Real-time feature selection technique with concept drift detection using adaptive micro-clusters for data stream mining

Article

Accepted Version

Hammoodi, M. S., Stahl, F. ORCID: https://orcid.org/0000-0002-4860-0203 and Badii, A. (2018) Real-time feature selection technique with concept drift detection using adaptive micro-clusters for data stream mining. Knowledge-Based Systems, 161. pp. 205-239. ISSN 0950-7051 doi: 10.1016/j.knosys.2018.08.007 Available at https://centaur.reading.ac.uk/78678/

Real-Time Feature Selection Technique with Concept Drift Detection using Adaptive Micro-Clusters for Data Stream Mining

Mahmood Shakir Hammoodi[a,b], Frederic Stahl[a], Atta Badii[a]

[a]*Department of Computer Science, University of Reading, PO Box 225, Whiteknights, Reading, RG6 6AY, UK*
[b]*Department of Computer Center, University of Babylon, Iraq*

Abstract

Data streams are unbounded, sequential data instances that are generated with high *Velocity.* Classifying sequential data instances is a very challenging problem in machine learning with applications in network intrusion detection, financial markets and applications requiring real-time sensor-networks-based situation assessment. Data stream classification is concerned with the automatic labelling of unseen instances from the stream in real-time. For this the classifier needs to adapt to concept drifts and can only have a single pass through the data if the stream is fast moving. This research paper presents work on a real-time pre-processing technique, in particular feature tracking. The feature tracking technique is designed to improve Data Stream Mining (DSM) classification algorithms by enabling and optimising real-time feature selection. The technique is based on tracking adaptive statistical summaries of the data and class label distributions, known as Micro-Clusters. Currently the technique is able to detect concept drifts and identify which features have been influential in the drift.

Keywords: Data Stream Mining, real-time Feature Selection, Concept Drift Detection

1. Introduction

Velocity in Big Data Analytics [1] refers to data that is generated at ultra high speed and is live-streamed whereupon the processing and storing of it in real-time (Data Stream Mining, DSM) constitute significant challenges to current computational capabilities [2]. Thus data stream mining algorithms that are capable of learning over a single-pass through the training data are necessary. The general area of Data Stream Mining covered by this paper is Data Stream Classification, which is the prediction of class labels of new instances in the data stream in real-time. Potential applications that need real-time data stream classification techniques are for data streams in the chemical process industry [3], intrusion detection in telecommunications [4], etc. In order to keep as high a predictive accuracy as possible, data stream classification techniques need not only be able to learn incrementally but also be able to adapt to concept drifts.

A concept drift occurs if the pattern encoded in the data stream changes. DSM has developed various real-time versions of established predictive data mining algorithms that adapt to concept drift and keep the model accurate over time, such as CVFDT [5] and G-eRules [6]. The benefit of classifier independent concept drift detection methods is that they give information about the dynamics of data generation [7]. Common drift detection methods are for example ADaptive sliding WINdow (ADWIN) [8], Drift Detection Method (DDM) [9] and the Early Drift Detection Method (EDDM)[10]. However, no drift detection method devised to-date, can provide potentially highly valuable insights as to which features are involved in the concept drift. For example, if a feature is contributing to a concept drift it can be assumed that the feature may have become either more or less relevant to the current concept. This causal responsibility theoretic perspective of the evaluation of concept drift has inspired the development of a real-time feature tracking method based on feature contribution information for the purpose of feature selection to identify features that have become (more) relevant or irrelevant due to concept drift. Thus, an approach for detecting causality of drifts, providing the feature contribution information for the purpose of tracking features and identifying the relevant features for classification for the purpose of feature selection in real-time has been developed in this research.

Common feature selection techniques are for example Linear Discriminant Analysis (LDA), Canonical Correlation Analysis (CCA), Multi-View CCA, Principal Component Analysis (PCA) [11, 12], and Support Vector Machine (SVM) based techniques. These techniques can be applied on a sample of the data stream before commencing the training and adaptation of a data stream classifier. However, this would not account for changes in the relevance of features for the classification task at hand due

Email addresses: pre.mahmood.shakir@uobabylon.edu.iq, m.s.h.hammoodi@pgr.reading.ac.uk (Mahmood Shakir Hammoodi), F.T.Stahl@reading.ac.uk (Frederic Stahl), atta.badii@reading.ac.uk (Atta Badii)

to concept drift which can be dealt with by re-running the above methods to update the feature rankings in order to accommodate any drifts. However, this can potentially be an expensive procedure especially if there are many dimensions in the data, but it also depends on the user settings of how frequently this feature re-ranking is performed. Hence, the rationale for a single-pass method requiring the re-evaluation of only the features where the classification relevance has changed since the last pass.

This research therefore describes a concept drift detection method for data stream classification algorithms with the feature tracking capability. This enables linking features to concept drifts over a statistical sliding window for feature selection purposes. The method only needs to examine features that have potentially changed their relevance and only when there is an indication that the relevance of a feature may have changed. The proposed method can be used with any learning algorithm either as a real-time wrapper or a batch classifier or realised inside a real-time adaptive classifier [13, 6]. Previous work of the authors has developed a feature tracking technique [14], however, the techniques was not used for feature selection purposes as it suffered from over-fitting on noise and outliers. Thus the contributions of this paper are:

1. A new improved Concept Drift detection technique with Feature Tracking capabilities.
2. A feature selection technique based upon the causality of drifts obtained through the developed Feature Tracking method.

This paper is organised as follows: Section 2 describes related works, Section 3 summarises the MC-NN classifier whose data representation has been used and modified for the real-time feature selection method presented here. Section 4 introduces the drift detection method developed in this research and Section 5 explains the developed feature tracking method. Section 6 takes these developments forward to devise a real-time feature selection approach. Section 7 provides an empirical evaluation of the developed methods and concluding remarks are given in Section 8.

2. Related Work

2.1. Concept Drift Detection Techniques

A concept drift occurs if the pattern encoded in the data stream changes over time. The gathered data changes or shifts, after a stability period. Identifying a drift point as distinct from noise or outlier, is the first and most challenging task for drift detection algorithms [15, 7]. Thus analytics algorithms need to adapt. This issue of concept drift needs to be considered in order to mine relevant data with appropriate accuracy. At least four types of drift can be identified; gradual, sudden, recurring, and incremental, as well as noise and outliers which may occur in the data stream [15, 7, 16, 17].

Outliers and noise are not concept drifts. Often it is very difficult for concept drift detection methods to distinguish noise and outliers from real concept drift.

There exist standalone concept drift detection techniques that can be used in combination with batch learning algorithms and a sliding window approach. For example one of these techniques is CUmulative SUM (CUSUM) [18]. CUSUM raises alarms when the mean μ of the input data is significantly different from zero. CUSUM is considered to be 'almost memoryless', however, it can only be applied in one statistical direction, i.e., detecting only significant increases in feature values.

Another technique is the Drift Detection Method (DDM) [9]. DDM computes error statistics based on two time windows. A concept drift is triggered for sudden concept drifts only, while gradual concept drifts are not detected [19]. Early Drift Detection Method (EDDM) [10] is an extension of DDM, estimating distribution of the distances among classification errors, however, the method is susceptible to noise. The Exponential Weighted Moving Average (EWMA) [20] is also similar to DDM, but the estimate of the error rate is updated faster for each point in the data stream. The ADaptive sliding WINdow (ADWIN) method [8] makes use of a variable size sliding window, whereas the size is dependent on observed changes. If there is a change then the window size decreases, otherwise it increases. A problem with ADWIN is that windows can become potentially very large and thus the time to adapt a classifier may increase.

However, we are not aware of any drift detection method that also provides information as to the causes of the drift, i.e. which features were involved in the genesis of the drift.

2.2. Feature Selection Techniques

Feature selection is used to reduce the dimensions of data streams, aiming to retain only relevant features (i.e. features highly correlated with class labels). This is because learning good classifiers can be achieved by removing irrelevant features [21, 22].

Many techniques have been proposed for dimensionality reduction such as Linear Discriminate Analysis (LDA), Canonical Correlation Analysis (CCA), Multi-View CCA, and Principal Component Analysis (PCA) [11, 12]. LDA transforms two groups of class labels into two matrices to be matched together. CCA is a statistical method which aims to search a linear subspace in which the correlation between two sets of class labels is maximised. Multi-View CCA [11] is an extension of CCA which searches for a pairwise correlation between all sets of class labels. PCA aims to merge the different feature vectors in a low dimensional space of eigenvectors which keeps their direction unchanged when a linear transformation is applied to them. A very popular mechanism to select relevant features for a classification task is based on Information Gain measure which is also used in building decision trees [23]. The assumption is that the higher the Information Gain of an attribute the more likely it is relevant for the classification task.

The aforementioned feature selection techniques are by no means an exhaustive list of techniques. However, they all share a common placing in that they are typically applied before a classifier is induced. Thus these techniques are not designed for data streams as they do not take into consideration that the relevance of a feature for a classification task may change over time. Thus, real-time feature selection techniques are needed.

2.3. Data Stream Classification

Several methods have been proposed for predictive analytics on data streams. The most notable data stream classifier is probably the Hoeffding Tree family of algorithms. The Hoeffding Tree algorithm by Domingos and Hulten [13] introduces a decision tree incrementally in real-time. The Hoeffding Tree was improved in terms of speed and accuracy by proposing a Very Fast Decision Tree (VFDT) [5]. Although achieving high accuracy using a small sample is the main advantage of these algorithms, concept drift detection cannot be handled as a created sub-tree can only expand from the child nodes onwards. A new version of VFDT, termed CVFTD where C stands for Concept Drift [5], was introduced. Broadly speaking, in CVFDT alternative sub-trees can be induced over time and if an alternative sub-tree outperforms (i.e. in terms of accuracy) the current active sub-tree, then the current sub-tree has to be replaced with the alternative one. However, if sub-trees are growing continuously, then a large amount of memory is consumed, also CVFDT is susceptible to noise and unable to handle recurring drifts [24]. Further data stream classification algorithms have been proposed, such as a Similarity Search Structure called the Rank Cover Tree (RCT) [25], Online Data Stream Classification with Limited Labels [26], Prototype-based Classification Model [27], Adaptive Nearest Neighbour Classification for Data Streams [28], Ensemble based Classification [29], VFDR [30], Online Data Stream Classification with limited labels [26], Prototype-based Classification Model [27], DBScan [31], SFNClassifier [32], and G-eRules [6].

Although, most of these algorithms have a built-in concept drift detection capability, none of them takes online feature selection into consideration. If feature selection is applied at all, then it is mostly at the beginning of the data stream and it is assumed that the contribution of each feature to the concept remains invariant over time. However, this is an unrealistic assumption. The relevance of features for the concept may change over time and thus an online feature selection strategy may very well improve the predictive accuracy of the classifier.

2.4. Data Stream Clustering

Clustering is the partitioning and grouping of a particular set of observations according to the similarities of their characteristics. Several Data Stream Cluster Analysis algorithms with minimum time and memory demands have been proposed over the years. These algorithms typically need only one pass through the data in order to adapt to concept drifts. Some notable algorithms here are BIRCH [33], CluStream [34], HPStream [35], E-Stream [36], ClusTree [37], HUE-Stream [38], and MC-NN [39].

A notable development of these cluster analysis algorithms is the aforementioned BIRCH which builds statistical summaries of the clusters and is able to learn incrementally. These statistical summaries are also often referred to as Micro-Clusters. However, BIRCH does not forget concepts and thus its ability to adapt to concept drifts is limited. An extension of BIRCH is the ClusTree algorithm mentioned above, it implements a hierarchical data stream clustering algorithm. Also the aforementioned CluStream algorithm extends the Micro-Clusters with a time component, enabling the algorithm to forget old and obsolete concepts. A more recent extension of CluStream is the MC-NN algorithm. MC-NN is also built on a further extension of the CluStream Micro-Clusters, it adds splitting of Micro-Clusters to adapt to concept drift and uses these Micro-Clusters primarily for parallel predictive analytics in real-time. The research presented in this paper is inspired by the ability of MC-NN Micro-Clusters to adapt to concept drift and has developed a new Micro-Cluster structure. Thus MC-NN will be discussed in more detail in Section 3.

Please note, none of the aforementioned techniques is capable of tracking the causality of a concept drift which could be very useful for real-time feature selection. At the moment feature selection is typically ignored in data-stream mining tasks or simply applied prior to starting a workflow. The remainder of this paper discusses new concept drift detection method that is able to track feature involvement in concept drift. Furthermore a method is then used to facilitate feature selection in real-time.

3. Micro-Cluster Nearest Neighbour (MC-NN)

This section summarises the previously mentioned MC-NN approach [39]. MC-NN was originally developed for predictive data stream analytics, however, the underlying Micro-Cluster structure of MC-NN has been adapted and extended in this research in order to develop a feature tracker for online feature selection purposes. Thus MC-NN is discussed in greater detail. Essentially there are three operations to adapt MC-NN to concept drifts: (1) absorption of data instances into nearest Micro-Clusters, (2) splitting of Micro-Clusters with high variance and (3) removal of obsolete Micro-Clusters.

3.1. The Structure of MC-NN Micro-Clusters

Micro-Clusters in MC-NN provide statistical summaries of feature values over time stamps. In [39], the structure of Micro-Clusters is: $< CF2^x, CF1^x, CF1^t, n, CL, \epsilon, \Theta, \alpha, \Omega >$. Details about the Micro-Cluster structure components are listed in Table 1.

Table 1: The Structure of MC-NN Micro-Clusters.

Structure Component	**Description**
$CF2^x$	a vector with the sum of squares of the features
$CF1^x$	a vector with the sum of feature values
$CF1^t$	a vector with the sum of time stamps
n	the number of data instances in the cluster
CL	the majority class label of the cluster
ϵ	the error count
Θ	the error threshold for splitting the Micro-Cluster
α	the initial time stamp
Ω	a minimum (user defined) threshold for the Micro-Cluster minimum participation

The components listed in Table 1 are used to compute the centroid for a feature within a Micro-Cluster x:

$$centroid(x) = \frac{CF1^x}{n} \tag{1}$$

and the *Variance* of a feature within a Micro-Cluster:

$$Variance[x] = \sqrt{\left(\frac{CF2^x}{n}\right) - \left(\frac{CF1^x}{n}\right)^2} \tag{2}$$

3.2. The Training or Absorbing Instances

Each Micro-Cluster is updated by adding a new instance to its nearest Micro-Cluster's centroid if it is within the boundary (*Variance*) of the Micro-Cluster, (see Figure 1). The error ϵ is decremented by 1 if a new instance matches the CL. Otherwise, if the nearest Micro-Cluster does not match the CL, then the new instance is still added to the nearest Micro-Cluster, however, the Micro-Cluster error ϵ is incremented by one; also the error ϵ of the nearest Micro-Cluster that matches the CL is incremented by 1. In the case that the data instance is outside the boundary (*Variance*) of its nearest Micro-Cluster, then loosely speaking, the instance builds a new Micro-Cluster.

