ML-kNN-H: A Multi-Label Classification Model based on Hoeffding's Inequality

Mashail Althabiti[1,*], Manal Abdullah[1], Omaima Almatrafi[1]
[1]Faculty of Computing and Information Technology, King Abdulaziz University, 21589 Jeddah, Saudi Arabia
Emails: malthabiti0001@stu.kau.edu.sa; maaabdullah@kau.edu.sa; oalmatrafi@kau.edu.sa

Abstract

Multi-label data stream classification plays a crucial role in various applications, including recommendation systems, real-time monitoring systems, smart cities, social media analysis, and healthcare. Its ability to classify constantly generated, potentially unbounded data at a high rate is of utmost importance. Besides accommodating multiple labels, data streams may evolve due to concept drift and bias toward particular classes due to class imbalance. This research introduces the multi-label classification model based on Hoeffding inequality (ML-kNN-H). The proposed model aims to process multi-label data streams, handle concept drift, and class imbalance. ML-kNN-H removes instances introducing errors based on a dynamic value computed from the Hoeffding inequality instead of a fixed value, thereby enhancing the model's efficiency and applicability to different types of data streams. Several experiments have been conducted to assess the model's performance in the presence of concept drift (abrupt and gradual drift) and class imbalance. Particularly, it has been evaluated against six kNN multi-label classifiers on ten datasets: synthetic and real world. The results indicate that ML-kNN-H outperformed the other classifiers on benchmark datasets in terms of Subset Accuracy, Accuracy, Hamming Score, and F-score, except in running time. Statistical analysis has also been utilized to measure the significance of the ML-kNN-H compared to the state-of-the-art classifiers.

Keywords: K Nearest Neighbor; Multi-label Classification; Data Stream; Concept Drift; Hoeffding's Inequality

1. Introduction

Classification is a well-known task in machine learning that predicts one of two possible class labels for a given set of features [1]. Additionally, there are two more types of classification: multi-class and multi-label classification. Multi-class classification is similar to single-class classification in producing only one class label. However, it involves more than two class labels. On the other hand, multi-label classification is a type of classification where the prediction pertains to more than one label from many class labels.

As technologies evolve, data is no longer static. It becomes continuous, flowing at high speed and potentially unbounded, referred to as a data stream [2]. Data streams are massive and come in an ordered sequence over time. Multi-label data streams arrive as instances aligned with class labels [3]. They are accessed at once, so multi-label classifiers are required to process them on the fly. In addition, resource constraints such as limited memory and computational power must be considered while processing data streams [2].

Furthermore, data streams can drift over time due to concept drift, which is the change in the statistical properties of the data [2]. Concept drift could take different forms, including abrupt, gradual, incremental, and recurrent. In the abrupt form, the new concept suddenly replaces the old one. On the contrary, the slow shift from one concept to another is gradual. Incremental drift involves a change that occurs in small and incremental steps. In addition, the concept may reoccur after a particular time, referred to as a recurrent drift. This underscores the need for algorithms to be adaptive, as traditional multi-label algorithms may fail to produce highly accurate predictions. Another issue that might lead to poor performance is class imbalance. It refers to the situation where some class labels are significantly less than other class labels [4]. It is challenging since it causes the model bias to perform better in predicting majority classes and poor in predicting minority classes.

DOI: https://doi.org/10.54216/FPA.190226
Received: January 06, 2025 Revised: February 09, 2025 Accepted: March 08, 2025

Multi-label classification (MLC) of data streams has a wide range of applications in several domains. For example, it can be utilized in content recommendation systems based on user viewing history, in which the content contains more than one label, such as a movie with many genres [5] Another example is MLC employed in intrusion detection systems to monitor the network traffic and detect intrusions, where each can have multiple labels, such as the type of threat and its source [6]. Additionally, in disaster response, it also learns from the data streams and detects and responds to natural disasters such as floods [7].

Strategies used in multi-label classification are categorized into two main categories: Algorithm Adaptation and Problem transformation [8]. It is worth noting that some studies also consider ensemble as a third approach for MLC, which involves integrating multiple classifiers to improve the performance [9]. Methods belonging to problem transformation turn the multi-label classification task into a single classification task, which is then addressed by one of the traditional classifiers—for instance, Binary Relevance, Classifier Chains, and Label Powerset [9]. In contrast, algorithm adaptation methods modify the algorithms to accommodate multi-label data streams. Example of well-known method is Multi-Label k-Nearest Neighbors.

Multi-label k-nearest Neighbors (kNN) based algorithms advantage the kNN classifier to predict the labels of a data stream instance. Such algorithms aim to handle the complexities of multi-label classification. Whereas some algorithms have utilized two memory, long-term memory, and short-term memory, to detect concept drift, as shown by Roseberry & Cano in [3], Zheng & Li in [10], and Wang et al. in [11], others use the single window, sliding or fading, as shown by Roseberry et al. [12], Roseberry et al.[13], and Roseberry et al. [14]. Certain algorithms employ the maximum a posteriori principle for label prediction. This approach is inefficient in data streams because prior and posterior probabilities are recomputed with the arrival of each new instance, leading to high computational requirements [12]. Therefore, the majority vote technique is more commonly used. This technique predicts the labels that occur most frequently, making it simple and highly convenient for data stream processing.

This paper introduces ML-kNN-H, a multi-label classification model for data streams. ML-kNN-H employs a sliding window and utilizes Hoeffding's Inequality to address concept drift and class imbalance. It is a Java-based model developed using Massive Online Analysis (MOA) software [15]. Moreover, experiments with ten datasets from different domains were conducted to assess its performance, showing its potential in real-world applications. The main contribution of this work is proposing a versatile model for MLC for data streams and testing whether the proposed model performed better than state-of-the-art models statistically across different scenarios and datasets.

