelling.

Studies [18] [19] [20] explores the current state of active learning (AL) approaches for data streams. The study emphasizes that, while numerous active learning query strategies have been developed to identify the most informative data points for models, several challenges arise when applying these techniques in real-world scenarios. These challenges include algorithm scalability, data drift, and model interpretability.

As per the above study, the primary challenge is algorithm scalability, where the algorithms need to be both efficient and scalable to manage large datasets. As the data volume increases, the computational demands of these algorithms can grow exponentially. Furthermore, these techniques must provide clear interpretations of why specific data points are deemed informative, ensuring transparency and validating the algorithm's decisions. In response to these challenges, this study aims to develop an active learning query strategy that efficiently handles data streams while maintaining interpretability.

In conclusion, the literature highlights the need for robust AL approaches that can handle dynamic and evolving data streams. Furthermore, literature on batch data sets indicates that by blending existing AL strategies, more robust AL strategies can be invented. Therefore, it is valuable to extend these insights to streaming data settings, with the goal of creating robust AL approaches for data streams that prioritize simplicity and interpretability, while also ensuring computational efficiency.

III. STRATEGIES

A noteworthy finding from the literature is the effectiveness of density-based sampling (also referred to as Density-based sampling), particularly in the context of batch data settings. Although the term "density-based" implies a formal density function, these strategies, as noted by [3], leverage the broader concept of data distribution to identify informative data points for labeling. Imagine your data as points scattered in a space. Dense regions represent areas with many data points close together, while sparse regions have fewer points spread out. density-based active learning focuses on selecting points near the boundaries between these regions. These points are likely more uncertain for the model due to a lack of similar training examples. By prioritizing these "boundary points" for labeling, the model can improve its understanding of the decision boundaries between different classes, ultimately enhancing its performance.

This result is validated by Settles [3] as it was identified that density-based sampling approaches are superior to methods that do not consider density measure to measure the representativeness of the data point. Elaborating further, the data point A in Figure 1 is the most uncertain and influential observation for the model. However, data point A is not representative of other instances considered. So, querying the actual label of A is unlikely to improve the accuracy of the overall model. These issues can be addressed by explicitly incorporating the representativeness aspect when querying the incoming instances.

Building on this insight, we have developed two innovative active learning query strategies that integrate the principles of density-based methods with the Query-By-Committee (QBC) and uncertainty (refer to Appendix 3). Through the analysis, it was observed that density-based QBC outperforms density-based uncertainty in all three scenarios considered (refer to Appendix 4). Consequently, this paper exclusively focuses on the density-based QBC strategy. This approach capitalizes on the strengths of density-based sampling in conjunction with the QBC sampling technique.

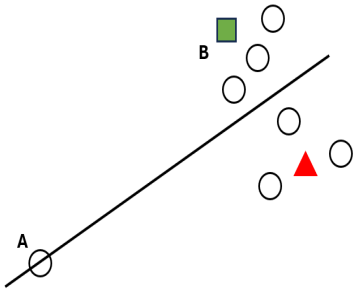

Fig. 1: Figure indicating that most uncertain data is not the most important [3]

Following the QBC algorithm, the most informative instances from the stream are determined using vote entropy. The selection criterion is to query the actual label of instances, which maximizes the following statistic.

$$x^*_{VE} = \arg\max_x \left[-\sum_i \frac{V(y_i)}{C} \log\left(\frac{V(y_i)}{C}\right) \right] \quad (1)$$

Where:

C represents the size of the committee

$V(y_i)$ is the count for each label obtained from the committee

The equation 3.1 gives vote entropy which can be used as a measure of informativeness. Further, we require a representative measure. The reason for including a representative measure alongside the informative measure is to account for the possibility that the most influential and outlying observations may be identified as the data points to be queried (the model is most uncertain). However, these points are unlikely to improve the overall model accuracy significantly. Using a representative measure, we explicitly incorporate the input distribution when querying the accurate labels to address this issue.

In the case of batch data, where the entire unlabeled set is available, it is possible to identify data that represents densely populated areas. However, this becomes impossible with streaming data, as the unlabeled set cannot be utilized due to its constant state of flux. Hence, it is proposed to use the labeled set to assess the representativeness of the incoming instances. Moreover, storing all labeled instances may not be feasible because data streams can be infinitely long. Instead, [21] suggests a moving window approach, which involves a user-defined window size that includes the most recent labeled

instances. These labeled data can then be used to compute the similarity measure.

Gower's distance has been utilized for calculating the distance-based similarity measure when assessing representativeness. If Euclidean distance, Manhattan distance, or Jaccard distance is employed as the similarity measure, our developed active learning query strategy would be constrained by the data type; it could only be applied with either numerical data or categorical data (as in the case of Jaccard Distance), but not both. To circumvent this limitation, the decision has been made to utilize Gower's distance, allowing the technique to be generalized to any streaming dataset regardless of it being categorical or quantitative. This choice empowers our model to handle various data types, including logical, numerical, categorical, or even text data. This flexibility makes Gower's distance a powerful and versatile tool in the computation of the representativeness measure in our density-based active learning query strategy [22]. Further, the utilization of Gower's distance as the similarity measure enables the application of the novel active learning strategy to any dataset.