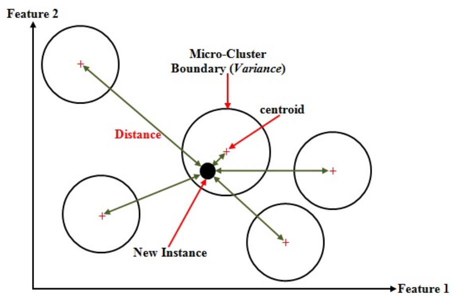


Figure 1: An example of adding a new instance to the nearest Micro-Cluster.

3.3. Splitting of a Micro-Cluster using Variance

MC-NN splits a Micro-Cluster in two new clusters once the error count ϵ reaches Θ and the original Micro-Cluster is removed in order to improve the fit to evolving data streams. The new Micro-Clusters are placed about the original Micro-Cluster feature that has the greatest $Variance$ for a feature x (illustrated in Figure 2). The assumption MC-NN makes here is that the feature with the highest *Variance* is the most likely to contribute to mis-absorption.

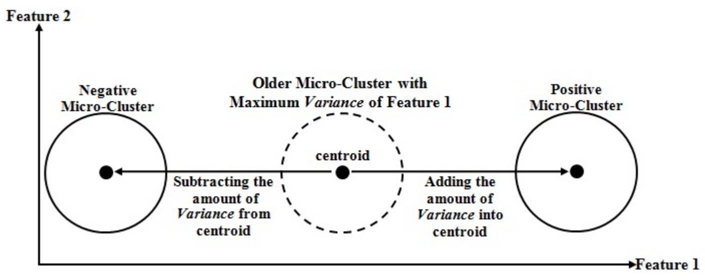

Feature 2
Negative Micro-Cluster
Older Micro-Cluster with Maximum *Variance* of Feature 1
Positive Micro-Cluster
centroid
Subtracting the amount of *Variance* from centroid
Adding the amount of *Variance* into centroid
Feature 1

Figure 2: Splitting of a Micro-Cluster according to the feature with the highest Variance.

3.4. Death and Removal of a Micro-Cluster using Triangle Numbers

MC-NN removes a Micro-Cluster (Micro-Cluster *Death*) if it has not participated recently in absorbing data instances, which can be calculated from $CF1^t$ by measuring the Triangle Number (Equation 3). The Triangle Number gives more weight to recent Micro-Clusters than the older ones. If the participation percentage of a Micro-Cluster is lower than Ω then the Micro-Cluster is removed. The assumption MC-NN makes is that older non-participating Micro-Clusters reflect old and potentially obsolete concepts.

$$Triangle\ Number\ \Delta(T) = ((T^2 + T)/2) \tag{3}$$

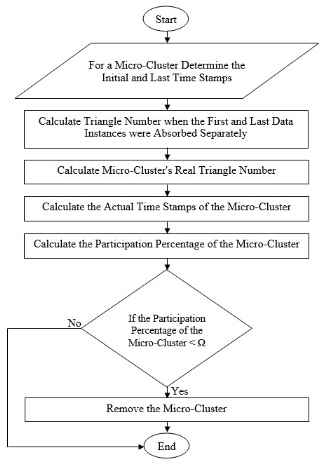

Start
For a Micro-Cluster Determine the Initial and Last Time Stamps
Calculate Triangle Number when the First and Last Data Instances were Absorbed Separately
Calculate Micro-Cluster's Real Triangle Number
Calculate the Actual Time Stamps of the Micro-Cluster
Calculate the Participation Percentage of the Micro-Cluster
If the Participation Percentage of the Micro-Cluster < Ω
No
Yes
Remove the Micro-Cluster
End

Figure 3: The process of calculating the Triangle Number for a Micro-Cluster.

The process of calculating the Triangle Number is given in Figure 3 and highlighted in an example in Figure 4. Consider the Micro-Cluster in the example. It was created at time stamp 2 and updated with instances at time stamps 4 and 6, the current time stamp is 7 but here it was not updated. On the right hand side of Figure 4 we have depicted the calculations for the different steps involved in the process depicted in Figure 3.

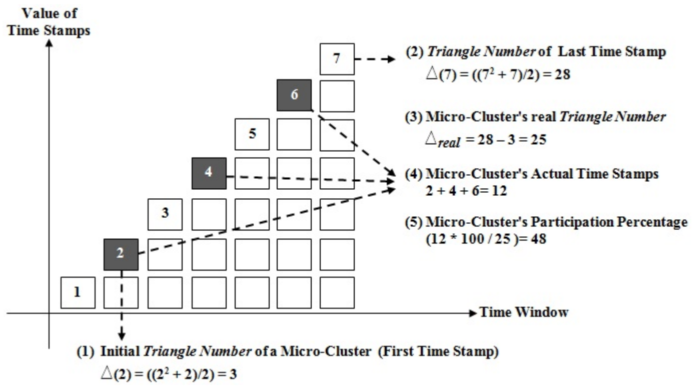

Value of Time Stamps
(2) Triangle Number of Last Time Stamp
$\triangle(7) = ((7^2 + 7)/2) = 28$
(3) Micro-Cluster's real Triangle Number
$\triangle_{real} = 28 - 3 = 25$
(4) Micro-Cluster's Actual Time Stamps
2 + 4 + 6 = 12
(5) Micro-Cluster's Participation Percentage
(12 * 100 / 25) = 48
Time Window
(1) Initial Triangle Number of a Micro-Cluster (First Time Stamp)
$\triangle(2) = ((2^2 + 2)/2) = 3$

Figure 4: An Example of Triangle Number calculation. The shaded areas signify the time stamps where the Micro-Cluster has participated in absorbing new instances for a particular time stamp

3.5. *Taking MC-NN Forward to Develop a Real-time Feature Selection Method*

Sections 4, 5 and 6 take some of MC-NN's ideas forward to facilitate real-time feature selection. The basic idea is to monitor Micro-Cluster *Split rates* and *Death rates* in order to detect concept drifts. It is expected that concept drifts will cause a peak in splitting and removing of Micro-Clusters. The *Velocity* and direction of movement of the Micro-Clusters can be used as an indication as to which features have changed and potentially contributed towards a concept drift. Then, in turn, features that have shown significant changes during a concept drift can be examined separately for their relevance to the classification task.

For clarity we note the semantic distinctions in our usage of three terms relating to time and temporal referencing of a point or an interval in the timeline, namely 'time', 'time window', 'time stamp'. In our analysis 'time' and 'time window' are regarded as equivalent references used interchangeably to mean a duration of time as encapsulating a set of data instances with sequential time intervals of fixed length. Whereas 'time stamp' is a means of indexing time, i.e. provides a mark-up for, a particular data instance at a particular point in time. For example assuming a stream has generated 2000 sequential data instances and each time window is of length 1000, then there would be two time windows, time window 1 from time stamp 0 to 999 and time window 2 from time stamp 1000 to 1999. This may be for example referred to in the paper as time 1 or time 2 meaning time window 1 and 2.

4. Real-Time Concept Drift Detection Technique using Adaptive Micro-Clusters

The MC-NN algorithm aims to keep a recent and accurate summary of the data stream using Micro-Clusters. Significant changes to these summaries are used in this research to detect concept drift.

4.1. *Detecting Concept Drift using Adaptive Micro-Clusters*

It is expected that during a concept drift the data distribution changes (causing larger *Variance* within Micro-Clusters), but also it is expected that Micro-Clusters absorb more incorrect data instances (different class labels than the Micro-Cluster). When new concepts arrive, MC-NN adapts by applying *Split* (Micro-Cluster Splitting) and *Death* (Micro-Cluster removal), as explained in Section 3.

Loosely speaking, regarding *Split*, two new Micro-Clusters are generated from the original one when the error counter reaches the error threshold due to false positive participation. Likewise some of the existing Micro-Clusters may stop to participate as they become obsolete and are removed. Both will result in a higher number of *Splits* and *Deaths* over time. In this research we measure *Split* and *Death* rates over time simply by using Equation 4 for each time window. Here n is the number of instances in a time window and i is the index of the current data instance within the time window, starting with 1 within each time window. Figure 5 illustrates an example of monitoring a *Split* or *Death* rate. In this sense the calculation of *Split* and *Death* is conducted gradually instance by instance within a time window.

$$\frac{\sum_{i=1}^{n}(Number\ of\ Splits\ or\ Deaths_i - Number\ of\ Splits\ or\ Deaths_{i-1})}{n} \quad (4)$$

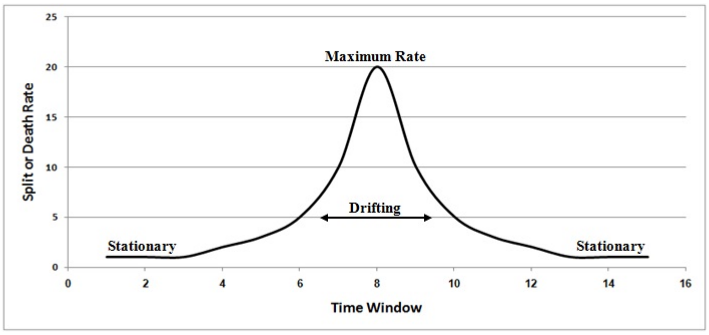


Figure 5: An example of Micro-Cluster Split and Death Rate.

The assumption is the larger both rates are, the more likely it is a concept drift has happened. Using a windowing approach on the data stream, a running average of the *Split* and *Death* rates are calculated separately, as well as the *Percentage Difference* which is given by Equation 5. In the equation i denotes the current time window.

$$Percentage\ Difference = \frac{|Rate_i - Rate_{i-1}|}{(Rate_i + Rate_{i-1})/2} * 100 \quad (5)$$

If the *Percentage Differences* of both, the *Split* and *Death* rates differ from the μ (of the current and previous window) by 50% (default value), then this is considered as a concept drift. In this work the default value of 50% has been used for all experiments as it yielded accurate results in most cases. However, the user may change this to a different threshold. This is illustrated in Algorithm 1. A worked example is presented in Section 4.2.

Algorithm 1 Detection of Drifts

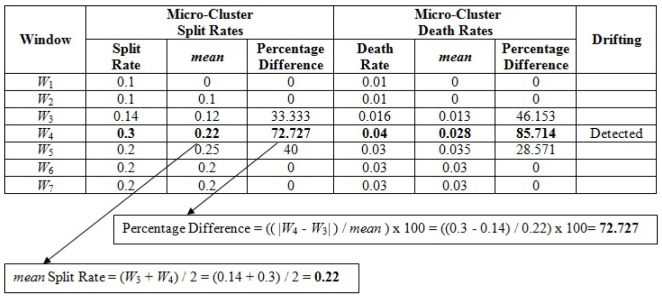
Input: Micro-Cluster(s) *Split* and *Death* rates for current and previous window

Output: Detection of Drifts

```
1: Calculate the $\mu$ of Split and Death rates over the current and previous window
2: Calculate the Percentage Difference for both, Split and Death rates
3: if new Split rate $> \mu$ (Split rate) AND new Death rate $> \mu$ (Death rate) AND Percentage Difference (Split rate) $> 50\%$ AND Percentage Difference (Death rate) $> 50\%$ then
4:     Concept Drift
5:     reset MC(s)
6: else
7:     No Concept Drift
8: end if
```

4.2. Worked Example

Split and *Death* rates of Micro-Clusters are as shown below in Figure 6 for the seven time windows.



Window	Micro-Cluster Split Rates			Micro-Cluster Death Rates			Drifting
	Split Rate	*mean*	Percentage Difference	Death Rate	*mean*	Percentage Difference	
W_1	0.1	0	0	0.01	0	0	
W_2	0.1	0.1	0	0.01	0	0	
W_3	0.14	0.12	33.333	0.016	0.013	46.153	
W_4	**0.3**	**0.22**	**72.727**	**0.04**	**0.028**	**85.714**	Detected
W_5	0.2	0.25	40	0.03	0.035	28.571	
W_6	0.2	0.2	0	0.03	0.03	0	
W_7	0.2	0.2	0	0.03	0.03	0	

Percentage Difference = (($|W_4$ - $W_3|$) / *mean*) x 100 = ((0.3 - 0.14) / 0.22) x 100= **72.727**

mean Split Rate = (W_3 + W_4) / 2 = (0.14 + 0.3) / 2 = **0.22**

Figure 6: An example of Concept Drift Detection using the Micro-Clusters Split and Death Rates.

Consider the *Split* and *Death* rates of window (W_4), which are equal to 0.3 and 0.04. This shows that the *Split* and *Death* rates differ from the μ value (i.e., 0.22 and 0.028) by 50% as the *Percentage Difference* of *Split* rate equals to 72.727, and the *Percentage Difference* of *Death* rate equals to 85.714. This indicates that a drift has occurred. After a concept drift is detected our method identifies which features have been involved in the drift by examining the Micro-Clusters change of *Velocity*, trajectory and spread of data such as *Variance* or Inter Quartile Range (IQR). This is described in Section 5.

5. Real-Time Feature Tracking Technique using Adaptive Micro-Clusters

The *Velocity* or *Variance* of a feature can be derived from MC-NN Micro-Clusters and are calculated once a drift is detected as described in Section 4. The *Velocity* and *Variance* can then be analysed to identify features that have been involved in the drift. This information can be used for feature selection purposes which will be explained in Section 6. The tracking of features is potentially influenced by feature-bias, outlier and noise. Thus our method incorporates approaches to counter these influences. The remainder of this section discusses in Section 5.1 how the proposed method addresses potential feature-bias, in Section 5.2 it is explained how the method addresses the problem of noise, Section 5.3 then explains how the features are tracked and outliers are taken into consideration. The proposed method uses a different means of measuring participation of Micro-Clusters than original MC-NN, (explained in Section 5.3), which is compared qualitatively in Section 5.3.6.