The remainder of the paper is structured into six main sections. The second section is related work, briefly presenting the multi-label k nearest neighbour algorithms developed for data streams. The third section introduces the proposed model and discusses how it works. Moreover, section four provides the experiments, algorithms used in the comparison, and their parameters. Finally, in sections five and six, the results and conclusion are presented, respectively.

2. Related Work

This section reviews the related work on the multi-label classification of a data stream. All reviewed algorithms employ a multi-label k nearest neighbour as a base classifier. Roseberry and Cano [3] have adopted the SAM-kNN for multi-label data, which was developed for single-label predictions. The proposed method pairs two memories to a simple multi-label kNN classifier in order to handle various types of drift in the stream. A sliding window is also employed as short-term memory to maintain the most recent examples, while a long-term memory remembers historical and infrequent data. Multiple sliding window sizes are evaluated based on the hamming score. So, windows with the highest score are used to take the subsequent instance. Otherwise, instances are sent to the long-term memory. The long-term memory is then compressed via a clustering method whenever it reaches its maximum size.

Roseberry et al. [12] have also proposed a multi-label k nearest neighbour algorithm that utilizes a self-adjusting memory to store recent instances. Imbalance and concept drift are addressed via punitive removal techniques. It penalizes instances introducing errors. The error value is cumulative during the learning process, and instances are removed whenever they exceed a threshold. The method stores the distances between all instances in the window and the window performance during the learning process to be reused, reducing the evaluation time. Similarly, Zheng and Li [10] have adopted a punishment strategy over a sliding window to overcome stream challenges such as concept drift and class imbalance. The proposed algorithm also employs a long-term memory for penalized instances, which is compressed whenever they exceed their maximum size.

Wang et al. [11] have proposed an algorithm that uses a technique that adapts quickly to changes within the data stream. It aggregated two memory models to process drifting data streams like in previous work. However, it differs in that the long-term memory is based on reservoir sampling and the Learning Vector Quantization method.

DOI: https://doi.org/10.54216/FPA.190226
Received: January 06, 2025 Revised: February 09, 2025 Accepted: March 08, 2025

Roseberry et al. [14] proposed an algorithm with adaptive tuning parameters. The algorithm employs a self-adjusting memory with a punitive measure that penalizes instances introducing errors. Neighbour's labels with errors are disabled in the window. If an instance is used again as one of the nearest neighbours and still introduces an error, it will be removed; otherwise, it will be enabled again. In addition, the k value is automatically adjusted with each instance's arrival. In addition, this algorithm has been used as a base classifier with the advantages of ensembles to adapt to concept drift in the multi-label environment [16]. Each classifier in the ensemble is trained on a different subset of the data and monitored by the ADWIN drift detector [17]. The ensemble also adopts a strategy of keeping the accurate base classifiers and removing the weakest classifier.

Roseberry et al. [13] have proposed an algorithm combining different techniques for effective multi-label classification of drifting data streams. It utilizes a fading window; each instance is associated with weight, fading over time. When the window reaches its maximum size, it removes the instances with the lowest weight. Instances are also weighted according to accuracy and removed if the weight is below a threshold. Althabiti et al.[18] have introduced a multi-label classification model with a sliding window to incrementally process the drifting Internet of Things IoT data streams, called IDD-kNN. The model comprises three main steps: classification, diagnosis, and update. It measures the similarities between the newly arrived instance and the selected nearest neighbors. It removes instances from the window if the similarities computed at a time are significantly different from the high similarity reached during the process, which is bounded by a fixed threshold of one.

Building on the IDD-kNN model, which showed improvement over other kNN models, this research aims to further enhance the classification performance and the model's versatility, making it applicable to various domains. Multi-label learning, particularly in dynamic environments, still faces significant challenges. Adaptive multi-label classification models that effectively handle the dynamic nature of data streams while maintaining high-performance need to be developed more.

3. ML-kNN-H Model

ML-kNN-H is a multi-label classification model that handles concept drift and class imbalances in data streams with dynamic value of epsilon changing within the stream. As shown in Figure 1, it generally consists of six parts: input, sliding window, multi-label kNN, neighbors' estimator, drift detector, and prediction.

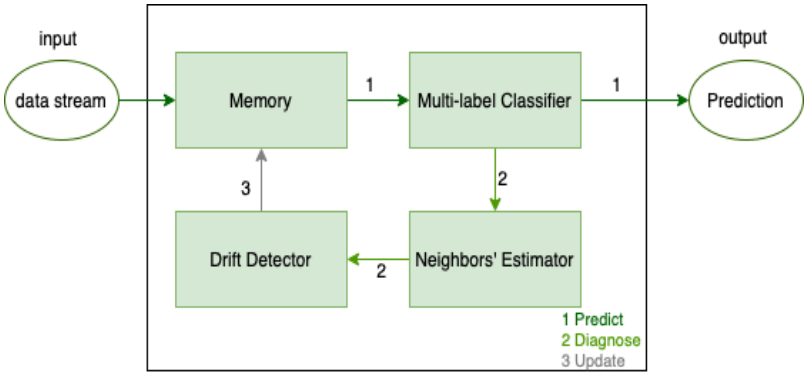

Figure 1. The ML-kNN-H Generic Model.

ML-kNN-H is a supervised learning model where the input to the model is a data stream in the form of (X, Y), where X represents features, and Y represents labels. As shown in Algorithm 1, instances are processed sequentially, where each instance at a time is denoted as s_t, inserting them into a sliding window as short-term memory with predetermined maximum size (line 2). In the classification, Euclidean distance is utilized to identify nearest neighbors NN, which are the instances that have the least distance values between the s_t and all the instances in the window (line 4). Then, the occurrences of each label in the selected NN are counted (lines 5 to 7). After that, each neighbor's (NNi) label set is checked against the new instance's label set (lines 9 to 12). If their label set is completely the same, the counter of a number of neighbors having similar labels is incremented by one. Otherwise, the error of each neighbor is calculated by comparing the label set of NNi to the label set of the current instance in which each unmatched label in the label set is counted as 1. If there are two unmatched labels, then the error becomes 2, and so on.