The suggested expression to calculate the representativeness of each instance is as follows:

$$d = \underset{x}{\operatorname{argmin}} \left(\frac{1}{L} \sum_{n=1}^{L} \operatorname{dist}(x, x_n) \right)^{\beta} \tag{2}$$

L - Size of the labeled dataset including recent data

β - Weight parameter

Our goal is to select instances that provide valuable information to the model and to represent the overall dataset. Combining equations 3.1 and 3.2, equation 3.3 can be obtained. Equation 3.3 is formulated in a way such that it will try to pinpoint instances where the model exhibits the least confidence (highest vote entropy or highest information) and simultaneously try to select instances most representative of the entire data set (lowest distance measure). Therefore this strategy generates a statistic that jointly consider both informativeness and representativeness where a higher value for the statistic imply high value of that particular label to the model.

$$x_{DWQBC} = \arg\max_{x} \left[-\sum_{i} \frac{V(y_i)}{C} \log \left(\frac{V(y_i)}{C} \right) \right] - \left(\arg\min_{x} \left(\frac{1}{L} \sum_{n=1}^{L} \operatorname{dist}(x, x_n) \right)^{\beta} \right) \tag{3}$$

where $\beta > 0$, using the result in Appendix A, The derived expression for density-based query by committee active learning strategy is as follows:

$$x_{DWQBC} = \arg\max_{x} \left[-\sum_{i} \frac{V(y_i)}{C} \log \left(\frac{V(y_i)}{C} \right) \right] - \left(\frac{1}{L} \sum_{n=1}^{L} \operatorname{dist}(x, x^n) \right)^{\beta} \tag{4}$$

If the calculated x_{DWQBC} is greater than the threshold, the actual label for the corresponding instance will be queried; otherwise, it will be discarded. If we look at the equation 3.4 closely, we can see that the primary concept behind the developed query strategy involves modifying the QBC algorithm by introducing a penalty for the loss function, precisely, the vote entropy, based on a centrality measure. This modification aims to discourage the selection of data points distant from most of the dataset under the assumption that such points may be outliers. The rationale is that querying the actual label of an outlier is unlikely to enhance the model's accuracy significantly.

While outliers might provide valuable information, in our approach, a parameter, denoted as β, is employed to balance the weight between the representative measure and the vote entropy (informativeness). This strategic use of β ensures that we can balance between informativeness and representativeness of data points depending on the requirements (β is a tune-able parameter).

The model's training is done similarly to the existing active learning query strategies, where the classifier is initialized with a small subset of the data. Then, the actual label will be queried depending on the decision rule. Algorithm 1 depicts the most basic form of the developed query strategy, the where we query the data if the calculated x_{DWQBC} value is greater than a certain threshold.

Algorithm 1 Density-Based Query-By-Committee for Streaming Data - Fixed Strategy

Require: X_t – Incoming instance, θ - Threshold, W - Window Size, β - Density parameter, C - Committee

Ensure: Whether to request the actual label or not {true, false}

1: Initialize classifier L
2: **for** each X_t in incoming instance **do**
3: Calculate

$$T_t = \arg\max_{x} \left(\sum_{i} \left(\frac{V(y_i)}{C} \right) \log \left(\frac{V(y_i)}{C} \right) - \left(\frac{1}{L} \sum_{n=1}^{L} \operatorname{dist}(x, x^L) \right)^{\beta} \right)$$

4: **if** $T_t > \theta$ **then**
5: Request the true label y_t of instance X_t
6: Train classifier L with (X_t, y_t)
7: **end if**
8: **end for**

A. Variable Threshold Strategy

Distributing the labeling workload over time is one of the most complex aspects of using active learning query strategies with streaming data. When a fixed threshold is employed, the classifier may exhaust the labeling budget or reach the threshold certainty after some time. In the case of a fixed threshold, it prematurely terminates the learning process and cannot adapt to future changes.

The concept behind the variable threshold strategy is to use a decision criterion with a value that can vary across the data stream. To achieve this, it is suggested to label the cases that exhibit the lowest confidence within a specific time frame. As a result, the threshold is adjusted to fluctuate based on the incoming data from the stream, allowing for efficient budget utilization. If a classifier becomes more stable, the we will have a larger threshold that will not query even the most uncertain cases. Conversely, in the event of changes in data distribution (concept drift), which may lead to an increased need for labeling, the threshold is narrowed to selectively query the most informative cases instead of labeling all cases.

This variable threshold strategy is designed so that more labels are requested when the learner lacks confidence, and fewer labels are needed when conditions are stable. The dynamic threshold ensures a consistent number of labels are requested, providing control over the labeling budget throughout the data stream. The underlying concept of the variable threshold strategy is presented in Algorithm 2.

Algorithm 2 Density-Based Query-By-Committee for Streaming Data - Variable Threshold Strategy

Require: X_t – Incoming instances, B – Labeling budget, s – Adjusting step, P – Labeling cost per observation

Ensure: Whether to request the actual label or not {true, false}

Initialize total labeling cost $C = 0$
if $C < B$ **then**
  Calculate

$$T_t = \arg\max_x \left(\sum_i \left(\frac{V(y_i)}{C} \right) \log \left(\frac{V(y_i)}{C} \right) - \left(\frac{1}{L} \sum_{n=1}^{L} \text{dist}(x, x^L) \right)^{\beta} \right)$$

  if $T < \theta$ **then**
    Increase labeling cost: $C = C + P$
    Narrow the threshold: $\theta = \theta(1 - s)$
  else
    Widen the threshold: $\theta = \theta(1 + s)$
  end if
else
  return false
end if

B. Randomized Variable Threshold Strategy

In some situations, concept drift may occur in input data as well as in labels. If concept drift occurs in labels, the classifier cannot notice it without labels. So, to capture such changes in the data, cases about which the classifier is very confident are queried from time to time. This is done by multiplying the labeling threshold by a random variable with a normal distribution following $\mathcal{N}(1, \delta)$ and computing a new threshold. Remaining implementation is similar to that of variable threshold strategy. Algorithm 3 illustrates how the randomized variable threshold variant is implemented for the newly developed query strategy.