5.1. Minimising the Effect of Feature-Bias using Real-Time Normalisation

Normalisation is applied to fit the data (i.e., each feature of a new training instance) to be almost distributed in a pre-defined boundary such as [0,100]. *Normalisation* is used to avoid feature-bias which can potentially lead to mis-classifications as the relevant relations between target class labels and features are considered by the classifier to be more or less important than they actually are.

Three frequently used types of *Normalisation* are i.e. Min-Max (see Equation 6), Decimal Scaling (see Equation 7), and Z-Score (see Equation 8) [40]. These have been considered for inclusion in the proposed method.

$$x = \left(\frac{current\ value - \min x}{\max x - \min x}\right) * (\max range - \min range) + \min range \quad (6)$$

$$x = \frac{x}{(10^d)} \quad (7)$$

$$x = \frac{x - \mu}{\sigma} \quad (8)$$

x is a feature value of the new data instance. Min-Max is a simple *Normalisation* technique to fit the data in a pre-defined boundary with a min *range* (default 0) and a max *range* (default 100). Decimal Scaling moves the decimal point of a feature value x depending on its maximum absolute value whereby d equals to $MAX(|\ y\ |) < 1$ (i.e., the smallest value of a feature y). Lastly, Z-Score normalises a feature value x according to its corresponding μ and σ feature values.

In this research, Min-Max *Normalisation* technique was used as minimum and maximum values of a feature x can easily be reinitialised in real-time when new instances arrive. Equation 6 is applied for every new data instance as shown in Figure 7. The alternative techniques mentioned above rely on the standard deviation and the mean and thus require the buffering of data before *Normalisation* can be applied. This is undesirable in real-time data analytics. However, the *Normalisation* process described

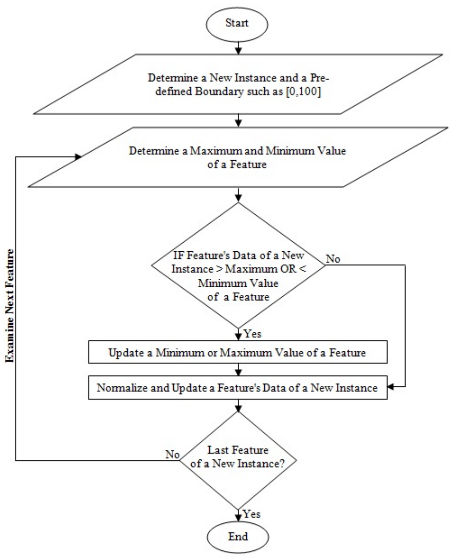
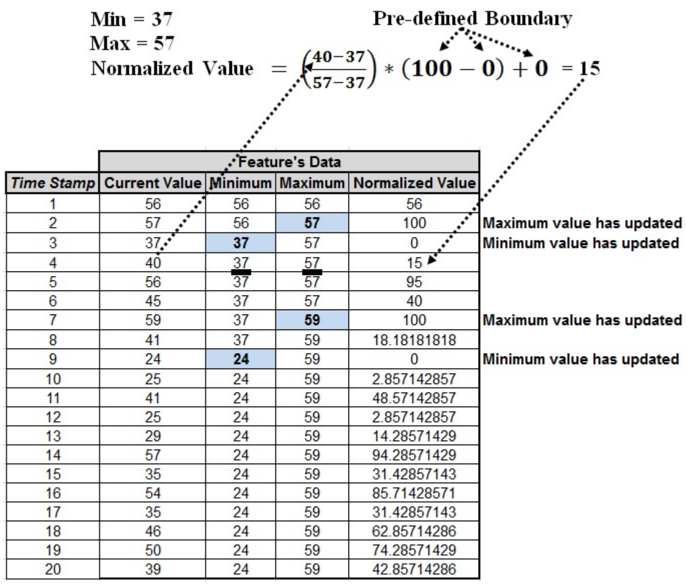

Start
Determine a New Instance and a Pre-defined Boundary such as [0,100]
Determine a Maximum and Minimum Value of a Feature
IF Feature's Data of a New Instance > Maximum OR < Minimum Value of a Feature
No
Yes
Update a Minimum or Maximum Value of a Feature
Normalize and Update a Feature's Data of a New Instance
Last Feature of a New Instance?
No
Yes
End
Examine Next Feature

Figure 7: Flowchart of Min-Max Normalisation.



Min = 37
Max = 57
Normalized Value $= \left(\frac{40-37}{57-37}\right) * (100-0) + 0 = 15$
Pre-defined Boundary

	Feature's Data				
Time Stamp	Current Value	Minimum	Maximum	Normalized Value	
1	56	56	56	56	
2	57	56	**57**	100	Maximum value has updated
3	37	**37**	57	0	Minimum value has updated
4	40	37	57	15	
5	56	37	57	95	
6	45	37	57	40	
7	59	37	**59**	100	Maximum value has updated
8	41	37	59	18.18181818	
9	24	**24**	59	0	Minimum value has updated
10	25	24	59	2.857142857	
11	41	24	59	48.57142857	
12	25	24	59	2.857142857	
13	29	24	59	14.28571429	
14	57	24	59	94.28571429	
15	35	24	59	31.42857143	
16	54	24	59	85.71428571	
17	35	24	59	31.42857143	
18	46	24	59	62.85714286	
19	50	24	59	74.28571429	
20	39	24	59	42.85714286	

Figure 8: An Example of Min-Max Normalisation in real-time.

here, Min-Max *Normalisation*, can be updated incrementally instance by instance. Old instances cannot be re-normalised as the original data values are not buffered but absorbed in the statistics of a Micro-Cluster. This could lead to Micro-Clusters not fitting the current concept well anymore. However, this would also contribute to the detection of the concept drift (Micro-Cluster *Split*) as the *Variance* of the Micro-Cluster would increase and potentially also the error count. Figure 8 shows an example of Min-Max *Normalisation* in real-time.

Next the Micro-Cluster by which the normalised data instance should be absorbed has to be determined. For this the training task of MC-NN described in Section 3.1 is applied. However, before the data instance is absorbed a Low Pass Filter (*LPF*) is applied to the Micro-Cluster to minimise the effect of noise.

5.2. Minimising the Effect of Noise using Low Pass Filter (LPF)

LPF is a filter that passes signals (i.e., the *Velocity* of a feature) with a frequency lower than a certain cut-off frequency α and attenuates signals with frequencies higher than the cut-off frequency. *LPF* can be used to minimise the effect of noise [41, 42]. However, there are some other techniques proposed for the purpose of filtering such as the Kalman filter and Grid-based filter which require buffering of data before filtering [43]. Hence, in this work, *LPF* is used as it can be calculated over a sliding window without buffering of data. Each normalised feature of a new training instance was filtered by *LPF* together with its nearest Micro-Cluster using Equation 9 in order to improve concept drift detection and feature tracking. This is illustrated in Figure 9 showing a flowchart of *LPF*.

$$new\ filter[x] = \alpha * new\ value[x] + (1-\alpha) * old\ filter[x] \quad (9)$$

The α threshold is set for 0.5 by default, as this value yielded good results in most cases. However, the user may change this to a different threshold.

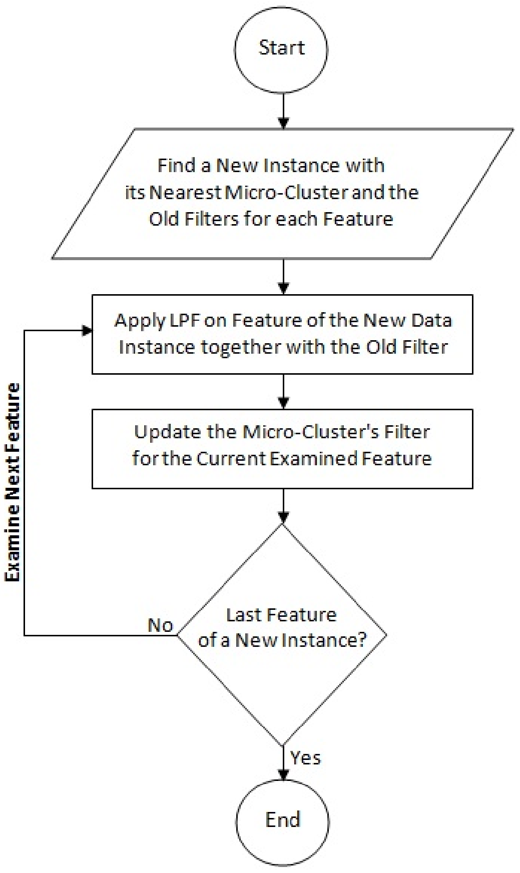
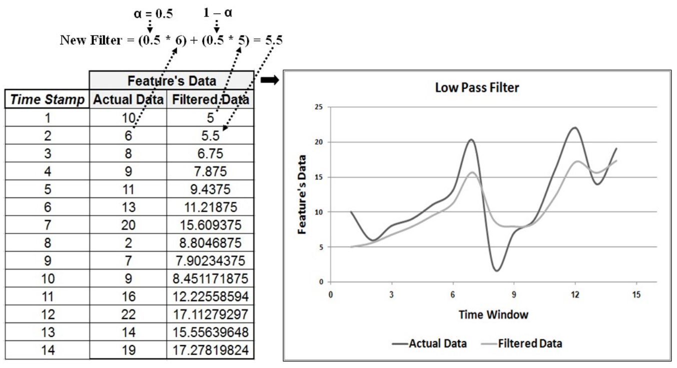

Start
Find a New Instance with its Nearest Micro-Cluster and the Old Filters for each Feature
Apply LPF on Feature of the New Data Instance together with the Old Filter
Update the Micro-Cluster's Filter for the Current Examined Feature
Last Feature of a New Instance?
No
Yes
Examine Next Feature
End

Figure 9: Flowchart of Low Pass Filter.

Figure 10 shows an example of *LPF*. The centroid of each filtered feature of a Micro-Cluster can be calculated. Micro-Clusters with normalised and filtered features are passed to the concept drift detection method which was described in Section 4.



	Feature's Data	
Time Stamp	Actual Data	Filtered Data
1	10	5
2	6	5.5
3	8	6.75
4	9	7.875
5	11	9.4375
6	13	11.21875
7	20	15.609375
8	2	8.8046875
9	7	7.90234375
10	9	8.451171875
11	16	12.22558594
12	22	17.11279297
13	14	15.55639648
14	19	17.27819824

Figure 10: An Example of Low Pass Filter.

5.3. Real-Time Feature Tracking

This section describes real-time feature tracking measuring the *Velocity* and spread of a feature using both *Variance* and Inter Quartile Range (*IQR*) which can be derived from the captured statistics of the Micro-Clusters.

5.3.1. Velocity of Features

Velocity can be tracked through an extension of the MC-NN Micro-Cluster structure by: $< CF1^{hx}, n_h >$. Where these components are equivalent to $CF1^x$ and n. However, the h denotes that these components are *historical* summaries (taken from the previous time stamp). The *Velocity* of a feature x can then be calculated using Equation 10. Figure 11 shows an example of feature *Velocity* with a Micro-Cluster consisting of two features.

$$Velocity[x] = \left| \frac{CF1^x}{n} - \frac{CF1^{h,x}}{n_h} \right| \tag{10}$$

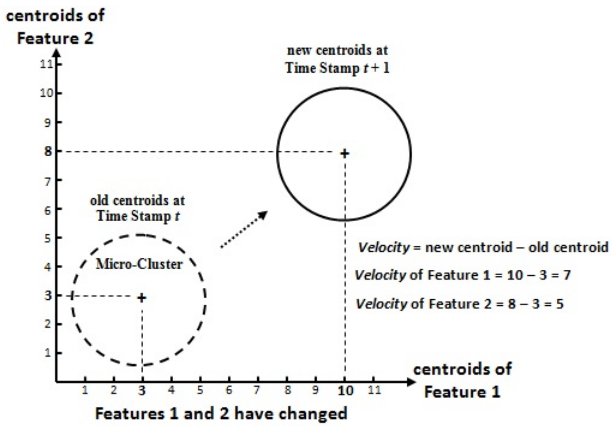

centroids of Feature 2
new centroids at Time Stamp $t+1$
old centroids at Time Stamp t
Micro-Cluster
Velocity = new centroid – old centroid
Velocity of Feature 1 = 10 – 3 = 7
Velocity of Feature 2 = 8 – 3 = 5
centroids of Feature 1
Features 1 and 2 have changed

Figure 11: An Example of Feature Velocity.

5.3.2. Measuring of the Data Stream Spread using IQR

In original MC-NN the spread of the values absorbed in a Micro-Cluster is measured using *Variance*. The spread quantifies how varied the set of a feature's values are. However, the performance of *Variance* can be affected negatively by outliers as they may significantly change μ. Whereas, *IQR* is considered a more robust method with respect to outliers [44, 45] compared with *Variance*. The first version of the here presented feature tracker also adopted *Variance* as a measure of the spread of a feature. However, as expected it was observed that this approach did not work very well due to its sensitivity to outliers. Thus *IQR* was explored as an alternative to *Variance* in order to obtain a more reliable measure of the Micro-Cluster spread. The evaluation results presented in Section 7.2 compare the performance of the proposed feature tracker as obtained using either *Variance* or *IQR*.

This section describes the new extension of the MC-NN Micro-Cluster structure with *IQR* to split a Micro-Cluster. *IQR* measures the spread of an ordered set of data (i.e., ascending order) by dividing it into quartiles such as Lower Quartile ($Q1$), *median* , and Upper Quartile ($Q3$). $Q1$ and $Q3$ are the middle numbers of the first and second half of an ordered list of feature values, respectively; *median* is the middle number of an ordered list of data values. *IQR* is the difference between $Q3$ and $Q1$. This is applied for the purpose of splitting a Micro-Cluster which is explained in the next section.

5.3.3. Splitting of a Micro-Cluster using IQR

One purpose of using *IQR* in this research is to *Split* a Micro-Cluster once its error count ϵ reaches Θ. The use of ϵ and Θ was described in Section 3. This research prefers *IQR* over *Variance* as it is more robust to outliers. The assumption here is that the larger the IQR of a feature, the greater the range of values that have been seen for that feature and thus it may contribute to miss-absorption of new data instances. The new Mirco-Clusters resulting from the *Split* are generated with *Q1* and *Q3* quartiles. The centroids of the new Micro-Clusters are replaced with either *Q1* or *Q3* in all dimensions (features). In particular one Micro-Cluster only uses *Q1* values and the other only uses *Q3* values. This is shown for one dimension only in Figure 12. The original Micro-Cluster is deleted after the *Split* is performed.