To make the prediction, the relative frequency is computed for each label in the neighbors' set to determine the final output label set, whose relative frequency is greater than 0.5 (lines 16-22). After the prediction is made, ML-kNN-H diagnoses for concept drift by calculating the probability of similarity of labels for the entire nearest neighbor (PNN_t), computed using Equation 1 (line 27).

DOI: https://doi.org/10.54216/FPA.190226
Received: January 06, 2025 Revised: February 09, 2025 Accepted: March 08, 2025

$$PNN_t = \frac{\textit{Number of Neighbors having similar labels to } s_t}{k} \quad (1).$$

Algorithm 1 Pseudo Code of ML-kNN-H Model

Require: $k, winodw[] = null, w(max), Votes[] = null, Predictions[] = null, PNN_t = 0, PNN_{max} = 0, error = 0.$

1: **while** having a stream **do**
2: Insert the instance S_t in the window
3: $n = window.size$
4: Look into the k nearest neighbors from the window ▷ Euclidean distance
5: **for** for each neighbor NN_i **do**
6: **for** for each label l in L **do**
7: Votes[l] = Votes[l] +1 ▷ Collect occurrences of labels
8: **end for**
9: **if** instance.labels = = neighbor.labels **then**
10: NumberOfNeighborsHavingSimilarLabels++ ▷ Count the number of neighbors that have similar labels to the instance actual labels.
11: **else**
12: Neighbor Error= Number of Dissimilar Labels
13: **end if**
14: **end for**
15: **Prediction**
16: **for** each label l in L **do**
17: **if** $votes[l]/k > 0.5$ **then**
18: Predictions[l] = 1
19: **else**
20: Predictions[l] = 0
21: **end if**
22: FinalPredictions = Predictions
23: **end for**
24: **Check the Neighbors**
25: $\epsilon \leftarrow \sqrt{\left(\frac{1}{2n}\right) \cdot \ln\left(\frac{1}{\delta}\right)}$
26: **Calculate** PNN_t
27: $PNN_t = (NumberOfNeighborsHavingSimilarLabels)/(k)$ ▷ Probability of similarity of labels for all the selected neighbors
28: **if** $PNN_{max} < PNN_t$ **then**
29: $PNN_{max} = PNN_t$
30: **end if**
31: $\Delta = PNN_{max} - PNN_t$
32: **if** $\Delta P > \epsilon$ **then**
33: **for** each neighbor NN_i **do**
34: **if** Neighbor Error/number of labels =0.5 **then**
35: Delete the neighbor from the window
36: **end if**
37: **end for**
38: **end if**
39: **Update The Window Size**
40: **if** n>w(max) **then**
41: Remove oldest instance in the window.
42: **end if**
43: **end while**

The highest value for PNN_t is always stored in a variable called PNN_{max}, which is initially set to zero and updated when PNN_t has a greater value (lines 28 & 29). ML-kNN-H detects concept drift if a significant differences between PNN_{max} and PNN_t is found, which is bounded by an epsilon value based on the Hoeffding's inequality (line 32). In probability theory, the Hoeffding's inequality sets the upper bound for the allowed deviation between random variables and their expected value [19]. Hoeffding's inequality has also been used in machine learning algorithms to make reliable decisions such as in [20] and[21]. It bounds the probability that sums of bounded random variables are too large or too small. ML-kNN-H utilizes Hoeffding's inequality to find the epsilon value which is calculated using Equation 2 (line 25).

$$(\varepsilon) = \sqrt{\frac{1}{2 * window\ size}\ In\ \frac{1}{\delta}} \quad (2).$$

The epsilon value indicates how far it is acceptable to deviates from PNN_{max} to PNN_t, given the confidence δ. This value is computed at each instance arrival, which means that it changes each time.

If the difference exceeds the epsilon value, ML-kNN-H checks the selected nearest neighbors' errors and deletes neighbors with an error rate above 0.5 (lines 34 & 35). The removal task adopted in the model evaluates each instance's impact on the classifier performance and removes instances contributing to errors, promoting balanced classes. This is based on the assumption that instances leading to errors are likely to be skewed towards the majority classes [12]. Instances with low error rates are kept in the window, assuming they may belong to new concepts. If no significant difference is found then it will take the subsequent instance s_{t+1}. Furthermore, if the window reaches its predetermined maximum size, it slides the instances by removing the old instance in the window (lines 40 & 41) and takes a new instance (going back to line 1).

DOI: https://doi.org/10.54216/FPA.190226
Received: January 06, 2025 Revised: February 09, 2025 Accepted: March 08, 2025

- Example of How the ML-kNN-H Model Works

Figure 2 illustrates an example of how the proposed model works. Suppose k=3, the maximum window size is ten, and the window is occupied with eight instances, and a new instance arrives s_t. After the instance is inserted and the prediction z is made, the epsilon-based Hoeffding inequality ε is computed, equal to 0.876. After that, PNN_t, is calculated, and PNN_{max}, takes the value since it is greater than 0.3 (computed for s_{t-1}arrival). The difference here between the two variables did not exceed the epsilon; therefore, no action was taken. The window size is checked against the maximum size allowed, and there is no action here since it is below the maximum size (10). The second part shows the arrival of s_{t+1}, where a similar scenario unfolds. The only differences are the epsilon value, which is 0.831, and the window size at its maximum of ten. The model adapts by removing old instances to make space for the next one. The third part displays the model's ability to detect a significant change as the difference exceeded 0.831. Two instances with errors above 0.5 are promptly removed from the window, demonstrating the model's flexibility and adaptability.