Algorithm 3 Density-Based Query-By-Committee for Streaming Data - Randomized Variable Threshold Strategy

Require: X_t – Incoming instances, B – Labeling budget, s – Adjusting step, P – Labeling cost per observation, δ – Randomization Threshold

Ensure: Whether to request the actual label or not {true, false}

Initialize total labeling cost $C = 0$
if $C < B$ **then**
  Calculate

$$T_t = \arg\max_x \left(\sum_i \left(\frac{V(y_i)}{C} \right) \log \left(\frac{V(y_i)}{C} \right) - \left(\frac{1}{L} \sum_{n=1}^{L} \text{dist}(x, x^L) \right)^{\beta} \right)$$

  Compute randomized threshold: $\theta_{\text{randomized}} = \theta \times \delta$
  if $T < \theta_{\text{randomized}}$ **then**
    Increase labeling cost: $C = C + P$
    Narrow the threshold: $\theta = \theta(1 - s)$
  else
    Widen the threshold: $\theta = \theta(1 + s)$
  end if
else
  return false
end if

IV. Analysis

The datasets used in this analysis were obtained from an online repository containing various benchmark streaming datasets. All of these datasets have been previously utilized in the literature on streaming settings. The available datasets include binary classification, multiclass classification, datasets with concept drift, datasets without concept drift, and imbalanced datasets.

A. Description of the datasets

The datasets used in this analysis are not novel; instead, they have been extensively employed in previous studies on streaming machine learning. Their prior usage in research provides a rich background of information, enhancing the validity of our findings and facilitating more relevant interpretations. Consequently, our study not only contributes to the existing knowledge associated with each dataset but also advances the broader field of streaming machine learning. The drift status of the datasets was determined based on the results presented in Appendix B.

This study investigated two main approaches for detecting drift in data. The first set of techniques focused solely on the distribution of the input data (x). These included methods like the Hoeffding Drift Detection Method Average (HDDA) and Moving Average (HDDM) [23]. However, these methods

yielded significantly different results compared to the Adaptive Window Technique (ADWIN) and the Drift Detection Method (DDM). This discrepancy suggests that relying solely on the input distribution might only detect superficial changes in the data, is referred to as "virtual drift." This could lead to inaccurate conclusions about the underlying trends. To overcome this limitation and identify true changes in the data ("real drift"), techniques that analyze posterior probabilities were necessary. Therefore, the well-established ADWIN algorithm [24] was employed for accurate drift detection.

TABLE I: Dataset Information

Dataset	Instances	Attributes	Labels	Drift Status
Sea stream ([25])	40,000	4 (1+3)	2	No-Drift
Timestamp Iris ([26])	300	5 (1+4)	3	No-Drift
Electricity ([27])	45,312	8 (1+7)	2	Drift
Airlines ([25])	539,383	7 (5+2)	2	Drift
Cover Type ([28])	581,012	54 (44+10)	7	Serious Drift
Hyperplane Fast ([25])	10000	11 (1+10)	2	Serious Drift

B. Experimental Design

1) Description: The primary objective of the experiment is to compare the performance of newly proposed active learning query strategies with existing strategies. The experimental setup involves evaluating the performance of the active learning query strategy along with its variants on the datasets stated in Table 1.

The performance of each strategy will be evaluated using two machine learning techniques for streaming data: the Hoeffding tree and the Naive Bayes algorithm. These choices are supported by literature, which highlights the efficiency of the Hoeffding tree classifier in stream data classification due to its time-saving nature and superior attribute comparison capabilities [29]. Similarly, the Naive Bayes algorithm is favored for its simplicity, low computational cost, and incremental learning capabilities, despite assuming attribute independence [13]. This study primarily focuses on utilizing these algorithms during the experimental setup.

2) Active Learning Performance Strategy: The evaluation covers a wide range of scenarios, including comparisons with a baseline model (random sampling) and a model that incorporates all labeled data (Conventional Streaming Machine Learning model). When assessing the effectiveness of an active learning query strategy against the random model, the study does not merely compare overall accuracy. It emphasizes that for an active learning model to be considered effective, its performance must consistently outperform the random model throughout the data stream. This is important due to the additional computational cost associated with active learning strategies compared to randomly sampling instances from the stream. Therefore, active learning model must consistently outperform random sampling, aligning with the ultimate goal of developing a more accurate model while controlling the labeling budget. As per the evaluation technique, prequential evaluation [30] was the chosen method for performance assessment.

3) Parameter Settings: The performance of these active learning query strategies will be evaluated using default parameters $s = 0.01$ and $\delta = 1$, as [13] utilized in their uncertainty sampling study. Furthermore, [12] suggested that labeling 20% of the total stream is sufficient to achieve an accuracy similar to the fully supervised approach. This insight guided the setting of our labeling budget to 20% of the total stream for all active learning query strategies, where such control was feasible.

In addition, parameter β was set using a parameter grid, which ranges from [0, 2]. For fixed threshold strategy, θ was set using a parameter tuning and the best performing θ value was selected when training the final model. For randomized and variable randomized θ was initialized with 1 and the threshold is set in a way that it will change depending on the sampling rate.

4) Additional Information: To evaluate vote entropy in QBC, a committee of models is necessary. Settles (2008) [8] suggests that even a committee of three models can effectively perform in batch data scenarios. Thus, a committee size of three was employed in the experimental study. Additionally, ensemble methods tend to perform well in committees [31]. The committee consisted of the Adaptive Random Forest, Ensemble Bagging Classifier, and Ensemble Leverage Bagging Classifier. Furthermore, the models within the committee were updated simultaneously with the true labels queried from the stream. The analysis was conducted using open-source Python libraries, specifically designed for data streams in online environments, such as river ml and scikit-multiflow."

V. Results

The experimental results under the three circumstances, the absence of concept drift, the presence of concept drift, and the presence of severe concept drift, are as follows.