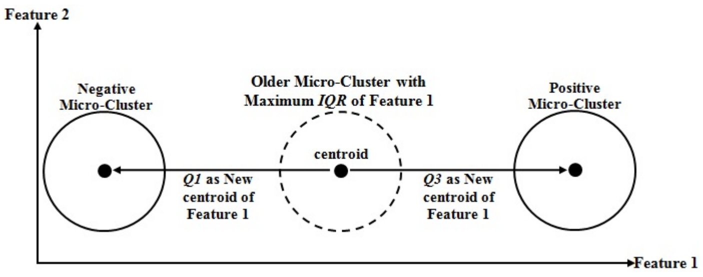

Feature 2
Negative Micro-Cluster
Older Micro-Cluster with Maximum IQR of Feature 1
Positive Micro-Cluster
centroid
Q1 as New centroid of Feature 1
Q3 as New centroid of Feature 1
Feature 1

Figure 12: Splitting of a Micro-Cluster with IQR.

Results presented in Section 7.2 compare a version of our approach only using *Variance* and one using the proposed *IQR*. The next section provides a worked example on Micro-Cluster splitting for both using *Variance* and *IQR*.

5.3.4. Comparison of Splitting using IQR or Variance

This section compares splitting of a Micro-Cluster containing an outlier using IQR and $Variance$ to demonstrate that *IQR* is a more robust metric to *Split* Micro-Clusters if there are outliers. For simplicity of this illustration the *Split* along only one feature is observed.
Consider a feature's values $[x] = 1, 1, 1.2, 2.5, 2.8, 3, 4.6, 5, 3000$, and consider the value 3000 to be an outlier. Then $centroid[x]$ $= \mu = (1 + 1 + 1.2 + 2.5 + 2.8 + 3 + 4.6 + 5 + 3000)/9 =$ 335.6778.

- ***Splitting of*** x ***with*** IQR**:**
  Median $= 2.8$,
  $Q1[x] = (1 + 1.2)/2 = 1.1$,
  $Q3[x] = (4.6 + 5)/2 = 4.8$,
  $IQR[x] = Q3 - Q1 = 3.7$,
  $centroid[x]$ of a Negative Micro-Cluster $= Q1[x] =$ 1.1,
  $centroid[x]$ of a Positive Micro-Cluster $= Q3[x] =$ 4.8

- ***Splitting of*** x ***with*** $Variance$**:**
  $\left(\frac{CF2^x}{n}\right) = ((1*1) + (1*1) + (1.2*1.2) + (2.5*2.5) + (2.8*2.8) + (3*3) + (4.6*4.6) + (5*5) + (3000*3000))/9$ $= 1000008.077$,
  $\left(\frac{CF1^x}{n}\right)^2 = ((1 + 1 + 1.2 + 2.5 + 2.8 + 3 + 4.6 + 5 + 3000)/9)^2 = 112679.5705$,
  $Variance[x] = \sqrt{\left(\frac{CF2^x}{n}\right) - \left(\frac{CF1^x}{n}\right)^2} = 941.9815$,
  $centroid[x]$ of a Negative Micro-Cluster = old $centroid[x]$ - $Variance[x] = 335.6778 - 941.9815 = -606.3037$,
  $centroid[x]$ of a Positive Micro-Cluster = old $centroid[x]$ $+ Variance[x] = 335.6778 + 941.9815 = 1277.6593$.

In the example above it can be seen that the original Micro-Cluster centroid is strongly influenced by the outlier. However, it can also be seen that the new centroids after splitting using *IQR* are within the value range of the feature's values (excluding the outlier), whereas the new centroids after splitting using Variance, as it is performed in original MC-NN, are shifted towards the outlier outside the original Micro-Cluster's value range. Thus, in this research *IQR* is used to perform Micro-Cluster splits to mediate the influence of outliers.

To use the *IQR*, the feature data values need to be sorted in an ascending order. To achieve this in a computationally efficient manner, First-In-First-Out ($FIFO$) principle combined with *SkipList* is used.

5.3.5. First-In-First-Out (FIFO) with SkipList

Firstly, in order to save the feature data values a First-In-First-Out ($FIFO$) queue is applied where the oldest entry of the queue is handled first. $FIFO$'s size can be set by a user threshold which is less or equal to the size of the statistical windows (time windows). $FIFO$ keeps the most recent data (i.e., feature's data) and needs to be set to a lowest possible size (i.e., less or equal to a window's size) which yields good results. In an initial unpublished feasibility study it was found that a queue size of 1000 works well in most cases in terms of computational efficiency and accuracy. Thus a queue of size 1000 has been used in all experiments presented in this paper. It was also observed in most cases that even if the statistical window size is much larger than 1000, that it is unlikely that more than 1000 instances are absorbed by a single Micro-Cluster. Thus the potential loss of information due to a queue size limit is also unlikely.

Second, for sorting the $FIFO$ queue of a feature x, a *SkipList* is used, which is a data structure that allows fast search and insert $O(\log n)$ rather than $O(n)$ operations by updating a linked hierarchy of sub-sequences within an ordered sequence of elements [46, 47]. *SkipList* is superior to alternative sorting algorithms in terms of computational efficiency as indicated in [48]. Each node in a *SkipList* consists of four directions (up, down, left, and right), as shown in Figure 13. Skipping over fewer nodes than the previous one with each consecutive sub-sequence is the main structure of a *SkipList*. Once splitting of a Micro-Cluster with larger or maximum IQR is applied, the $FIFO$ queue of each feature x of a Micro-Cluster is then sent to a *SkipList*.

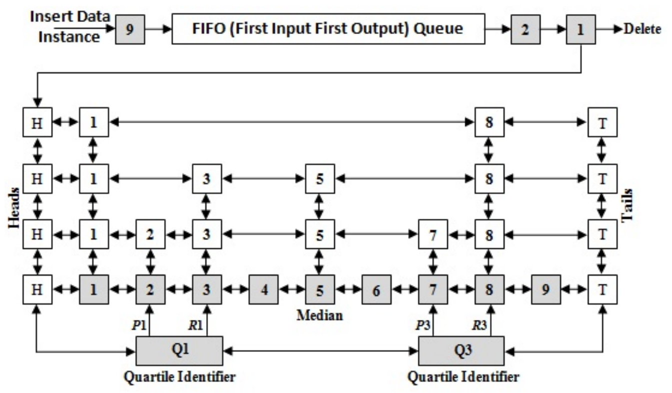


Figure 13: Real-Time sorting of the feature data values using a *SkipList*.

In order to locate and update the position of $Q1$ and $Q3$, *Quartile Identifiers* have been created, which are a data structure connected with head and tail of a *SkipList* as illustrated in Figure 13. A *Quartile Identifier* consists of two nodes P and R, P is a pointer to either the value or the value next to the left of the quartile $Q1$ or $Q3$, and R is a pointer to the next value right of P. The index of P and R is either 1 or 3, which denotes if the pointers refer to Q1 or Q3 respectively. The *Quartile Identifier* skips either to the left or right position after each insertion of a new *SkipList* node, as shown in Algorithms 2, 3, and 4. *Quartile Identifier* of $Q1$ is located in the middle of the left half of *median* of the *SkipList* and connected with the head of the *SkipList* indicated with letter H in Figure 13. While the *Quartile Identifier* of $Q3$ is located in the middle of the right half and connected with tail of the *SkipList* indicated with letter T in Figure 13.

Given P and R of a *Quartile Identifier* and if the number of feature values left/right of the median is either even or odd, then quartiles $Q1$ and $Q3$ can be calculated using the following equations, where y refers to either $Q1$ and $Q3$. If the number of feature values left/right of the median is even then Equation 11 is used and if it is odd then Equation 12 is used.

$$Q_y[x] = (P_y + R_y)/2 \tag{11}$$

$$Q_y[x] = P_y \tag{12}$$

When adjusting the *Quartile Identifiers* in real-time there are three cases to consider depending on whether the number of values of the feature is odd or even and also if the number of values to the left and right of the median is odd or even. For each of the three cases three further scenarios exist depending where the value is inserted in the *SkipList*. These cases and scenarios are described below including examples.

Case 1: *After inserting the new value, the total number of feature values in the SkipList is even and the number of values left or right of the median is odd. In this particular case Algorithm 2 is applied.*

Algorithm 2 Adjusting *Quartile Identifiers* for lower and upper quartiles ($Q1$ and $Q3$) in terms of **Case 1**

Input: new feature value f, nodes of a *SkipList* after adding a new feature value f, previous *Quartile Identifiers* $P1$ and $P3$.
Output: updated *Quartile Identifiers* $P1$ and $P3$

```
1:  if $f \leq value\ of\ P1$ then
2:      $P1$ and $P3$ remain at the same position
3:  else
4:      if $f > value\ of\ P1$ and $f \leq value\ of\ P3$ then
5:          Skip $P1$ to the next value right of $P1$
6:      else
7:          if $f > value\ of\ P3$ then
8:              Skip $P1$ to the next value right of $P1$
9:              Skip $P3$ to the next value right of $P3$
10:         end if
11:     end if
12: end if
```

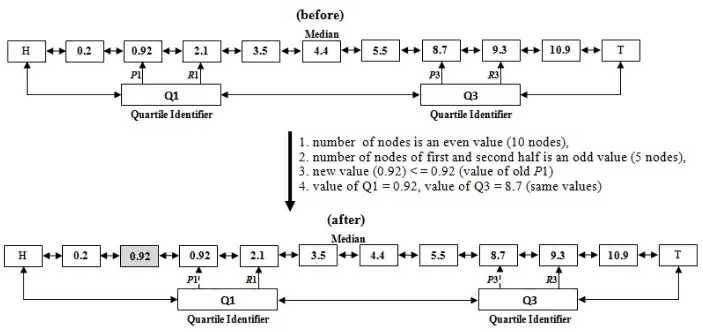
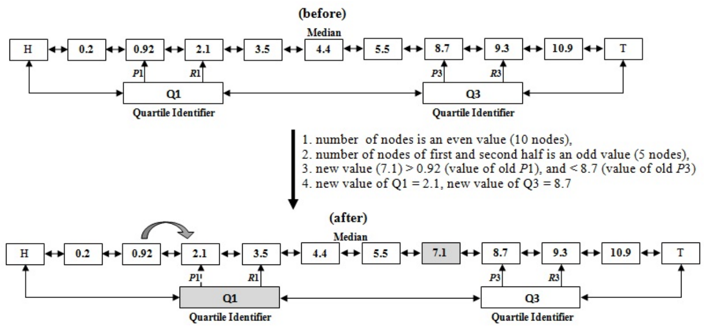

(before)
Median
H
0.2
0.92
2.1
3.5
4.4
5.5
8.7
9.3
10.9
T
P1
R1
P3
R3
Q1
Q3
Quartile Identifier
Quartile Identifier
1. number of nodes is an even value (10 nodes),
2. number of nodes of first and second half is an odd value (5 nodes),
3. new value (0.92) <= 0.92 (value of old P1)
4. value of Q1 = 0.92, value of Q3 = 8.7 (same values)
(after)
Median
H
0.2
0.92
0.92
2.1
3.5
4.4
5.5
8.7
9.3
10.9
T
P1
R1
P3
R3
Q1
Q3
Quartile Identifier
Quartile Identifier

Figure 14: An example of *Quartile Identifiers* update for Case 1 with value inserted left of Q1.

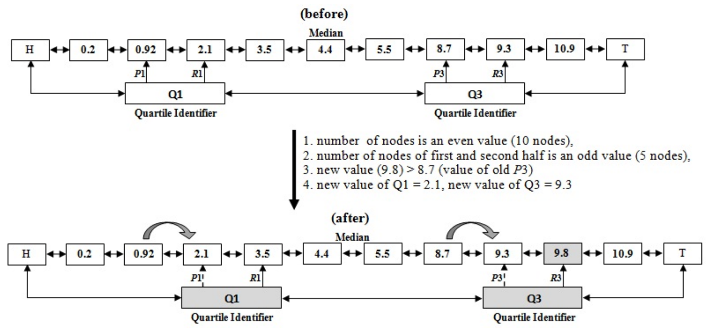
In Figure 14, the number of nodes of the *SkipList* for the feature is an even value (10 nodes) after adding the new feature value 0.92, and the number of nodes of the first and second half of a *SkipList* is an odd value (5 nodes). The new value $f \leq value\ of\ P1$, thus lines 1 and 2 in Algorithm 2 are executed and $Q1$ and $Q3$ are computed, in this scenario $P1$ and $P3$ remain unchanged. The Quartiles are computed using Equation 12.



1. number of nodes is an even value (10 nodes),
2. number of nodes of first and second half is an odd value (5 nodes),
3. new value (7.1) > 0.92 (value of old P1), and < 8.7 (value of old P3)
4. new value of Q1 = 2.1, new value of Q3 = 8.7

Figure 15: An example of *Quartile Identifiers* update for Case 1 with value inserted between of Q1 and Q3.

In Figure 15, the number of nodes of the *SkipList* for the feature is an even value (10 nodes) after adding the new feature value 7.1, and the number of nodes of the first and second half of a *SkipList* is an odd value (5 nodes). The new value $f > value\ of\ P1\ and\ f \leq value\ of\ P3$, thus lines 4 and 5 in Algorithm 2 are executed and $Q1$ and $Q3$ are computed. In this case $P1$ is skipped to the next value right of $P1$ and $P3$ remains unchanged. The Quartiles are computed using Equation 12.



1. number of nodes is an even value (10 nodes),
2. number of nodes of first and second half is an odd value (5 nodes),
3. new value (9.8) > 8.7 (value of old P3)
4. new value of Q1 = 2.1, new value of Q3 = 9.3

Figure 16: An example of *Quartile Identifiers* update for Case 1 with value inserted right of Q3.