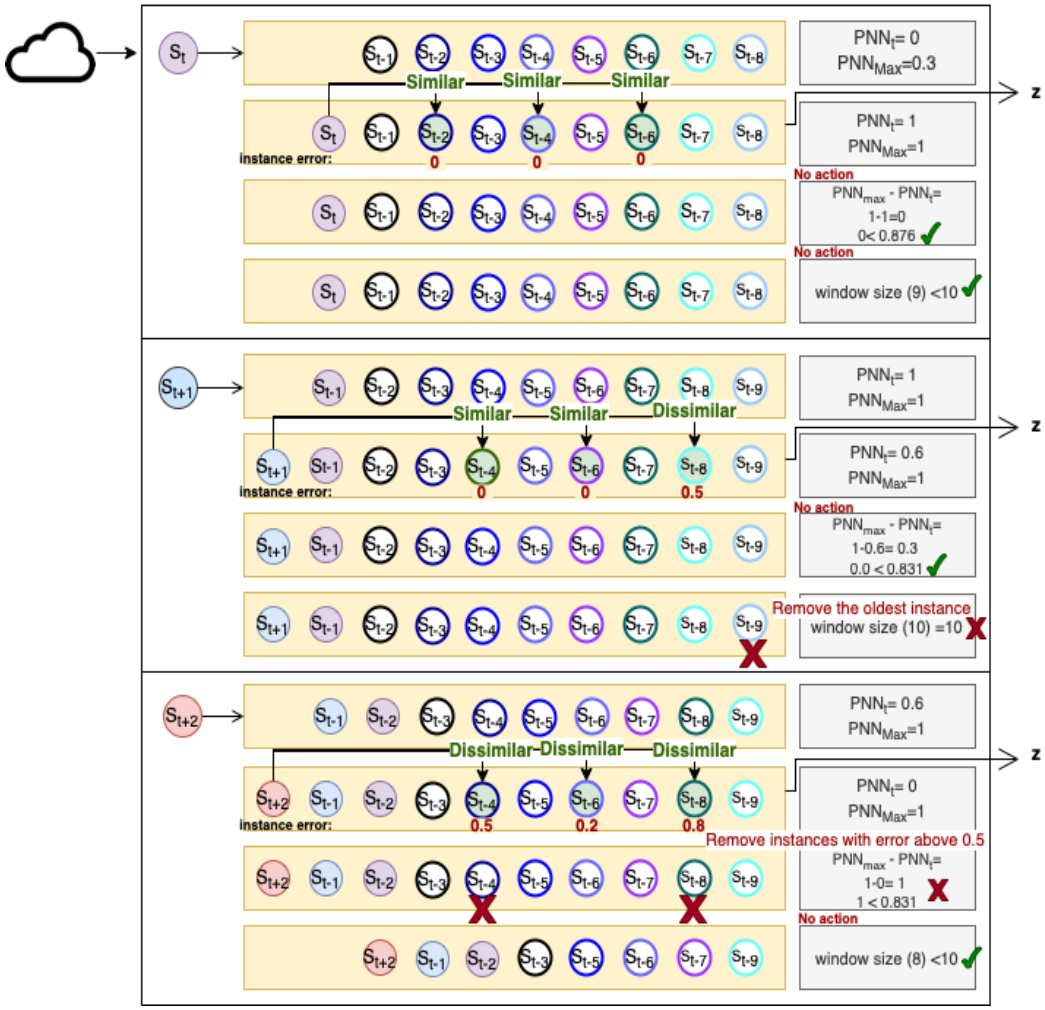

Figure 2. Example of How the ML-kNN-H Model Works.

4. Experimental Setting

This section presents the experimental setting, where the model is executed on ten datasets using the Massive Online Analysis tool (MOA) [15]. The prequential evaluation method is used, with the model undergoing testing, training, and updating for each incoming instance [22]. ML-kNN-H's prediction performance is assessed using five performance metrics. The first metric is the Subset Accuracy, which measures the ratio of instances predicted with all labels exactly as the truth label to the total number of predicted instances. The second metric is Accuracy, which measures the ratio of correctly predicted labels to the union of true and predicted labels. The third metric used is the Hamming Score, which measures the ratio of correctly predicted labels to the total number of labels, then averaged over all instances. The fourth metric is the F-measure, the harmonic mean of precision and recall, and the last metric is the running time, which the time is taken to run the model and get the result.

DOI: https://doi.org/10.54216/FPA.190226
Received: January 06, 2025 Revised: February 09, 2025 Accepted: March 08, 2025

A. Parameters

Five state-of-the-art classifiers and one baseline classifier are selected for the comparison analysis; all of them are multi-label classifiers based on kNN algorithm. The experiments employ the default parameters as they set in MOA tool [15], presented in Table 1.

Table 1: Models' Parameters Used in the Experiments.

Methods	Parameters
ML-kNN-H	K=3, MaxWindowSize =2,000, The confidence level= 0.000001.
IDD-kNN	K=3, MaxWindowSize =1,000, threshold=1.
ArkNN	K=3, windowSize=1,000, fitness=0.001.
ODM	K=3, windowSize=50, Sizeofclusters=4, ReservoirSize =100.
AESAKNNS	BaseLeaner: MLSAkNNSubspaces, ensembleSize=10, windowSize=1,000.
ML-KNN	K=10, windowSize=1,000.