A. Absence of Concept Drift

When examining the sea stream and the time stamp iris datasets, it was determined that neither exhibited signs of concept drift. Figures 2-5 showcase the comparative accuracies of the density-based QBC (DWQBC) strategies and how they fare against two baseline models (random model and all data model). This indicates that the DWQBC variants consistently outperformed the random model and in some instances (Figures 2 and 4) even the model with fully labeled data. This reaffirms the results of previous studies where active learning models outperform models with all the data, and such observations are prevalent in batch data, where active learning eliminates the effects of over-fitting, as indicated by [31].

B. Presence of Concept Drift

In situations where concept drift is prevalent, the widely used active learning query strategies tend to underperform when paired with the Hoeffding tree model. This limitation becomes particularly evident when evaluating their performance on two benchmark datasets: the 'electricity' and the 'airlines' datasets as observed in Appendix D. Figures 6 and 7 depict

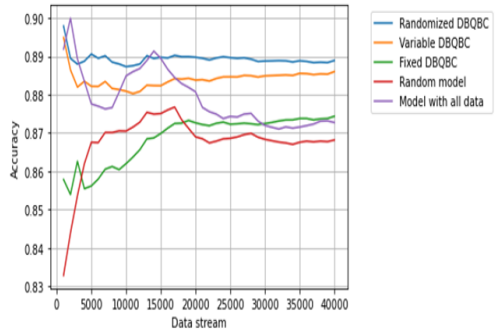

Fig. 2: Density weighted QBC for sea stream dataset - HT

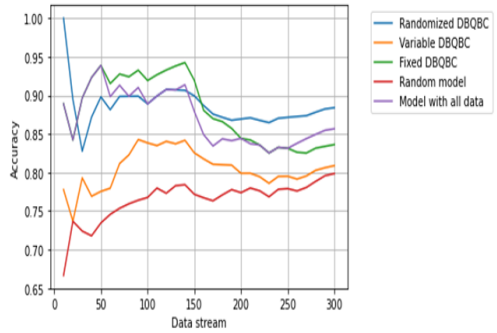

Fig. 3: Density weighted QBC for time stamp iris dataset - HT

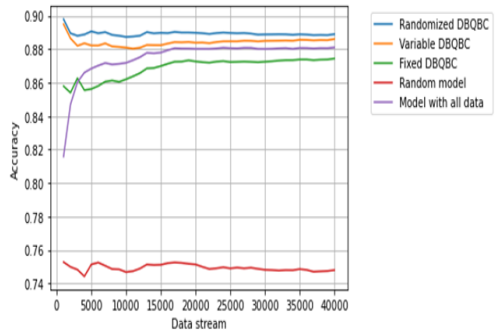

Fig. 4: Density weighted QBC for sea stream dataset - NB

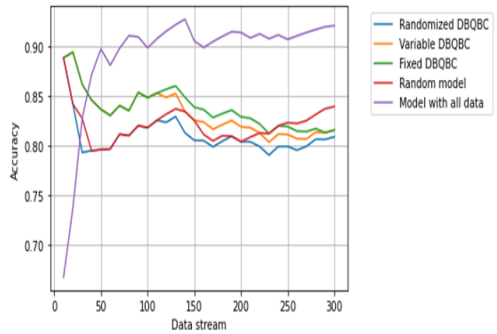

Fig. 5: Density weighted QBC for time stamp iris dataset - NB

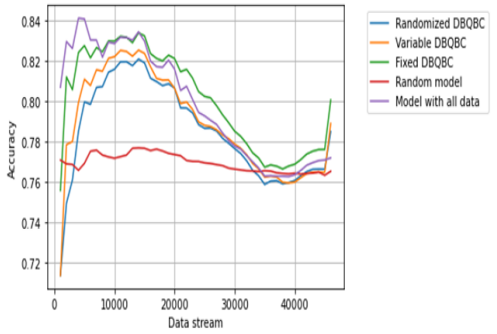

Fig. 6: Density weighted QBC for electricity dataset - HT

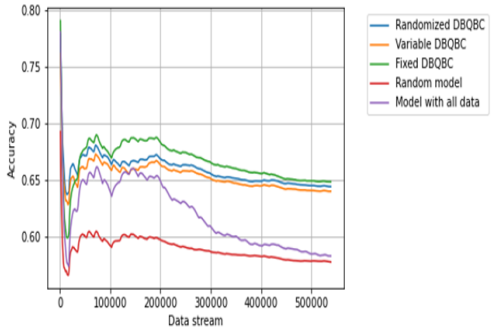

Fig. 7: Density weighted QBC for airlines dataset - HT

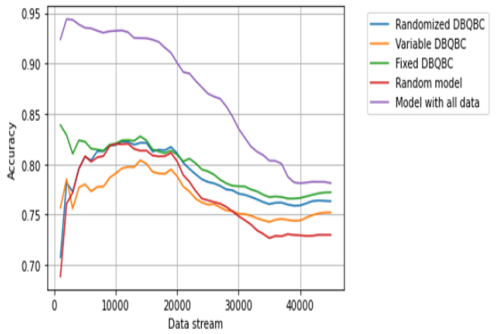

Fig. 8: Density weighted QBC for electricity dataset - NB

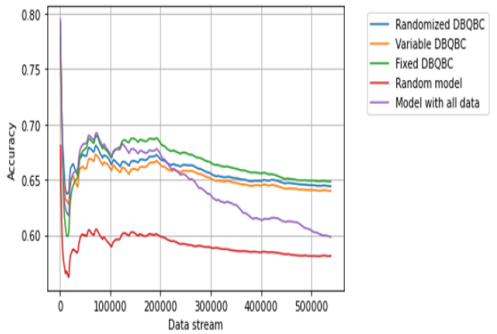

Fig. 9: Density weighted QBC for airlines dataset - NB

the performance of DWQBC sampling and its variants with the Hoeffding tree model on both the electricity dataset and the airlines dataset. These figures highlight the effectiveness of the DWQBC query strategy in comparison to random querying. The same observation is evident in Figures 8 and 9 when using the Naive Bayes classifier. The DWQBC query strategy consistently produces a better performance than the random model. Additionally, the variable threshold strategy and the randomized variable threshold strategy, show significant efficacy compared to the fixed threshold strategy. These techniques demonstrate commendable performance even when only 20% of the complete data stream is labeled.