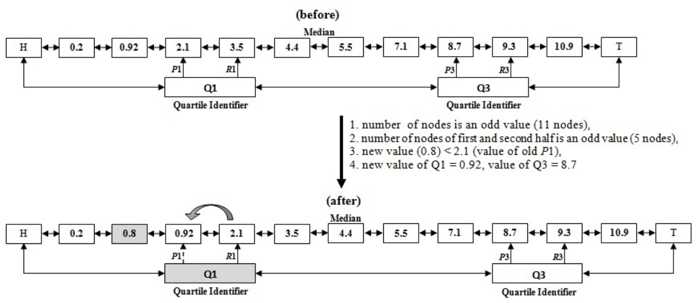
In Figure 16, the number of nodes of the *SkipList* for the feature is an even value (10 nodes) after adding the new feature value 9.8, and the number of nodes of the first and second half of a *SkipList* is an odd value (5 nodes). The new value $f > value\ of\ P3$, thus lines 7 to 10 in Algorithm 2 are executed and $Q1$ and $Q3$ are computed. In this case $P1$ and $P3$ are skipped to the next value right of $P1$ and $P3$ respectively. The Quartiles are computed using Equation 12.

Case 2: *After inserting the new value, the total number of feature values in the SkipList is an odd value. In this particular case Algorithm 3 is applied.*

Algorithm 3 Adjusting *Quartile Identifiers* for lower and upper quartiles ($Q1$ and $Q3$) in terms of **Case 2**

Input: new feature value f, nodes of a *SkipList* after adding a new feature's value f, previous *Quartile Identifiers* $P1$ and $P3$.
Output: updated *Quartile Identifiers* $P1$ and $P3$

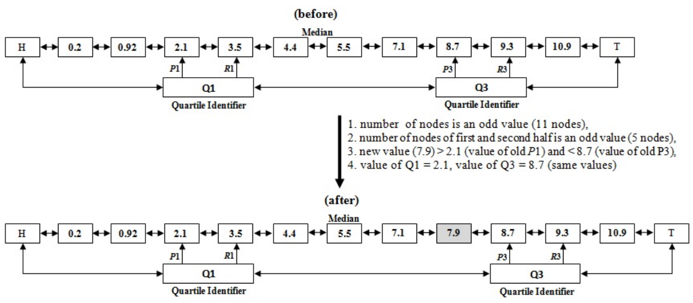
```
1:  if $f \leq value\ of\ P1$ then
2:      Skip $P1$ to the next value left of $P1$
3:  else
4:      if $f > value\ of\ P1\ and\ f \leq value\ of\ P3$ then
5:          $P1$ and $P3$ remain at the same position
6:      else
7:          if $f > value\ of\ P3$ then
8:              Skip $P3$ to the next value right of $P3$
9:          end if
10:     end if
11: end if
```



1. number of nodes is an odd value (11 nodes),
2. number of nodes of first and second half is an odd value (5 nodes),
3. new value (0.8) < 2.1 (value of old P1),
4. new value of Q1 = 0.92, value of Q3 = 8.7

Figure 17: An example of *Quartile Identifiers* update for Case 2 with value inserted left of Q1.

In Figure 17, the number of nodes of the *SkipList* for the feature is an odd value (11 nodes) after adding the new feature value 0.8. The new value $f \leq value\ of\ P1$, thus lines 1 and 2 in Algorithm 3 are executed and $Q1$ and $Q3$ are computed. In this case $P1$ is skipped to the next value left of $P1$ and $P3$ remains unchanged. The Quartiles are computed using Equation 12.

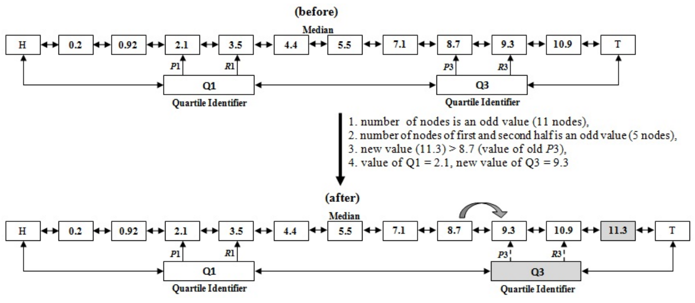


1. number of nodes is an odd value (11 nodes),
2. number of nodes of first and second half is an odd value (5 nodes),
3. new value (7.9) > 2.1 (value of old P1) and < 8.7 (value of old P3),
4. value of Q1 = 2.1, value of Q3 = 8.7 (same values)

Figure 18: An example of the update for the *Quartile Identifiers* for Case 2 with a value inserted between Q1 and Q3.

In Figure 18, the number of nodes of the *SkipList* for the feature is an odd value (11 nodes) after adding the new feature value of 7.9. The new value $f > value\ of\ P1\ and\ f \leq$

value of $P3$, thus lines 4 and 5 in Algorithm 3 are executed and $Q1$ and $Q3$ are computed. In this case $P1$ and $P3$ remains unchanged. The Quartiles are computed using Equation 12.



(before)
Median
Quartile Identifier
1. number of nodes is an odd value (11 nodes),
2. number of nodes of first and second half is an odd value (5 nodes),
3. new value (11.3) > 8.7 (value of old P3),
4. value of Q1 = 2.1, new value of Q3 = 9.3
(after)

Figure 19: An example of *Quartile Identifiers* update for Case 2 with value inserted right of Q3.

In Figure 19, the number of nodes of the *SkipList* for the feature is an odd value (11 nodes) after adding the new feature value 11.3. The new value $f > value\ of\ P3$, thus lines 7 to 9 in Algorithm 3 are executed and $Q1$ and $Q3$ are computed. In this case $P3$ is skipped to the next value right of $P3$ and $P1$ remains unchanged. The Quartiles are computed using Equation 12.

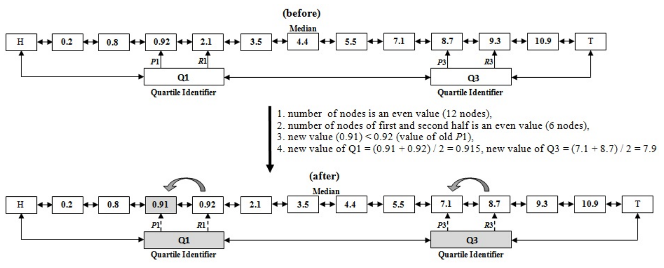
Case 3: *After inserting the new value, the total number of feature values in the SkipList is even and the number of values left or right of the median is even. In this particular case Algorithm 4 is applied.*

Algorithm 4 Adjusting *Quartile identifiers* for lower and upper quartiles ($Q1$ and $Q3$) in terms of **Case 3**

Input: new feature value f, nodes of a *SkipList* after adding a new feature value f, previous *Quartile Identifiers* $P1$ and $P3$.

Output: updated *Quartile Identifiers* $P1$ and $P3$

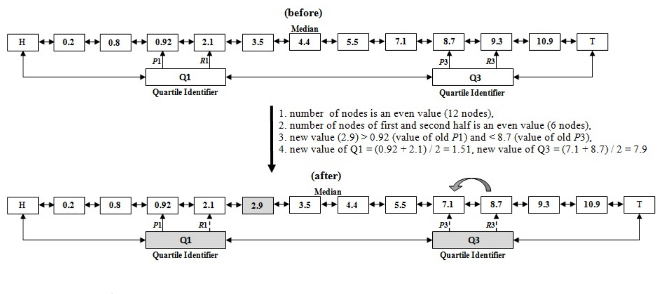
```
1: if $f \leq value\ of\ P1$ then
2:     Skip $P1$ to the next value left of $P1$
3:     Skip $P3$ to the next value left of $P3$
4: else
5:     if $f > value\ of\ P1$ and $f \leq value\ of\ P3$ then
6:         Skip $P3$ to the next value left of $P3$
7:     else
8:         if $f > value\ of\ P3$ then
9:             $P1$ and $P3$ remain at the same position
10:        end if
11:    end if
12: end if
```



(before)
Median
Quartile Identifier
1. number of nodes is an even value (12 nodes),
2. number of nodes of first and second half is an even value (6 nodes),
3. new value (0.91) < 0.92 (value of old P1),
4. new value of Q1 = (0.91 + 0.92) / 2 = 0.915, new value of Q3 = (7.1 + 8.7) / 2 = 7.9
(after)

Figure 20: An example of *Quartile Identifiers* update for Case 3 with value inserted left of Q1.

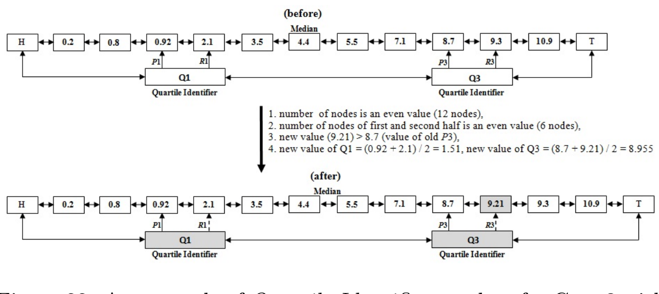
In Figure 20, the number of nodes of the *SkipList* for the feature is an even value (12 nodes) after adding the new feature value of 0.91. The new value $f \leq value\ of\ P1$, thus lines 1 to 3 in Algorithm 4 are executed and $Q1$ and $Q3$ are computed. In this case $P1$ and $P3$ are skipped to the next value left of $P1$ and $P3$ respectively. The Quartiles are computed using Equation 11.



(before)
Median
Quartile Identifier
1. number of nodes is an even value (12 nodes),
2. number of nodes of first and second half is an even value (6 nodes),
3. new value (2.9) > 0.92 (value of old P1) and < 8.7 (value of old P3),
4. new value of Q1 = (0.92 + 2.1) / 2 = 1.51, new value of Q3 = (7.1 + 8.7) / 2 = 7.9
(after)

Figure 21: An example of *Quartile Identifiers* update for Case 3 with value inserted between Q1 and Q3.

In Figure 21, the number of nodes of the *SkipList* for the feature is an even value (12 nodes) after adding the new feature value of 2.9. The new value $f > value\ of\ P1\ and\ f \leq value\ of\ P3$, thus lines 5 and 6 in Algorithm 4 are executed and $Q1$ and $Q3$ are computed. In this case $P3$ is skipped going to the next value left of $P3$ and $P1$ remains unchanged. The Quartiles are computed using Equation 11.



(before)
Median
Quartile Identifier
1. number of nodes is an even value (12 nodes),
2. number of nodes of first and second half is an even value (6 nodes),
3. new value (9.21) > 8.7 (value of old P3),
4. new value of Q1 = (0.92 + 2.1) / 2 = 1.51, new value of Q3 = (8.7 + 9.21) / 2 = 8.955
(after)

Figure 22: An example of *Quartile Identifiers* update for Case 3 with value inserted right of Q3.

In Figure 22, the number of nodes of the *SkipList* for the feature is an even value (12 nodes) after adding the new feature value of 9.21. The new value $f > value\ of\ P3$, thus lines 8 to 10 in Algorithm 4 are executed and $Q1$ and $Q3$ are computed. In this case $P1$ and $P3$ remain unchanged. The Quartiles are computed using Equation 11.

5.3.6. *Tracking and Removal of Micro-Clusters with False Positive Participation*

This section presents the proposed method for the removal of a Micro-Cluster. This method consists of two main tasks which are counting the number of false positive participations in the Micro-Cluster and measuring the 'difference' between the time stamps of the Micro-Cluster (i.e., difference between the current time stamp and the average of actual time stamps of the Micro-Cluster). Regarding false positive participation, a Micro-Cluster's false positive participation towards the absorption or training task is tracked by counting the number of times that this Micro-Cluster absorbs new instances which belong to a different class label than the actual class label of the Micro-Cluster. The initial value of this counter is 0. The counter is incremented by 1 if the particular Micro-Cluster absorbed a new instance incorrectly. Otherwise, it is decremented by 1 if the particular Micro-Cluster absorbed the new instances correctly (i.e., absorbs a new instance with the same class label as the Micro-Cluster). This subtraction is applied only if the counter value is greater than 0. Regarding measuring difference of time stamp of Micro-Clusters. This is given by the Equation 13.

$$Difference\ of\ Time\ Stamp = T_{current} - \frac{\sum_{i=1}^{n}(MC's\ Actual\ Time\ Stamp)}{n} \tag{13}$$

Where n is a number of data instances absorbed by the Micro-Cluster and $\sum_{i=1}^{n}(MC's\ Actual\ Time\ Stamp)$ is the sum of time stamps of all absorbed data instances of this Micro-Cluster (also known as $CF1^t$) and $T_{current}$ is the current time stamp. The smaller this Difference is, the more recently the Micro-Cluster has participated in absorbing data instances, as time stamps increase over time. After each time stamp, the difference between time stamps of each Micro-Cluster is calculated and the one with the largest difference is identified (the Micro-Cluster that has participated the least recently). If this Micro-Cluster also has a record of incorrect absorptions (absorption counter greater than 0), then the Micro-Cluster is removed. This is because the Micro-Cluster is considered out-dated and thus has potentially also incorrectly absorbed information.

This is a different method for estimating participation in this research compared with the Triangle Number method used in original MC-NN as described in Section 3. The reason for employing a different solution here is that the original MC-NN Triangle Method does not take false positive participation into account. Thus MC-NN may falsely regard a Micro-Cluster that has recently absorbed many false positive instances as relevant. In addition, the here proposed method only requires two steps to calculate the participation of a Micro-Cluster:

1. Calculate difference of time stamps.
2. Increment/decrement false positive counter.

The required steps for Triangle Number are outlined in Figure 3 in Section 3.4). Thus there are many more steps involved using the original method for calculating Micro-Cluster participation, and this results in unnecessary computational overheads.

6. Real-Time Feature Selection Technique using Adaptive Micro-Clusters

Three main tasks will be explained in this section which are feature analysis, feature selection, and monitoring the relevance of selected features. After detection of a concept drift, the statistical information of the features (i.e., *Velocity* and *IQR*) is analysed to identify which features were involved in the drift. Loosely speaking only features that had a significant change of their statistical information are re-examined for feature selection using Information Gain in each statistical window. This is based on the assumption that features that have not changed much are also likely to have maintained a similar Information Gain value. This can be used to reduce the computational cost of feature selection by assuming that the Information Gain has not changed for these features and thus re-calculating the Information Gain is not needed. Loosely speaking the method follows the following steps:

1. Detect Concept Drift
2. Use the Micro-Clusters statistical information (*Velocity* and *IQR*) to identify features involvement in the drift.
3. Re-calculate Information Gain only for features that have been involved in the drift.
4. Re-rank features according to their Information Gain.
5. Use new Information Gain ranking to select features.