B. Datasets

Two types of multi-label data streams are utilized to compare the proposed model's performances against other ML classifiers: synthetics and real-world data. In this study, four synthetic datasets were generated using the Radial Basis Function (RBF) supported in MOA [15]. Additionally, six real-world datasets [23] are used, each representing different fields of application, each described below:

- RBF-Abrupt - 3 Labels: A synthetic dataset contains abrupt drift, which is simulated by generating a data stream with 50 centroids, which is then changed to 90 centroids after a 50k instance. It has 10 features, 3 labels, and 100k instances. On average, each instance is associated with 1.68 labels.
- RBF-Abrupt- 5 Labels: A synthetic dataset contains abrupt drift, which is simulated by generating a data stream with 50 centroids, which is then changed to 90 centroids after a 50k instance. It has 10 features, 5 labels, and 100k instances. On average, each instance is associated with three labels.
- RBF- Gradual- 3 Labels: A synthetic dataset contains gradual drift, which is simulated by generating a data stream with 50 centroids, which is then changed to 90 centroids after 30k instances. The change from one concept to another takes 20k instances. It has 10 features, 3 labels, and 100k instances. On average, each instance is associated with 1.68 labels.
- RBF- Gradual- 5 Labels: A synthetic dataset contains gradual drift, which is simulated by generating a data stream with 50 centroids, which is then changed to 90 centroids after 30k instances. The change from one concept to another takes 20k instances. It has 10 features, 5 labels, and 100k instances. On average, each instance is associated with three labels.
- Birds: An audio-based dataset. It has 260 features, 19 labels, and 645 instances. On average, each instance is associated with 1.014 labels.
- EMOTIONS: A music-based dataset. It has 72 features, 6 labels, and 593 instances. On average, each instance is associated with 1.868 labels.
- CHD_49: A medicine-based dataset. It has 49 features, 6 labels, and 555 instances. On average, each instance is associated with 2.58 labels.
- Flags: An image-based dataset. It has 19 features, 7 labels, and 194 instances. On average, each instance is associated with 3.392 labels.
- Genbase: A biology-based dataset. It has 1186 features, 27 labels, and 662 instances. On average, each instance is associated with 1.252 labels.
- Medical: A Text-based dataset. It has 1449 features, 45 labels, and 978 instances. On average, each instance is associated with 1.245 labels.

DOI: https://doi.org/10.54216/FPA.190226
Received: January 06, 2025 Revised: February 09, 2025 Accepted: March 08, 2025

5. Results and Discussion

The results of running the ML-kNN-H method on ten datasets are organized based on the following performance metrics. The results also show the average ranking obtained using the Friedman test, which will be explained in the Statistical Analysis section.

A. Subset-Accuracy

Subset Accuracy is strict, ensuring that all labels are exactly predicted. ML-kNN-H achieved high subset accuracy in all four synthetic datasets as shown in Table 2. In the real dataset, ML-kNN-H also got the highest subset accuracy in all the datasets except CHD-49 and Genbase, where AESAKNNS is the superior method, yet ML-kNN-H achieved the second best in these two datasets.

Table 2: Subset Accuracy Metric Results for Ten Synthetic and Real-world Datasets

Datasets / Methods	ML-kNN-H	IDD-kNN	ArkNN	ODM	AESAKNNS	ML-kNN
RBF-Abrupt- 3 Labels	0.693	0.662	0.603	0.434	0.66	0.623
RBF-Abrupt- 5 Labels	0.338	0.295	0.232	0.13	0.211	0.274
RBF- Gradual- 3 Labels	0.659	0.633	0.556	0.387	0.618	0.58
RBF- Gradual- 5 Labels	0.322	0.278	0.22	0.126	0.21	0.264
Birds	0.499	0.499	0.473	0.472	0.464	0.488
EMOTIONS	0.248	0.235	0.211	0.215	0.127	0.214
CHD_49	0.158	0.157	0.104	0.121	0.219	0.11
Flags	0.11	0.105	0.089	0.079	0.074	0.05
Genbase	0.853	0.853	0.821	0.752	0.859	0.625
Medical	0.387	0.387	0.36	0.328	0.298	0.286
Average Ranking	1.35	2.05	4.3	4.9	4	4.4

B. Accuracy

ML-kNN-H outperformed most methods in terms of accuracy in seven datasets, as shown in Table 3. It followed AESAKNNS in CHD-49 and Genbase as the second-best method. Also, AESAKNNS scored 0.494 in the Flags dataset, followed by ML-kNN, IDD-kNN, and ML-kNN-H, all with similar scores of 0.47. It is important to note that AESAKNNS is an ensemble of self-adjusting kNN classifiers, resulting in effective prediction performance. Accuracy here measures the average similarity between the predicted and true label sets. Results on the Birds dataset show that classifiers may predict incorrect labels or miss labels that should be predicted.

Table 3: Accuracy Metric Results for Ten Synthetic and Real-world Datasets.

Datasets / Methods	ML-kNN-H	IDD-kNN	ArkNN	ODM	AESAKNNS	ML-kNN
RBF-Abrupt- 3 Labels	0.806	0.789	0.747	0.65	0.798	0.767
RBF-Abrupt- 5 Labels	0.654	0.623	0.571	0.476	0.572	0.624
RBF- Gradual- 3 Labels	0.786	0.772	0.719	0.621	0.776	0.744
RBF- Gradual- 5 Labels	0.647	0.614	0.563	0.478	0.572	0.618

DOI: https://doi.org/10.54216/FPA.190226
Received: January 06, 2025 Revised: February 09, 2025 Accepted: March 08, 2025

Birds	0.096	0.096	0.07	0.051	0.077	0.037
EMOTIONS	0.494	0.463	0.434	0.426	0.36	0.415
CHD_49	0.504	0.493	0.434	0.487	0.556	0.467
Flags	0.475	0.475	0.449	0.46	0.494	0.478
Genbase	0.882	0.882	0.853	0.787	0.89	0.652
Medical	0.459	0.459	0.445	0.401	0.371	0.353
Average Ranking	1.6	2.6	4.6	5.1	2.9	4.2

C. Hamming Score

Hamming Score provides detailed label-wise prediction performance. ML-kNN-H has the best Hamming score in seven datasets, as shown in Table 4. It also got the second-best score in three datasets, CHD_49, Flags, and Genbase, following AESAKNNS. In the Birds dataset, we noticed before that all classifiers struggle to predict the exact label sets, as shown by the subset accuracy. However, they perform better at predicting individual labels correctly. Results of ML-kNN-H show that 95.2% of the individual labels in Birds dataset are correctly predicted, making it accurate at the individual label level.