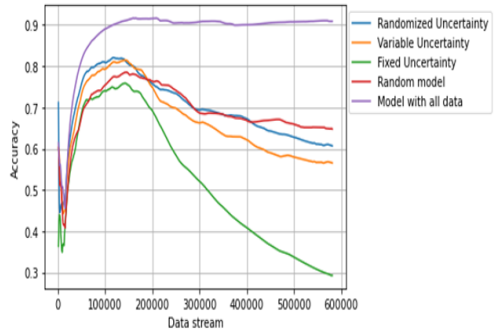

Fig. 10: Uncertainty sampling for cover type dataset - HT

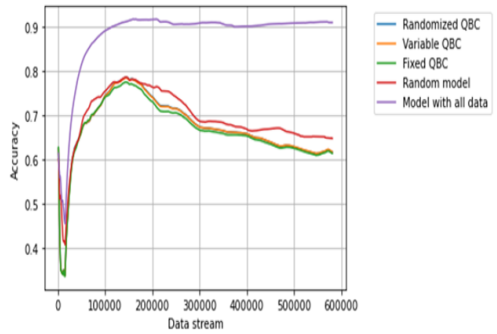

Fig. 11: QBC query strategy for cover type dataset - HT

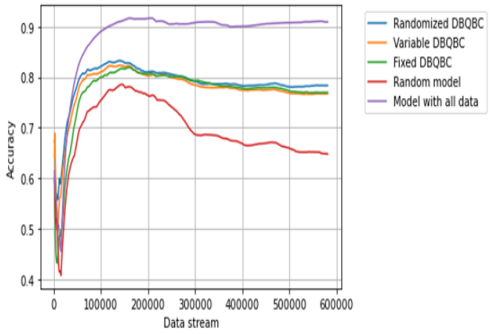

Fig. 12: Density-based QBC for cover type dataset - HT

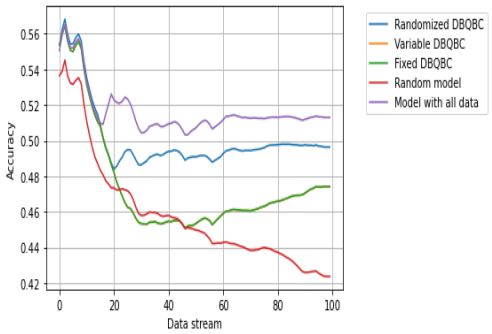

Fig. 13: Density Based QBC query strategy for Hyperplane dataset - HT

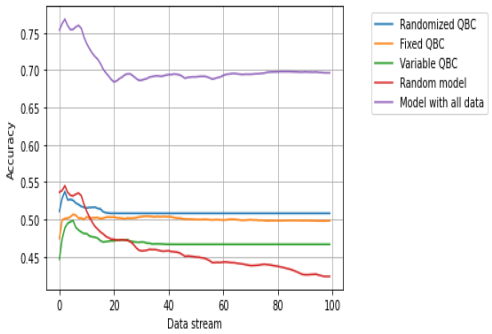

Fig. 14: Density-based QBC for Hyperplane type dataset - HT

C. Presence of Serious Concept Drift

Many classification tasks in practical, real-world scenarios deal with challenges such as multiclass categorization, class imbalance, and significant concept drifts. Recognizing these complexities, we aimed to evaluate the efficacy of the newly devised DWQBC query strategy, especially in a streaming setting that mirrors these challenges. For this context, we selected two datasets with severe concept drift (as shown in Appendix B): the hyperplane dataset and the well-established Forest Cover dataset.

In Figures 10 and 11, we can see the inadequacies of the two most commonly employed active learning query strategies for streaming data, Uncertainty Sampling and QBC (we implement these methods with different thresholds as in [13]). It is evident that these methods under-perform even when compared with the basic random strategy.

Figure 12, a pivotal outcome from the analysis, illustrates the better performance of the DWQBC variants in comparison to the Random model. This result is further supported by the results presented in Figures 13 and 14 which showcase the better performance of DWQBC variants over random model in the Hyperplane dataset. However, we noted that DWQBC does not perform as well as the model with all data in this scenario. This is an expected result as it would be impossible to capture all the patterns in a data stream that changes constantly with only limited instances.

VI. DISCUSSION

The learning curves in the results section show varying initial stages of learning. Some graphs start from the same point, while others do not. This variation arises from using a percentage of the entire stream for model initialization, with the initial 2% serving as the initial set. The number of entries used for training varies across datasets. On the other hand, accuracy for the plots is computed for batches of 5% of the entire training set. Due to this, we can expect differences in the initial stages of the plots. However, this observation has no impact on the long-term performance, which is our main concern.

In the analysis, it was observed that the two most widely used active learning query strategies (uncertainty sampling and QBC) for streaming data performed well only in the absence of concept drift. Notably, both of these strategies showed poor performance when confronted with severe concept drift. On the other hand, our newly proposed strategy DWQBC demonstrated effectiveness in the presence of both regular concept drift and severe concept drift as well as in the absence of concept drift. The tables below present the mean accuracy of the fitted models across different datasets using the Hoeffding

tree model.

The average accuracy of the active learning query strategy fitted with the Hoeffding tree model is summarized in Tables 2,3 and 4, classified according to the three scenarios considered in the study. Notably, the accuracy columns of the datasets in the presence of concept drift highlight the limitations of the widely used active learning query strategies in handling concept drift. The findings on the benchmark datasets suggest that the newly developed DWQBC query strategy consistently outperforms other query strategies with respect to average accuracy in all considered scenarios.