6.1. *Feature Analysis and Feature Selection*

Feature analysis is facilitated using historical data (i.e. a statistical window between *time-1* and point-of-drift *time*). For the purpose of feature analysis, a counter is kept for each feature of a Micro-Cluster. The counter is incremented by 1 if the particular feature was the feature with the highest *IQR*. In this paper this is referred to as the history of maximum *IQR*. Using a windowing approach over the data stream, a running average of *Velocity* rate is calculated. A high *Velocity* rate combined with maximum *IQR* during a concept drift indicates that the feature has changed. The assumption here is that this particular feature may have changed its contribution towards the classification technique. Thus feature selection can be limited to only examining features that have changed their *Velocity* rate and *IQR* when there is a concept drift detected. Features that have changed are temporarily regarded as irrelevant for the data mining task, as large statistical changes have a similar effect as noise. However, the contributions of these features towards the absorption in Micro-Clusters may stabilise after the drift and thus these features are re-examined in the following time window. This will be explained in more details in the next section. From then onwards only instances comprising the relevant features are passed on to the classifier. This Feature Analysis and Feature Selection method is described in Algorithm 5.

Algorithm 5 Feature Analysis and Feature Selection

Input: Micro-Clusters statistical information (*Velocity* rate [features] and history of maximum *IQR* [features]) of a statistical window between *time-1* and point-of-drift *time*, and instance.

Output: Instance with relevant features

```
 1: for each drift detection do
 2:     List $Velocity$ rate[features] in order from low to high
 3:     Identify $median$ value of the ordered list $Velocity$ rate[features]
 4:     for each feature with $Velocity$ rate $\geq$ $median$ value do
 5:         if feature has history of maximum $IQR > 0$ value then
 6:             $instance \Leftarrow delete\ irrelevant\ feature$
 7:         end if
 8:     end for
 9: end for
10: $Classifier \Leftarrow instance\ with\ relevant\ feature(s)$
```

6.2. Monitoring and Analysis of Temporarily Irrelevant Features

The features (the ones that are temporarily regarded as irrelevant) are monitored in the following statistical window to examine their relevance using Information Gain. The assumption here is that the contribution of this particular feature towards the classification result may have changed significantly. Thus checking relevance can be limited to examining only the features that have been temporarily regarded as irrelevant. If a feature $x's$ Information Gain is greater than μ of Information Gain between window at *time-1* and at current *time* with *Percentage Difference* greater than 50% it is selected as relevant to a classifier. It should be noted here that Information Gain has been chosen as feature selection metric, as it is a popular metric for this purpose. Thus all experiments in the next section have been obtained using this metric. However, the user may decide to implement a different metric for feature selection if appropriate for the particular application. Likewise the *Percentage Difference* can be adjusted by the user requirements. However, in the experiments presented in the next section a *Percentage Difference* of 50% was used as it worked well in most cases. Figure 23 presents the aforementioned procedure in a flow chart for more clarity and Figure 24 shows an example of the statistical information (i.e., *Velocity* rate and *IQR* as well as Information Gain) of two features for six time windows.

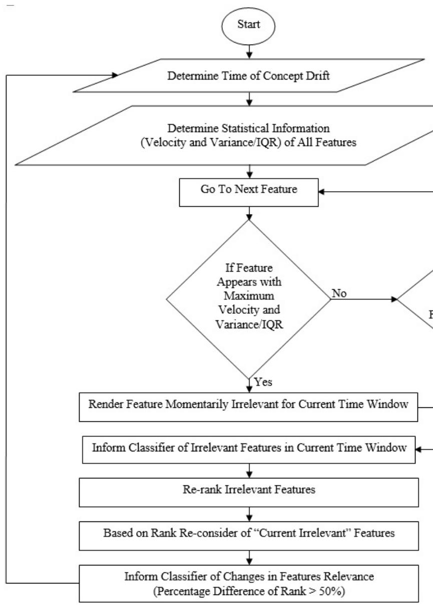
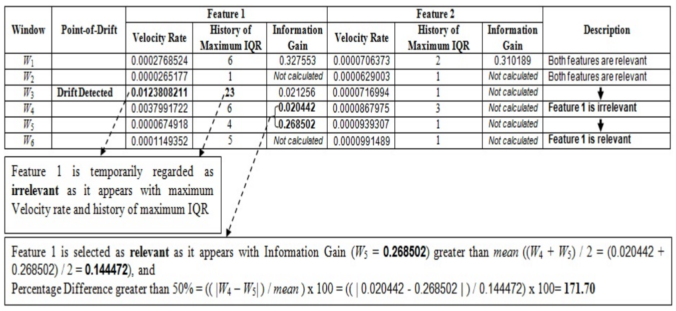

Start
Determine Time of Concept Drift
Determine Statistical Information (Velocity and Variance/IQR) of All Features
Go To Next Feature
If Feature Appears with Maximum Velocity and Variance/IQR
No
Has More Features
Yes
No
Yes
Render Feature Momentarily Irrelevant for Current Time Window
Inform Classifier of Irrelevant Features in Current Time Window
Re-rank Irrelevant Features
Based on Rank Re-consider of "Current Irrelevant" Features
Inform Classifier of Changes in Features Relevance (Percentage Difference of Rank > 50%)

Figure 23: Process of feature selection in real-time .



Window	Point-of-Drift	Feature 1			Feature 2			Description
		Velocity Rate	History of Maximum IQR	Information Gain	Velocity Rate	History of Maximum IQR	Information Gain	
W_1		0.0002768524	6	0.327553	0.0000706373	2	0.310189	Both features are relevant
W_2		0.0000265177	1	*Not calculated*	0.0000629003	1	*Not calculated*	Both features are relevant
W_3	**Drift Detected**	**0.0123808211**	**23**	0.021256	0.0000716994	1	*Not calculated*	↓
W_4		0.0037991722	6	**0.020442**	0.0000867975	3	*Not calculated*	**Feature 1 is irrelevant**
W_5		0.0000674918	4	**0.268502**	0.0000939307	1	*Not calculated*	↓
W_6		0.0001149352	5	*Not calculated*	0.0000991489	1	*Not calculated*	**Feature 1 is relevant**

Feature 1 is temporarily regarded as **irrelevant** as it appears with maximum Velocity rate and history of maximum IQR

Feature 1 is selected as **relevant** as it appears with Information Gain (W_5 = **0.268502**) greater than *mean* (($W_4 + W_5$) / 2 = (0.020442 + 0.268502) / 2 = **0.144472**), and
Percentage Difference greater than 50% = (($|W_4 - W_5|$) / *mean*) x 100 = ((| 0.020442 - 0.268502 |) / 0.144472) x 100= **171.70**

Figure 24: An example of feature analysis, feature selection, and monitoring of temporarily irrelevant features. Assumed Information Gains are indicated in italics and actual Information Gain calculations are not in italics.

Initially Information Gain is calculated for all features at window W_1. In this example it is assumed that both features are initially relevant. Drift is detected at window W_3, consider the *Velocity* rate and history of maximum *IQR* of feature 1 equal to 0.0123808211 and 23, concurrently. This represents that feature 1 appears with maximum *Velocity* rate and a history of maximum *IQR* compared with the remaining features (in this case only feature 2). Thus feature 1 is temporarily regarded as irrelevant to the classifier after W_3. Information Gains of irrelevant features are then calculated in every time window in order to monitor if they return to being relevant. In this case feature 1 appears again with an Information Gain at $W_5 = 0.268502$, which differs from μ by 50%. This indicates that feature 1 became relevant again for classification tasks after W_5. Please note that the Information Gain of features that have remained relevant are assumed not to have changed and thus are only re-calculated if there is a drift detected. In this example assumed Information Gains are indicated in italics. For example for feature 2 there is only one Information Gain calculation at the beginning in W_1. However, should there be another concept drift in the future and feature 2 appears with maximum *Velocity* rate and history of maximum *IQR*, then its Information Gain is likely to have changed and thus it would be re-calculated.

7. Experimental Evaluation

This section first provides information about the experimental setup and then presents an extensive empirical evaluation of the proposed techniques.

7.1. Experimental Setup

The implementation of experiments were realised in the Massive Online Analysis (MOA) framework [49]. Two types of data were used, artificial data stream generators from the MOA framework and real datasets. The reason for also using artificial datasets is because MOAs data stream generators enable the introduction of different kinds of concept drift deliberately and thus allow us to evaluate against a ground truth in terms of concept drift. The source code implemented for this paper has been made available here[1].

7.1.1. Artificial Datasets

The following artificial data stream generators were used: ***SEA Generator***, this data stream was introduced in [50], it generates data comprising continuous attributes, whereas the third attribute is irrelevant for distinguishing between the class labels. The ***HyperPlane Generator*** was also used, it creates a linearly separable model. It slowly rotates in 'D' dimensions continuously changing the linear decision boundary of the stream [51]. This constant concept change makes it very difficult for data stream classifiers to keep a good classification accuracy and remain computationally efficient. The final data stream generator used was the ***Random Tree Generator***, which was introduced in [13] and generates a stream based on a randomly generated tree. New examples are generated by assigning uniformly distributed random values to features, which then determine the class label using the random tree. Nine datasets were generated using the aforementioned stream generators, each comprising three features, two class labels and a concept drift. The concept drift was always induced gradually half way through the stream by both, inducing a concept drift through the MOA data stream generators implemented methods and by swapping features (sudden concept drift). Details about the concept drift methods of the individual streams can be found in various sources, notably [50, 51, 13]. The reason for inducing concept drift by swapping features is that this enables the testing as to which of the features has changed its contribution to the underlying model. One would expect the methods proposed here to identify the two swapped features as the cause of the concept drift. For each of these three datasets having concept drift, a second and third version of the datasets has been generated which included different levels of noise in order to validate the robustness of the concept drift detection and feature tracking methods. The participation threshold Ω was set to 50 for each experiment, which yielded good results in most cases [39]. The error threshold Θ was set to the best performing one for each dataset. Table 2 shows an overview of the generated streams including the setting of the proposed technique and which features have been swapped. In the experiments a window size equals to 10% of the total number of instances. The expression **Time t** refers to a particular time window. I.e. according to Table 2 time *T=1* refers to instances 1-1000, *T=2* to instances 1001-2000, etc. Regarding the *FIFO*, it was set to 1000 instances, as this setting yielded good results in terms of classification accuracy in most cases. The classification accuracy was calculated using the Prequential Testing method implemented in MOA [52], which essentially calculates a running average of the classification accuracy.

Table 2: Setup of the artificial datasets. Drifts were generated through individual data stream generators and by swapping features.

Dataset	Number of Instances Generated	Start of Concept Drift by Generator	Window Size	Θ	Index of Swapped Features	Start of Swapped Features	Percentage of Noise
SEA	10,000	5000	1000	3	2 with 3	6000	-
SEA	10,000	5000	1000	34	2 with 3	6000	15
SEA	10,000	5000	1000	75	2 with 3	6000	25
HyperPlane	10,000	5000	1000	6	1 with 2	6000	-
HyperPlane	10,000	5000	1000	13	1 with 2	6000	15
HyperPlane	10,000	5000	1000	27	1 with 2	6000	25
RandomTree	10,000	5000	1000	644	1 with 2	6000	-
RandomTree	10,000	5000	1000	726	1 with 2	6000	15
RandomTree	10,000	5000	1000	685	1 with 2	6000	25

7.1.2. Real Datasets

Two sets of experiments were setup. One controlled set of experiments to validate whether the method can detect concept drifts and causality of concept drift on real datasets correctly. For this features were swapped to generate a known ground truth. Four real datasets with continuous features were chosen randomly from the UCI Machine Learning Repository [53]. Table 3 shows an overview of real datasets including setting of the proposed technique and which features were swapped. The second set of experiments was uncontrolled, real data sets where used and no features were swapped, in order to show that the method presented in this paper is robust to more realistic application scenarios. The proposed concept drift detection method, feature tracking method and also feature selection method were applied. The robustness of the method was measured by applying a Hoeffding tree classifier and classification accuracy was monitored over time. Also a control group of experiments was setup, where only a Hoeffding tree classifier was applied thus excluding any

[1]https://github.com/mahmoodshakir/New-Micro-Cluster-Nearest-Neighbour-MC-NN-for-Real-Time-Preprocessing-Technique

of the techniques developed in this paper. The Hoeffding tree classifier has been chosen as it is one of the most popular data stream classifiers and best performing classifier in the MOA framework [54]. However, our method is independent of the classifier and the user may choose a different classification method. Six real datasets with continuous features were chosen randomly from the UCI Machine Learning Repository [53]. Table 4 shows an overview of real datasets including settings of the proposed technique. For both sets of experiments Ω (threshold for the Micro-Cluster minimum participation) was set to 50 as this yielded good results in most cases. Whereas Θ was set to a relevant value that retrieves good results for each individual dataset. The time window size and *FIFO* size were chosen the same way as for the artificial data streams (see Section 7.1.1).

Table 3: Setup of the real datasets for the controlled set of experiments for concept drift detection and feature tracking.

Dataset	Number of Instances	Number of Features	Number of Class Labels	Θ	Index of Swapped Features	Time of Swapping
Bank	41,188	10	2	1500	1 with 10	6
CoverType	581,012	10	7	12,000	6 and 7 with 8 and 9	3
Diabetes	768	8	2	37	1 with 2	4
KDDCUP	494,021	10	23	1000	2,3, and 4 with 8,9, and 10	4

Table 4: Setup of real datasets for the uncontrolled set of experiments for concept drift detection, feature tracking and feature selection.

Real Dataset	**Number of Instances**	**Number of Features**	**Number of Class Labels**	***FIFO* Size (Maximum)**	Θ
CoverType	581,012	54	7	1000	10,000
EEG Eye State	14,980	14	2	1000	200
Gesture Phase Segmentation	1,747	19	5	1000	56
Poker Hand	1,000,000	10	10	1000	51,000
Statlog (Landsat Satellite)	4,435	36	7	1000	28
Waveform (with noise)	5,000	40	3	1000	50

7.2. Results

The results presented as first in Section 7.2.1 analysed the effect of noise, feature-bias and outliers on the detection of concept drift. This evaluation is based on original MC-NN using the readily available *Variance* statistics of the Micro-Clusters and *Split* and *Death* rate to detect concept drift. Then in Section 7.2.2, the technique proposed for concept drift detection in Section 4 is compared with the results in Section 7.2.1 to evaluate the proposed feature tracking method. Finally Section 7.2.3 applies the method in combination with a data stream classifier for real-time feature selection and measures the change in accuracy.