Table 4: Hamming Score Metric Results for Ten Synthetic and Real-world Datasets.

Datasets / Methods	ML-kNN-H	IDD-kNN	ArkNN	ODM	AESAKNNS	ML-kNN
RBF-Abrupt- 3 Labels	0.854	0.84	0.804	0.738	0.85	0.832
RBF-Abrupt- 5 Labels	0.752	0.732	0.686	0.633	0.71	0.735
RBF- Gradual- 3 Labels	0.837	0.826	0.779	0.718	0.831	0.813
RBF- Gradual- 5 Labels	0.747	0.725	0.679	0.632	0.71	0.731
Birds	0.952	0.952	0.945	0.945	0.948	0.949
EMOTIONS	0.775	0.757	0.748	0.746	0.723	0.764
CHD_49	0.689	0.689	0.664	0.677	0.732	0.682
Flags	0.657	0.657	0.656	0.643	0.672	0.608
Genbase	0.99	0.99	0.987	0.983	0.991	0.979
Medical	0.978	0.978	0.975	0.976	0.976	0.977
Average Ranking	1.55	2.55	4.95	5.4	2.95	3.6

D. F-score

The F-score balances classifier recall and precision. ML-kNN-H performed better on five different datasets as presented in Table 5. In addition, AESAKNNS achieved the best F-score in three datasets, followed by ML-kNN-H, which was the second-best method in two of them. Also, ML-kNN-H and IDD-kNN same score in RBF-Abrupt-(3,5) Labels dataset. Overall, ML-kNN-H, IDD-kNN, and AESAKNNS consistently perform well in all the datasets, demonstrating strong capabilities in maintaining high precision and recall even in moderate to high imbalanced data such as Medical and Genbase.

DOI: https://doi.org/10.54216/FPA.190226
Received: January 06, 2025 Revised: February 09, 2025 Accepted: March 08, 2025

Table 5: F-score Metric Results for Ten Synthetic and Real-world Datasets.

Datasets / Methods	ML-kNN-H	IDD-kNN	ArkNN	ODM	AESAKNNS	ML-kNN
RBF-Abrupt- 3 Labels	0.847	0.834	0.801	0.728	0.847	0.818
RBF-Abrupt- 5 Labels	0.744	0.718	0.674	0.591	0.681	0.724
RBF- Gradual- 3 Labels	0.832	0.822	0.779	0.705	0.833	0.803
RBF- Gradual- 5 Labels	0.739	0.711	0.668	0.594	0.681	0.721
Birds	0.112	0.112	0.085	0.061	0.093	0.044
EMOTIONS	0.578	0.545	0.514	0.503	0.444	0.485
CHD_49	0.613	0.603	0.532	0.598	0.656	0.585
Flags	0.599	0.602	0.566	0.582	0.626	0.609
Genbase	0.89	0.89	0.862	0.798	0.899	0.661
Medical	0.484	0.484	0.474	0.425	0.395	0.377
Average Ranking	1.8	2.55	4.6	5.1	2.75	4.2

E. Running Time

ODM, the fastest model in this analysis, processes the data stream in the shortest time across all datasets. ArkNN, IDD-kNN, and ML-kNN-H. Follow it however; ML-kNN's performance was relatively poor due to the calculation of prior and posterior probabilities. AESAKNNS also resulted in slow runtime due to ensemble complexity. It is worth mentioning that the results are an average of ten runs.

Table 6: Running Time Metric Results for Ten Synthetic and Real-world Datasets.

Datasets / Methods	ML-kNN-H	IDD-kNN	ArkNN	ODM	AESAKNNS	ML-kNN
RBF-Abrupt- 3 Labels	210.7	69.9	27.4	8.3	1064.5	22153.7
RBF-Abrupt- 5 Labels	259.8	168.2	12.4	4.9	820.2	25356.6
RBF- Gradual- 3 Labels	193.3	70.5	20.5	5.4	838.3	21922.5
RBF- Gradual- 5 Labels	170.1	120.1	10.8	3.9	820.8	14363.7
Birds	1.3	1.2	0.5	0.2	34.8	311.4
EMOTIONS	0.2	0.3	0.2	0.1	3.9	87.9
CHD_49	0.06	0.1	0.6	0.08	4.9	34.7
Flags	0.01	0.01	0.01	0.01	1.3	1.4
Genbase	3.6	3.5	2.3	0.8	76.9	1248.7
Medical	0.8	0.6	1.1	0.4	177.9	499.2
Average Ranking	3.3	2.95	2.5	1.25	5	6

DOI: https://doi.org/10.54216/FPA.190226
Received: January 06, 2025 Revised: February 09, 2025 Accepted: March 08, 2025

F. Statistical Analysis

The Friedman test is used to compare the learning models [24]. It is a non-parametric test that ranks the models for each data set based on their performance; the best model gets one, the second model gets two, and so on. The average ranking for each model is then used to test the null hypothesis, stating that all the models are equivalent. Based on our results, the null hypothesis is rejected at a=0.05, which means that the models in the comparison are not equal.

Since the null hypothesis is rejected, we can proceed with another step, allowing us to compare two classifiers. The Nemenyi test, which is a post-hoc test, is used for this purpose [24]. It indicates that there is a significant difference between two classifiers if their average ranks differ by at least the critical difference. The critical difference depends on the number of models being compared and the number of datasets, equal to $CD_{\text{a=0.05}}$ = 2.3 in our experiment.

The critical difference (CD) diagram in Figure 3 presents the results in a graphical format, comparing each model's average rank on ten datasets using the Subset Accuracy metric. It shows significant performance differences between ML-kNN-H and the other models, except IDD-kNN. In terms of subset accuracy, IDD-kNN-H performs significantly better than ArkNN, ODM, ML-KNN, and AESAKNNS.