Furthermore, our results align with the findings of [13], that identified that uncertainty sampling with randomized and variable thresholds outperforms uncertainty sampling with fixed thresholds in the presence of concept drift. A similar observation is evident from our results as well, where randomized and variable threshold query strategies perform better than fixed threshold query strategies in the presence of concept drift.

TABLE II: Performance of Various Models on Sea and Time Stamp Iris Datasets for Hoeffding tree model (Absence of Concept Drift)

Model	Sea Dataset	Time Stamp Iris Dataset
Model with all data	0.87903	0.87074
Random model	0.86770	0.76459
Fixed uncertainty	0.80839	0.83944
Variable uncertainty	0.87126	0.85405
Randomized uncertainty	0.88182	0.85548
Fixed QBC	0.87496	0.89000
Variable QBC	0.88148	0.65658
Randomized QBC	0.88218	0.79356
Fixed DBQBC	0.88865	0.88017
Variable DBQBC	0.88410	0.89496
Randomized DBQBC	0.88930	0.88615

TABLE III: Performance of Various Models on Electricity, Airline, and Cover-Type Datasets for Hoeffding Tree Model (Presence of Concept Drift)

Model	Electricity Dataset	Airline Dataset
Model with all data	0.79848	0.64421
Random model	0.76931	0.59016
Fixed uncertainty	0.76582	0.60847
Variable uncertainty	0.77701	0.63943
Randomized uncertainty	0.77952	0.64960
Fixed QBC	0.77372	0.65532
Variable QBC	0.76831	0.65177
Randomized QBC	0.79106	0.65741
Fixed DBQBC	0.80550	0.65780
Variable DBQBC	0.78886	0.64563
Randomized DBQBC	0.78487	0.65564

Uncertainty sampling demonstrated good performance when compared to other strategies in the absence of concept drift. However, its effectiveness diminished in the presence of concept drift. Surprisingly, uncertainty sampling even with variable and randomized thresholds which are designed to detect changes in data distribution under-performed. In contrast, the QBC strategy outperformed uncertainty sampling. This is because QBC leverages a set of posterior probabilities from multiple models rather than relying on a single model, gathering more information about each instance to assess its informativeness.

TABLE IV: Performance of Various Models on Cover-Type and Hyperplane Datasets for Hoeffding Tree Model (Presence of Serious Concept Drift)

Model	CoverType Dataset	Hyperplane Dataset
Model with all data	0.79848	0.51301
Random model	0.76931	0.42369
Fixed uncertainty	0.76582	0.30452
Variable uncertainty	0.77701	0.40005
Randomized uncertainty	0.77952	0.41486
Fixed QBC	0.77372	0.39579
Variable QBC	0.76831	0.41574
Randomized QBC	0.79106	0.42000
Fixed DBQBC	0.78487	0.37418
Variable DBQBC	0.78886	0.47901
Randomized DBQBC	0.80550	0.49634

The DWQBC which uses both representativeness and informativeness of an instance before querying its original label performed better than the strategies that consumed only the informativeness measure. This emphasizes the idea that the performance of active learning query strategies is not solely influenced by the amount of information taken into account when evaluating instance to query. Hence, one can argue that representativeness is also crucial when determining whether the true label of an instance should be queried.

VII. Conclusion

Strategies that incorporate more information by considering both representativeness and informativeness generally outperform those that use less information. Based on the results obtained, it is evident that in the presence of severe concept drift, the randomized DWQBC strategy outperforms the other two variants of DWQBC. The reasoning behind this is that when significant concept drift occurs, the data distribution changes rapidly, and strategies with fixed or variable thresholds cannot effectively capture these changes. However, the randomized DWQBC strategy, which involves random labeling, includes the drifted data in the labeled set. Since this labeled set is used to compute the representativeness measure, the drifted data is not dismissed as outliers.

References

[1] S. Tong, "Active learning: Theory and applications," Ph.D. dissertation, 2001.

[2] H. Cai, "Active learning for graph embedding," in *Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*. IEEE, 2018, pp. 837–840.

[3] B. Settles, "Active learning with real annotation costs," 2008. [Online]. Available: https://burrsettles.com/pub/settles.nips08ws.pdf

[4] M. Mundt, Y. Hong, I. Pliushch, and V. Ramesh, "A wholistic view of continual learning with deep neural networks: Forgotten lessons and the bridge to active and open world learning," *Neural Networks*, vol. 160, pp. 306–336, 2020. [Online]. Available: https://doi.org/10.1016/j.neunet.2023.01.014

[5] S. Haque, Z. Eberhart, A. Bansal, and C. McMillan, "Semantic similarity metrics for evaluating source code summarization," in *IEEE International Conference on Program Comprehension*, vol. 2022-March, 2022, p. 36–47.

[6] Y. Freund, H. S. Seung, E. Shamir, and N. Tishby, "Selective sampling using the query by committee algorithm," 1991.

[7] A. McCallum and K. Nigam, "A comparison of event models for naïve bayes text classification," in *AAAI-98 workshop on learning for text categorization*, 1998.