7.2.1. Real-Time Concept Drift Detection Technique

The Micro-Clusters *Split* and *Death* rates were used for detecting drifts. For the experiments the default parameters stated in Table 2 of the technique were used. The evaluation incorporated several levels of noise in the artificial data stream, as listed in Table 2 and the real datasets are described in Table 3.

In Figures 25, 26, and 27 the *Percentage Difference* is displayed up to 100%, however, this can be much higher

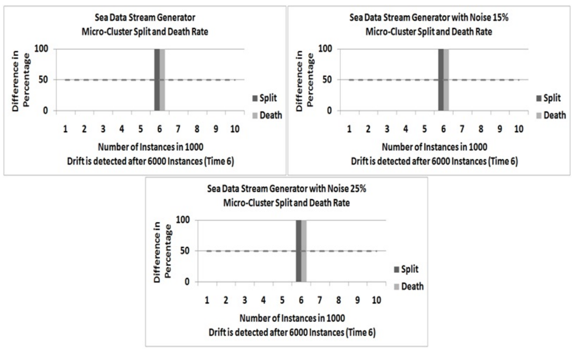


Figure 25: Results of Sea Data Stream Generator for drift detection using Micro-Cluster Percentage Difference of Split and Death rate.

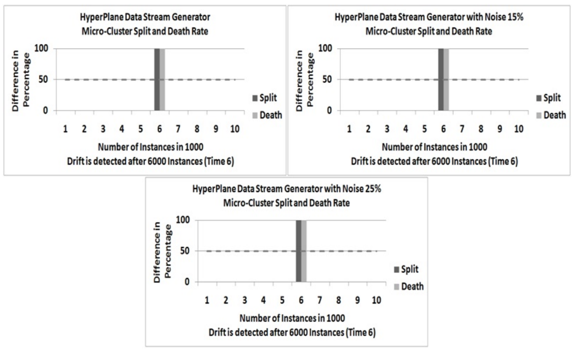


Figure 26: Results of HyperPlane Data Stream Generator for drift detection using Micro-Cluster Percentage Difference of Split and Death rate.

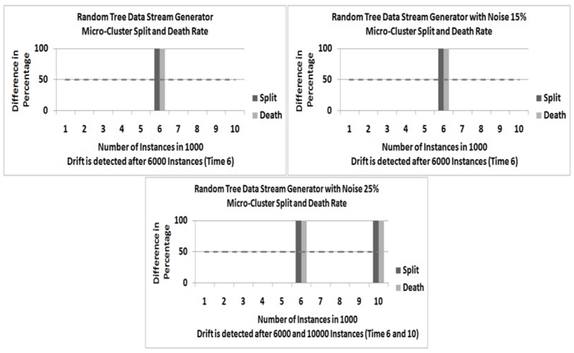


Figure 27: Results of Random Tree Data Stream Generator for drift detection using Micro-Cluster Percentage Difference of Split and Death rate.

than 100%. For readability of the figures a maximum of 100% difference is displayed. Displaying differences above 100% is not interesting as the concept drift detection was triggered once a difference of at least 50% had been reached for both *Split* and *Death* rates. This 100% cut-off is applied on all subsequent figures in this paper referring to *Percentage Differences* of *Split* and *Death* rates. During this time higher *Split* and *Death* rates are expected as the set of Micro-Clusters adapts to the new concept, the feature swap. In the figures, it can be seen that the *Split* and *Death* rates at the time of concept drift (after time 5) increase as expected for all artificial data streams and noise levels. The noise levels do not seem to affect the concept drift detection considerably; however, for the RandomTree generator with a noise level of 25%, at time window 10, the algorithm did arrive at a false positive detection i.e. detected a concept drift that was not there. This could be an indication that the Micro-Clusters for RandomTree became unstable due to noise. This is discussed further in the next section where mechanisms to deal with noise are incorporated in the feature tracking method.

Next the concept drift detection method was compared with existing state-of-the-art drift detection methods CUSUM, DDM, EDDM, EWMA, and ADWIN (see section 2 for more details about these methods) on the same artificial data streams. Table 5 shows the time when each of the methods including the proposed method, detected a drift and if it was detected on time. As it can be seen, the proposed technique always detected the drift at the correct time except for the RandomTree data stream with 25% noise as has been discussed previously.

Table 5: Adaptation to concept drift using the initially proposed and other state-of-the-art techniques.

Generator	Technique	Number of Drift Detections	Times when Drift Detected	Drift Detected
Sea	**The Proposed Technique**	1	6	**Correctly**
	CUSUM	1	5	Incorrectly
	DDM	2	5 and 8	Incorrectly
	EDDM	2	5 and 8	Incorrectly
	EWMA	9	1 to 9	**Correctly**
	ADWIN	1	5	Incorrectly
Sea with Noise 15	**The Proposed Technique**	1	6	**Correctly**
	CUSUM	2	5 and 8	Incorrectly
	DDM	1	5	Incorrectly
	EDDM	2	1 and 4	Incorrectly
	EWMA	9	1 to 9	**Correctly**
	ADWIN	2	5 and 9	Incorrectly
Sea with Noise 25	**The Proposed Technique**	1	6	**Correctly**
	CUSUM	2	5 and 6	**Correctly**
	DDM	1	4	Incorrectly
	EDDM	3	1,4, and 6	**Correctly**
	EWMA	9	1 to 9	**Correctly**
	ADWIN	3	4,5, and 8	Incorrectly
HyperPlane	**The Proposed Technique**	1	6	**Correctly**
	CUSUM	1	5	Incorrectly
	DDM	1	5	Incorrectly
	EDDM	1	5	Incorrectly
	EWMA	8	1 to 5 and 7 to 9	Incorrectly
	ADWIN	3	2,5, and 9	Incorrectly
HyperPlane with Noise 15	**The Proposed Technique**	1	6	**Correctly**
	CUSUM	1	5	Incorrectly
	DDM	2	5 and 7	Incorrectly
	EDDM	1	5	Incorrectly
	EWMA	9	1 to 9	**Correctly**
	ADWIN	2	2 and 5	Incorrectly
HyperPlane with Noise 25	**The Proposed Technique**	1	6	**Correctly**
	CUSUM	2	5 and 6	**Correctly**
	DDM	2	5 and 6	**Correctly**
	EDDM	2	4 and 7	Incorrectly
	EWMA	9	1 to 9	**Correctly**
	ADWIN	3	2,5, and 8	Incorrectly
RandomTree	**The Proposed Technique**	1	6	**Correctly**
	CUSUM	1	5	Incorrectly
	DDM	1	5	Incorrectly
	EDDM	1	5	Incorrectly
	EWMA	9	1 to 9	**Correctly**
	ADWIN	2	5 and 9	Incorrectly
RandomTree with Noise 15	**The Proposed Technique**	1	6	**Correctly**
	CUSUM	1	5	Incorrectly
	DDM	1	5	Incorrectly
	EDDM	1	5	Incorrectly
	EWMA	9	1 to 9	**Correctly**
	ADWIN	2	5 and 9	Incorrectly
RandomTree with Noise 25	**The Proposed Technique**	2	6 and 10	**Correctly**
	CUSUM	1	7	Incorrectly
	DDM	-	-	-
	EDDM	-	-	-
	EWMA	9	1 to 9	**Correctly**
	ADWIN	-	-	-

If a method detects a drift falsely, then this would cause unnecessary adaptation by the classifier to a non-existing drift. This is referred to as a *false positive*. Correctly detected drifts are referred to as *true positives*. The number of experiments (each with one actual drift) are shown in Table 6. As there are 9 experiments, there is a total of 9 concept drifts to be detected. In the table it is indicated how many *true* and *false positives* each method detected. As can be seen, the proposed technique detected all concept drifts and only had one *false positive* detection. The best competitor in this regard, EWMA, detected 8 *true positives*, 1 less than the proposed method. However, EWMA has a very high *false positive* number of 72, compared with only 1 for the proposed method. Thus EWMA is triggering frequent and unnecessary adaptation to concept drift. Also the remaining competitors found fewer *true positives* and a much higher number of *false positives* compared with the proposed method.

Table 6: Summary of concept drift adaptation experiments referring to Table 5.

Technique	True Positives	False Positives
The Proposed Technique	**9**	**1**
CUSUM	2	10
DDM	1	10
EDDM	1	12
EWMA	8	72
ADWIN	0	18
Total:	21	123

The method has also been applied on real datasets as shown in Figure 28 where for each case 1 concept drift has been introduced through the swapping of features as listed in Table 3.

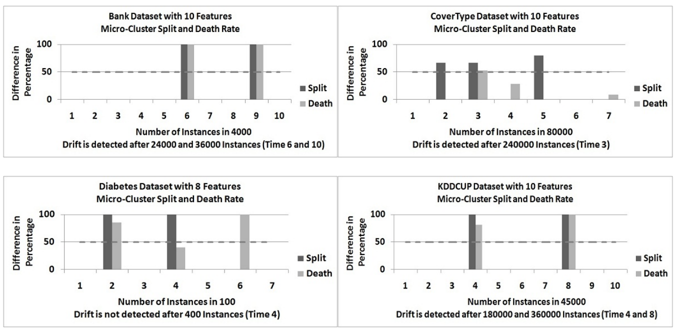
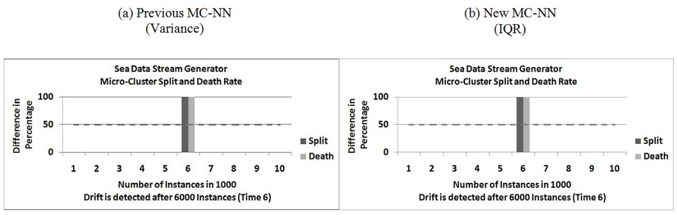


Figure 28: The results of real-dataset for drift detection using Micro-Cluster Percentage Difference of Split and Death rate.

In Figure 28, it can be seen that the *Split* and *Death Percentage Differences* at the time of concept drift increase with *Bank* dataset, *CoverType* dataset, and *KDDCUP* dataset, indicating that the current set of Micro-Clusters does not fit the concept encoded in the data anymore. Whereas, in the figure, drift is not detected with the *Diabetes* dataset. As these data streams are based on real datasets it is believed that they potentially contain feature-bias, noise and outliers and that these may be the reasons for the proposed method not being able to detect the concept drift. The next section will compare this method with the new proposed feature tracking method, which is more robust to feature-bias, noise and outliers.

7.2.2. Tracking Features using Variance and IQR

The evaluation was based on original MC-NN using the readily available *Variance* statistics of the Micro-Clusters and *Split* and *Death* rates to detect concept drift. This section applies *Normalisation* and *LPF* in order to address these effects and compares these results with those from Section 7.2.1. Furthermore this section compares the previously discussed approach for feature tracking based on *Variance* combined with feature *Velocity* to the more robust approach based on *IQR* combined with feature *Velocity.* For the experiments the default parameters stated in Table 2 (artificial datasets) of the technique were used unless stated otherwise. Regarding the artificial datasets, no *Normalisation* was applied as the data generators already produced normalised data.



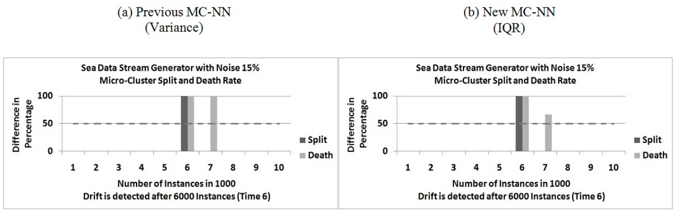
Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 2	**0.01221**	**2**	True Positive
Feature 3	**0.00997**	**1**	True Positive
Feature 1	0.00202	1	

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 3	**0.00229**	**3**	True Positive
Feature 2	**0.00158**	**2**	True Positive
Feature 1	0.00118	0	

Figure 29: The results of SEA data stream generator with a noise level of 0% for tracking features using Micro-Clusters.



Features are ordered based on *Velocity* rate

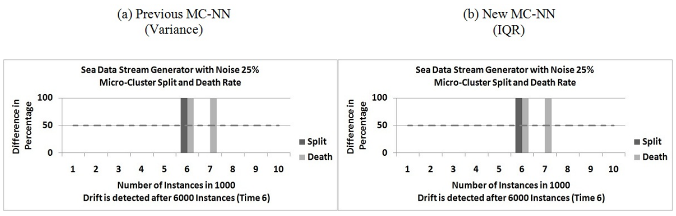
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 2	0.00258	0	False Negative
Feature 3	**0.00212**	**1**	True Positive
Feature 1	0.00202	0	

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 3	**0.00230**	**2**	True Positive
Feature 1	0.00094	0	
Feature 2	0.00078	0	False Negative

Figure 30: The results of SEA data stream generator with a noise level of 15% for tracking features using Micro-Clusters.

Part (a) of Figures 29, 30, 31, 32, 33, 34, 35, 36, and 37 shows the results for using MC-NN with *Variance* and history of maximum *Variance* for concept drift detection and



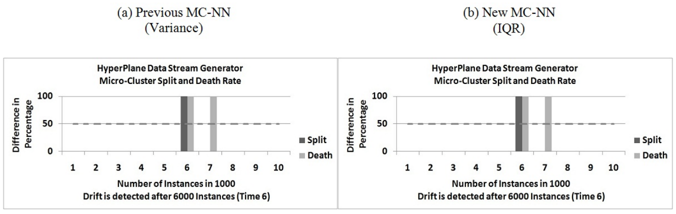
Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 2	0.00253	0	False Negative
Feature 3	**0.00227**	**1**	True Positive
Feature 1	0.00216	0	

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 3	**0.00121**	**1**	True Positive
Feature 1	0.00021	0	
Feature 2	0.00016	0	False Negative

Figure 31: The results of SEA data stream generator with a noise level of 25% for tracking features using Micro-Clusters.



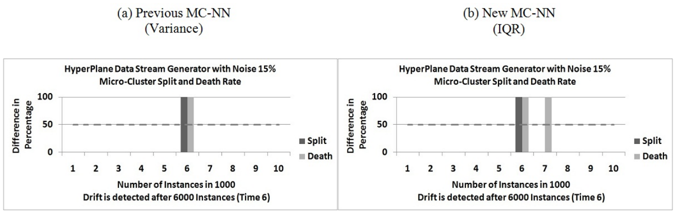
Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00046	1	False Positive
Feature 2	0.00035	0	False Negative
Feature 1	0.00033	0	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00016**	**1**	True Positive
Feature 2	0.00009	0	False Negative
Feature 3	0.00008	1	

Figure 32: The results of HyperPlane data stream generator with a noise level of 0% for tracking features using Micro-Clusters.