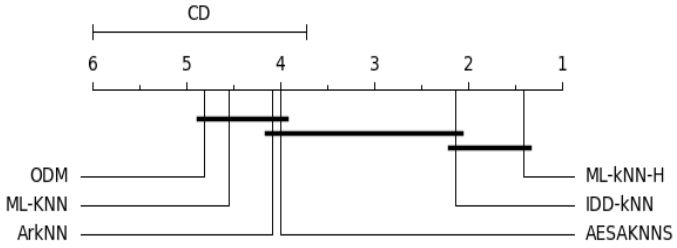

Figure 3. Nemenyi Post Hoc Test for Subset Accuracy Metric.

Figure 4 exhibits the ranking of the models based on their accuracy performance. IDD-kNN-H performs significantly better than ArkNN, ODM, and ML-KNN. Nevertheless, the differences between (ML-kNN-H and IDD-kNN) and (ML-kNN-H and AESAKNNS) are insignificant, indicating statistically similar performance across the datasets.

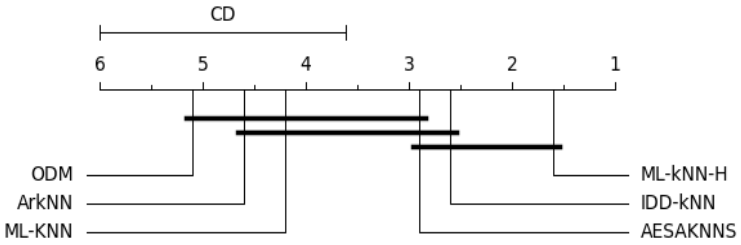

Figure 4. Nemenyi Post Hoc Test for Accuracy Metric.

Regarding the Hamming Score, ML-kNN-H performs significantly better than ArkNN and ODM, as shown in Figure 5. The differences between (IDD-kNN-H and IDD-kNN), (IDD-kNN-H and ML-KNN), and (IDD-kNN-H and AESAKNNS) are not significant.

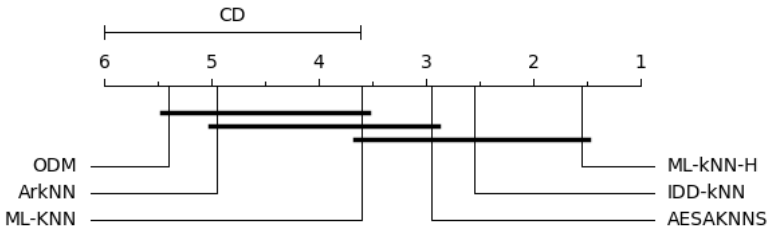

Figure 5. Nemenyi Post Hoc Test for Hamming Score Metric.

Similar to accuracy, IDD-kNN-H performs significantly better than ArkNN, ODM, and ML-KNN regarding F-score. While no significant difference is found between (IDD-kNN-H and IDD-kNN) and (IDD-kNN-H and AESAKNNS), as shown in Figure 6.

DOI: https://doi.org/10.54216/FPA.190226
Received: January 06, 2025 Revised: February 09, 2025 Accepted: March 08, 2025

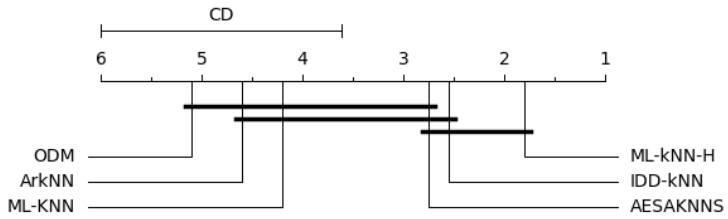

Figure 6. Nemenyi Post Hoc Test for F-score Metric.

Furthermore, as shown in Figure 7, ODM is the best-performing model in terms of model running time, with the lowest average rank. However, the differences between (ODM and ML-kNN-H), (ODM, IDD-kNN), and (ODM and ArkNN) are not statistically significant. Also, ML-KNN performs significantly worse than ML-kNN-H, IDD-kNN, ArkNN, and ODM.

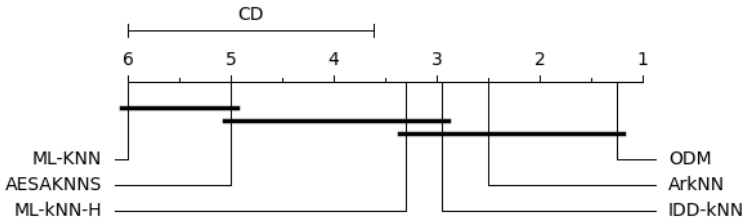

Figure 7. Nemenyi Post Hoc Test for Running Time Metric.

6. Conclusion

In this study, we proposed a multi-label classification model with drift detection in imbalanced data streams. The proposed method, which incorporates an adaptive epsilon value, has been tested on ten datasets from different domains. The experiment results demonstrate our method's effectiveness in handling non-stationary data streams. It improves the Subset Accuracy, Accuracy, Hamming Score, and F-score compared to state-of-the-art methods. ML-kNN-H method has been developed for processing data streams in various real-world applications, and its importance lies in its ability to handle data that is evolving over time and adapt to new patterns as they emerge. Future work will focus on further adopting feature selection techniques and hyperparameter tuning for the k value. Also, using other distance metrics to enhance the performance and efficiency of the classifier.

Funding: "This research received no external funding"

Conflicts of Interest: "The authors declare no conflict of interest."

References

[1] F. Herrera, F. Charte, A. J. Rivera, and M. J. del Jesus, *Multilabel Classification: Problem Analysis, Metrics and Techniques*, 1st ed. Springer Publishing Company, Incorporated, 2016.

[2] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv*, vol. 1, no. 1, 2013, doi: 10.1145/0000000.0000000.

[3] M. Roseberry and A. Cano, "Multi-label kNN classifier with self-adjusting memory for drifting data streams," in *Proc. Mach. Learn. Res.*, 2018, pp. 23–37.