[8] B. Settles and M. Craven, "An analysis of active learning strategies for sequence labeling tasks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing - EMNLP '08*, 2008.
[9] A. Das, M. S. Nair, and D. S. Peter, "Batch mode active learning on the riemannian manifold for automated scoring of nuclear pleomorphism in breast cancer," *Artificial Intelligence in Medicine*, vol. 103, p. 101805, 2020. [Online]. Available: https://doi.org/10.1016/j.artmed.2020.101805
[10] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, 1994, pp. 3–12.
[11] H.-L. Nguyen, *Concurrent Semi-supervised Learning with Active Learning of Data Streams*. [Publisher of the book], 2004, ch. Concurrent Semi-supervised Learning with Active Learning of Data Streams, p. [Page range of the chapter], 774 Accesses.
[12] M. Woźniak, P. Ksieniewicz, B. Cyganek, A. Kasprzak, and K. Walkowiak, "Active learning classification of drifted streaming data," *Procedia Computer Science*, vol. 80, pp. 1724–1733, 2016.
[13] I. Žliobaite, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with evolving streaming data," in *Machine Learning and Knowledge Discovery in Databases*, 2011, pp. 587–612.
[14] E. Lughofer, "Single-pass active learning with conflict and ignorance," *Evolving Systems*, pp. 251–271, 2012.
[15] E. Lughofer and M. Pratama, "Online active learning in data stream regression using uncertainty sampling based on evolving generalized fuzzy models," *IEEE Transactions on Fuzzy Systems*, pp. 292–309, 2018.
[16] M. Pratama, S. G. Anavatti, and J. Lu, "Active learning using pre-clustering," *IEEE Transactions on Fuzzy Systems*, vol. 23, no. 6, pp. 2048–2066, 2015.
[17] S. Mohamad, "Active learning for data streams." Ph.D. dissertation, Bournemouth University, 2017.
[18] D. Cacciarelli and M. Kulahci, "Active learning for data streams: a survey," *Machine Learning*, vol. 113, pp. 185–239, 2024.
[19] Z. Zhang, E. Strubell, and E. Hovy, "A survey of active learning for natural language processing," 2023. [Online]. Available: https://arxiv.org/abs/2210.10109
[20] A. Tharwat and W. Schenck, "A survey on active learning: State-of-the-art, practical challenges and research directions," *Mathematics*, vol. 11, no. 4, p. 820, 2023. [Online]. Available: https://doi.org/10.3390/math11040820
[21] M. Datar and S. Muthukrishnan, "Estimating rarity and similarity over data stream windows," 2005.
[22] J. C. Gower, "A general coefficient of similarity and some of its properties," *Biometrics*, vol. 27, no. 4, 1971.
[23] S. Putatunda, *Practical Machine Learning for Streaming Data with Python: Design, Develop, and Validate Online Learning Models*. Apress, 2021.
[24] A. Bifet and G. Gavaldà, "Learning from time-changing data with adaptive windowing," in *Proceedings of the Seventh SIAM International Conference on Data Mining*, Minneapolis, Minnesota, USA, April 26–28 2007.
[25] H. M. Gomes, J. Read, A. Bifet, J. P. Barddal, and J. Gama, "Machine learning for streaming data: state of the art, challenges, and opportunities," *ACM SIGKDD Explorations Newsletter*, vol. 21, no. 2, pp. 6–22, 2019.
[26] R. A. Fisher, "The use of multiple measurements in taxonomic problems," 1936.
[27] M. Harries and N. Wales, "Splice-2 comparative evaluation: Electricity pricing," 1999, dataset.
[28] J. A. Blackard and D. J. Dean, "Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables," *Computers and Electronics in Agriculture*, vol. 24, no. 3, pp. 131–151, 2000.
[29] F. Banar, A. Tabatabaei, and M. Saleh, "Stream data classification with hoeffding tree: An ensemble learning approach," 2023.
[30] A. Bifet, R. Gavalda, G. Holmes, and B. Pfahringer, *Machine Learning for Data Streams: with Practical Examples in MOA*, ser. Adaptive Computation and Machine Learning series. The MIT Press, 2018.
[31] B. Settles, "Active learning literature survey," 2009.

APPENDIX

A. Appendix A: Proof for Equation (3.4)

Data:

Let $f : D \to \mathbb{R}$. There exists a such that $f(a) \geq f(x)$ for all $x \in D$. Also, $b \in D$ exists such that $f(b) \leq f(x)$ for all $x \in D$.

Need to prove:

For all $x\_1, x\_2 \in D$, we have $f(x) - f(b) \geq f(x\_1) - f(x\_2)$.

Proof:

Let $x\_1 \in D$. We have $f(a) \geq f(x\_1)$ from the first equation.

Let $x_2 \in D$. From the second equation, we have $-f(b) \geq f(x_2)$ for all $x_2 \in D$.

By adding the first equation to the second equation, we get:

$$f(x_1) - f(x_2) \leq f(a) - f(b) \quad \text{for all } x \in D.$$

This proves that for all $x_1, x_2 \in D$, we have $f(x) - f(b) \geq f(x_1) - f(x_2)$.

B. Appendix B

The results of the carried-out drift detection tests are as follows. Once a drift is detected, it will be plotted as a red line in the plot.