[4] F. Charte, A. J. Rivera, M. J. del Jesus, and F. Herrera, "Addressing imbalance in multilabel classification: Measures and random resampling algorithms," *Neurocomputing*, vol. 163, pp. 3–16, Sep. 2015, doi: 10.1016/j.neucom.2014.08.091.

[5] S. Kumar, N. Kumar, A. Dev, and S. Naorem, "Movie genre classification using binary relevance, label powerset, and machine learning classifiers," *Multimed. Tools Appl.*, vol. 82, no. 1, pp. 945–968, 2023, doi: 10.1007/s11042-022-13211-5.

[6] E. Hallaji, R. Razavi-Far, and M. Saif, "Expanding analytical capabilities in intrusion detection through ensemble-based multi-label classification," *Comput. Secur.*, vol. 139, Apr. 2024, doi: 10.1016/j.cose.2024.103730.

DOI: https://doi.org/10.54216/FPA.190226
Received: January 06, 2025 Revised: February 09, 2025 Accepted: March 08, 2025

[7] B. K. Mishra, D. Thakker, S. Mazumdar, D. Neagu, M. Gheorghe, and S. Simpson, "A novel application of deep learning with image cropping: A smart city use case for flood monitoring," *J. Reliab. Intell. Environ*, vol. 6, no. 1, pp. 51–61, Mar. 2020, doi: 10.1007/s40860-020-00099-x.

[8] G. Tsoumakas and I. Katakis, "Multi-label classification: An overview," in *Int. J. Data Warehous. Min.*, IGI Publishing, 2007, pp. 1–13, doi: 10.4018/jdwm.2007070101.

[9] X. Zheng, "A survey on multi-label data stream classification," *IEEE Access*, vol. 8, pp. 1249–1275, 2020, doi: 10.1109/ACCESS.2019.2962059.

[10] X. Zheng and P. Li, "An efficient framework for multi-label learning in non-stationary data stream," in *Proc. 12th IEEE Int. Conf. Big Knowl.*, ICBK 2021, IEEE, 2021, pp. 149–156, doi: 10.1109/ICKG52313.2021.00029.

[11] X. Wang, P. Kuntz, F. Meyer, and V. Lemaire, "Multi-label kNN classifier with online dual memory on data stream," in *IEEE Int. Conf. Data Min. Workshops*, ICDMW, IEEE, 2021, pp. 405–413, doi: 10.1109/ICDMW53433.2021.00056.

[12] M. Roseberry, B. Krawczyk, and A. Cano, "Multi-label punitive kNN with self-adjusting memory for drifting data streams," *ACM Trans. Knowl. Discov. Data*, vol. 13, no. 6, Oct. 2019, doi: 10.1145/3363573.

[13] M. Roseberry, S. Džeroski, A. Bifet, and A. Cano, "Aging and rejuvenating strategies for fading windows in multi-label classification on data streams," in *Proc. ACM Symp. Appl. Comput.*, ACM, Mar. 2023, pp. 390–397, doi: 10.1145/3555776.3577625.

[14] M. Roseberry, B. Krawczyk, Y. Djenouri, and A. Cano, "Self-adjusting k-nearest neighbors for continual learning from multi-label drifting data streams," *Neurocomputing*, vol. 442, pp. 10–25, Jun. 2021, doi: 10.1016/j.neucom.2021.02.032.

[15] A. Bifet, "MOA: Massive Online Analysis Learning Examples," *J. Mach. Learn. Res.*, vol. 11, pp. 1601–1604, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1859890.1859903.

[16] G. Alberghini, S. Barbon Junior, and A. Cano, "Adaptive ensemble of self-adjusting nearest neighbor subspaces for multi-label drifting data streams," *Neurocomputing*, vol. 481, pp. 228–248, Apr. 2022, doi: 10.1016/j.neucom.2022.01.075.

[17] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *Proc. SIAM Int. Conf. Data Min.*, 2013, pp. 443–448, doi: 10.1137/1.9781611972771.42.

[18] M. Althabiti, M. Abdullah, and O. Almatrafi, "Multi-label classification for drift detection in IoT data streams," *Commun. Math. Appl.*, vol. 15, 2024.

[19] W. Hoeffding, "Probability inequalities for sums of bounded random variables," in *The Collected Works of Wassily Hoeffding*, N. I. Fisher and P. K. Sen, Eds. Springer, New York, 1994, pp. 409–426, doi: 10.1007/978-1-4612-0865-5_26.

[20] I. Frías-Blanco, J. Del Campo-Ávila, G. Ramos-Jiménez, R. Morales-Bueno, A. Ortiz-Díaz, and Y. Caballero-Mota, "Online and non-parametric drift detection methods based on Hoeffding's bounds," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 3, pp. 810–823, Mar. 2015, doi: 10.1109/TKDE.2014.2345382.

[21] A. Pesaranghader and H. Viktor, "Fast Hoeffding drift detection method for evolving data streams," in *Mach. Learn. Knowl. Discov. Databases*, Springer, Cham, 2016, pp. 96–111, doi: 10.1007/978-3-319-46227-1_7.

[22] X. Wang, P. Kuntz, F. Meyer, and V. Lemaire, "Multi-label kNN classifier with online dual memory on data stream," in *IEEE Int. Conf. Data Min. Workshops*, ICDMW, IEEE, 2021, pp. 405–413, doi: 10.1109/ICDMW53433.2021.00056.

[23] "Multi-label classification dataset repository," *Knowl. Discov. Intell. Syst. KDIS*, Univ. Córdoba. Accessed: Jul. 14, 2024. [Online]. Available: https://www.uco.es/kdis/mllresources/.

[24] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," 2006.

DOI: https://doi.org/10.54216/FPA.190226
Received: January 06, 2025 Revised: February 09, 2025 Accepted: March 08, 2025