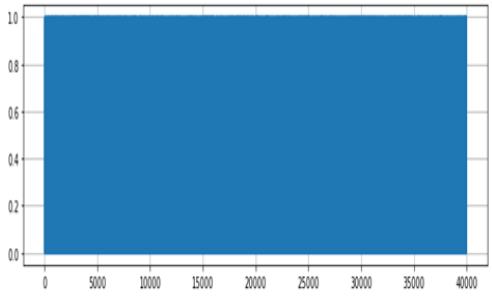

Fig. 15: Drift Detection Plot for Sea Stream Dataset

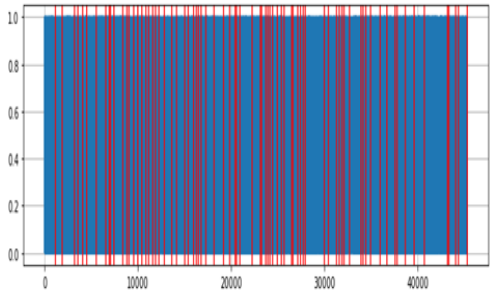

Fig. 16: Drift Detection Plot for electricity Dataset

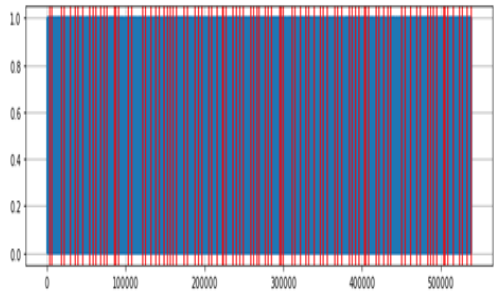

Fig. 17: Drift Detection Plot for Airlines Dataset

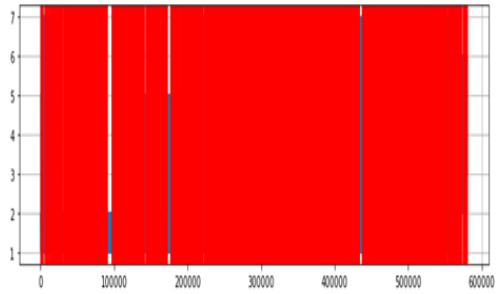

Fig. 18: Drift Detection Plot for Cover Type Dataset

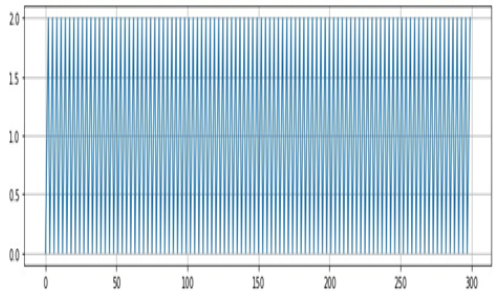

Fig. 19: Drift Detection Plot for Time stamp Iris Dataset

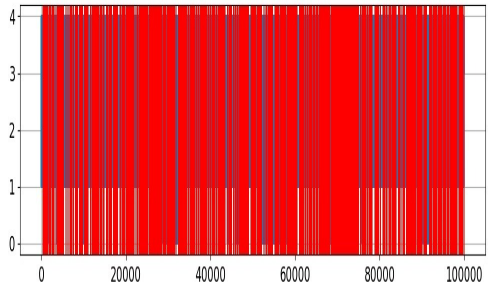

Fig. 20: Drift Detection Plot for hyper_f Dataset

C. Appendix C

The equation for the developed density-based uncertainty active learning query strategy for streaming data utilizing Gower's distance along with margin sampling is as follows.

For Binary classification:

$$x_{DWU} = \arg\min_x \left[P_\theta(\hat{y}_1|X_t) - P_\theta(\hat{y}_2|X_t)\right] \times \left(\frac{1}{L}\sum_{n=1}^{L} \text{dist}(x, x^{(L)})\right)^{\beta}$$

For Multiclass classification (Uses cross entropy):

$$x_{DWU} = \arg\min_x \left[-\sum_i P_\theta(y_i|X_t)\log\left(P_\theta(y_i|X_t)\right)\right] \times \left(\frac{1}{L}\sum_{n=1}^{L} \text{dist}(x, x^{(L)})\right)^{\beta}$$

$P_\theta(y_i \mid x)$: Probability of y_i given$x$ and parameterized by θ,

θ : Threshold parameter,

L : Window size,

β : Density parameter.

D. Appendix D

This section presents the results obtained for the density-based uncertainty query strategy using the same data sets analyzed above.

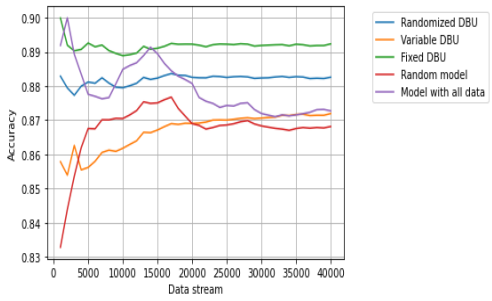

Fig. 21: Density-based uncertainty query strategy for sea stream dataset- HT

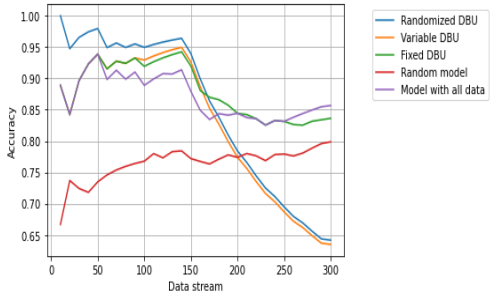

Fig. 22: Density-based uncertainty query strategy for time stamp iris dataset- HT

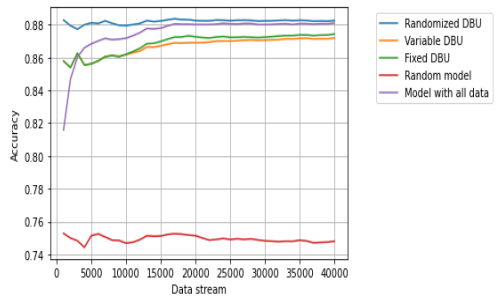

Fig. 23: Density-based uncertainty query strategy for sea stream dataset- NB

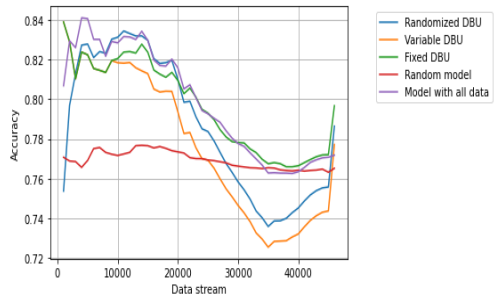

Fig. 24: Density-based uncertainty query strategy for electricity dataset- HT