Features are ordered based on *Velocity* rate

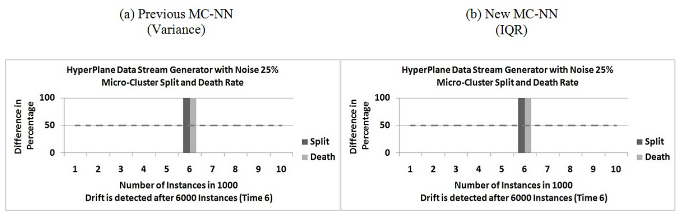
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 2	0.00126	0	False Negative
Feature 3	0.00117	2	False Positive
Feature 1	0.00117	0	False Negative

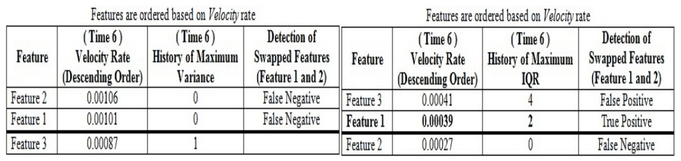
Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00019**	**1**	True Positive
Feature 3	0.00014	0	
Feature 2	0.00013	1	False Negative

Figure 33: The results of HyperPlane data stream generator with a noise level of 15% for tracking features using Micro-Clusters.

feature tracking, and part (b) of the figures shows the corresponding results for using MC-NN with *IQR* and history of maximum *IQR*. Different figures correspond to different data streams and noise levels. Regarding concept drift detection, the same results as the previous section have been achieved for both, using *Variance* and *IQR*. All concept drifts were detected correctly, however, both methods also detected a false concept drift for time 10 of the RandomTree generator at noise level 25%. However, it is





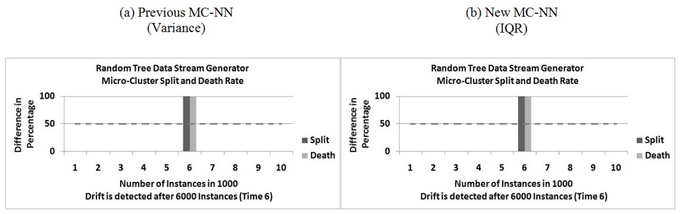
Features are ordered based on *Velocity* rate

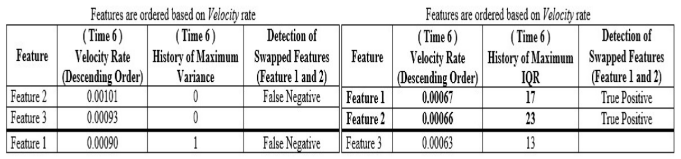
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 2	0.00106	0	False Negative
Feature 1	0.00101	0	False Negative
Feature 3	0.00087	1	

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00041	4	False Positive
Feature 1	**0.00039**	**2**	True Positive
Feature 2	0.00027	0	False Negative

Figure 34: The results of HyperPlane data stream generator with a noise level of 25% for tracking features using Micro-Clusters.





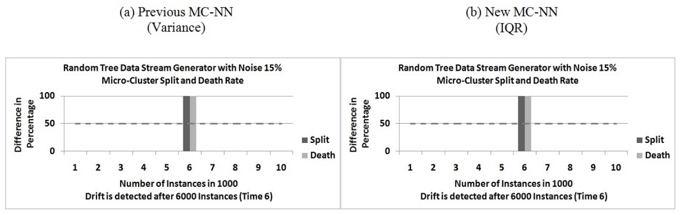
Features are ordered based on *Velocity* rate

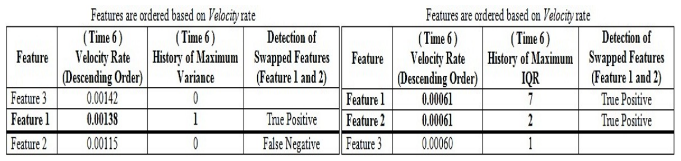
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 2	0.00101	0	False Negative
Feature 3	0.00093	0	
Feature 1	0.00090	1	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00067**	**17**	True Positive
Feature 2	**0.00066**	**23**	True Positive
Feature 3	0.00063	13	

Figure 35: The results of Random Tree data stream generator with a noise level of 0% for tracking features using Micro-Clusters.





Features are ordered based on *Velocity* rate

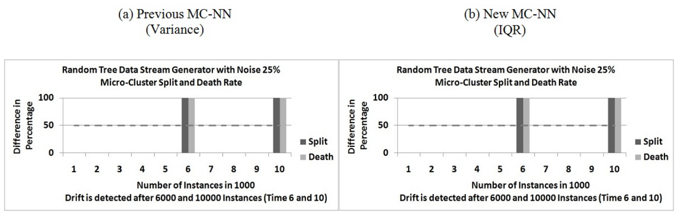
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00142	0	
Feature 1	**0.00138**	**1**	True Positive
Feature 2	0.00115	0	False Negative

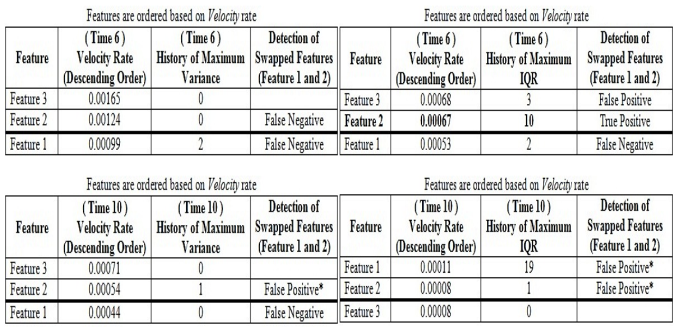
Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00061**	**7**	True Positive
Feature 2	**0.00061**	**2**	True Positive
Feature 3	0.00060	1	

Figure 36: The results of Random Tree data stream generator with a noise level of 15% for tracking features using Micro-Clusters.

likely not to cause a degradation of classification accuracy, as features are merely flagged up to the feature selection method to have potentially changed their relevance but it is then up to the feature selection method to evaluate these features and decide if they should be included. If a concept drift is detected, then the methods use the feature *Velocity* and history of maximum *Variance* or *IRQ* respectively to decide which features should be flagged up to be considered for inclusion or removal from the currently considered





Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00165	0	
Feature 2	0.00124	0	False Negative
Feature 1	0.00099	2	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00068	3	False Positive
Feature 2	**0.00067**	**10**	True Positive
Feature 1	0.00053	2	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 10) Velocity Rate (Descending Order)	(Time 10) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00071	0	
Feature 2	0.00054	1	False Positive*
Feature 1	0.00044	0	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 10) Velocity Rate (Descending Order)	(Time 10) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	0.00011	19	False Positive*
Feature 2	0.00008	1	False Positive*
Feature 3	0.00008	0	

Figure 37: The results of Random Tree data stream generator with a noise level of 25% for tracking features using Micro-Clusters.

features. The results for this are depicted at the bottom part in the aforementioned figures. As a reminder, the features are ranked according to their velocities and the 50% features with the highest *Velocity* are examined closer. As there are 3 features in each stream, the 2 features with the highest *Velocity* are always selected. These features are then flagged up if they also appeared in the history of maximum *Variance* or *IQR* in the previous time window. This has already been explained in greater detail in Section 6.1. It is expected here that features that have been swapped during the concept drift are more likely to be flagged up for inclusion or removal from the currently considered set of features.

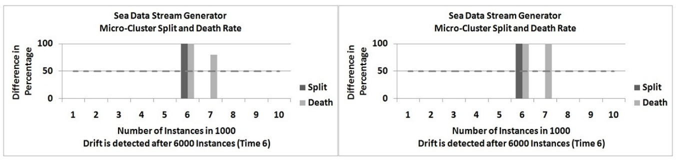
Table 7: Summary of the experimental results with artificial datasets generated with noise levels of 0%, 15% and 25%. The results are reported for the Time 6 which is the time of swapped features.

Generator	True Positive (*Variance*)	True Positive (*IQR*)	False Positive (*Variance*)	False Positive (*IQR*)
Sea with Noise 0%	2	2		
Sea with Noise 15%	1	1		
Sea with Noise 25%	1	1		
HyperPlane with Noise 0%		1	1	
HyperPlane with Noise 15%		1	1	
HyperPlane with Noise 25%		1		1
RandomTree with Noise 0%		2		2
RandomTree with Noise 15%	1	2		
RandomTree with Noise 25%		1	1*	2 + 2*
Total:	5	12	3	7

Table 7 summarises the feature tracking results presented in Figures 29 to 37. Numbers indicated with a * indicate potentially false positives detections of features that changed their relevance. Potentially in the preceding sentence is to reflect the fact that in this case that there was also an unexpected concept drift. However, it is not possible to verify with absolute certainty if the detection was indeed a false positive. It can be seen that the here presented method based on maximum *IQR* achieves a much higher *true positive* detection of features that changed. However, it also has a higher number *false positive* detections compared with the method based on *Variance*. Now the *true positive* figures are based on the fact the features are known to have changed as they have been swapped at the time of concept drift. However, considering the fact that here the native method of each artificial stream generator has also been used for inducing a concept drift then some of the *false positive* detections may very well be *true positives*. Even if they were not *true positives*, it would merely mean that they are flagged to the feature selection method to be reconsidered for inclusion or exclusion of the feature set to be considered for adaptation. Loosely speaking, correct *true positive* detection numbers are more important than *false positive* detection numbers.

In order to remove the possibility of changing feature relevance due to artificial concept drift generators, the feature tracking experiments on the artificial datasets were re-conducted. This time no concept drift was introduced apart from the swapping of the features. Thus the only features that have changed their relevance must be the swapped features. The results of these experiments are displayed in Figures 38 to 46. As it can be seen, and as expected, both methods have a lower *false positive* count of flagging features that changed relevance. Again the proposed method based on *IQR* performs best, it detected all features that were swapped.

(a) Previous MC-NN (Variance) | (b) New MC-NN (IQR)



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 3	**0.00772**	**2**	True Positive
Feature 2	0.00754	0	False Negative
Feature 1	0.00721	0	

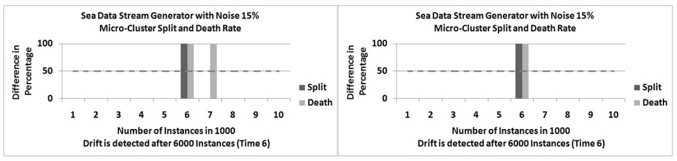
Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 3	**0.00180**	**5**	True Positive
Feature 2	**0.00097**	**1**	True Positive
Feature 1	0.00071	0	

Figure 38: The results of SEA data stream generator with a noise level of 0% for tracking features using Micro-Clusters.

Table 8 summarises the feature tracking results presented in Figures 38 to 46. Altogether there were 18 features swapped so they have all potentially changed their relevance and thus should be identified by the methods. As it can be seen the method based on original MC-NN with *Variance* identifies 4 features correctly and also resulted in 3 *false positives*. The proposed method based on *IQR* detected 16 features that potentially changed their relevance correctly and resulted in 2 *false positives*. It can be seen that the proposed method based on maximum *IQR* achieves a much higher *true positive* detection of features

(a) Previous MC-NN (Variance) | (b) New MC-NN (IQR)



Features are ordered based on *Velocity* rate

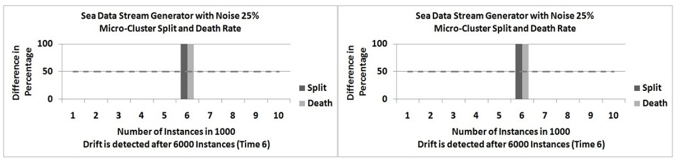
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 2	0.00278	0	False Negative
Feature 3	**0.00236**	**1**	True Positive
Feature 1	0.00223	0	

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 2	**0.00301**	**11**	True Positive
Feature 3	**0.00281**	**21**	True Positive
Feature 2	0.00280	3	

Figure 39: The results of SEA data stream generator with a noise level of 15% for tracking features using Micro-Clusters.

(a) Previous MC-NN (Variance) | (b) New MC-NN (IQR)



Features are ordered based on *Velocity* rate

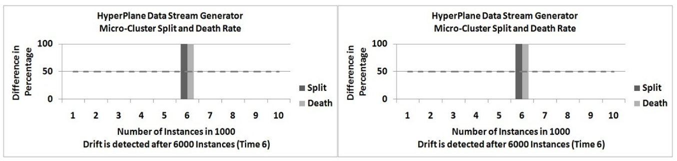
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 2 and 3)
Feature 1	0.01186	0	
Feature 2	0.01090	0	False Negative
Feature 3	0.01017	2	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 2 and 3)
Feature 2	**0.00371**	**7**	True Positive
Feature 3	**0.00342**	**4**	True Positive
Feature 1	0.00334	1	

Figure 40: The results of SEA data stream generator with a noise level of 25% for tracking features using Micro-Clusters.

(a) Previous MC-NN (Variance) | (b) New MC-NN (IQR)



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00092	1	False Positive
Feature 1	**0.00086**	**1**	True Positive
Feature 2	0.00070	0	False Negative

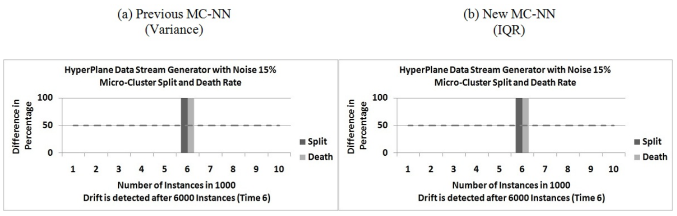
Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00013**	**2**	True Positive
Feature 2	0.00009	0	False Negative
Feature 3	0.00006	1	

Figure 41: The results of HyperPlane data stream generator with a noise level of 0% for tracking features using Micro-Clusters.

that changed with only 2 *false positives*. The *true positive* figures are based on features that are known to have changed as they have been swapped at the time of concept drift.

Next both approaches were tested on real datasets. For the experiment the default parameters, (stated in Table 3) of the technique were used unless stated otherwise. Real-time Min and Max *Normalisation* and *LPF* were applied to minimise the effect of feature-bias and noise, respec-



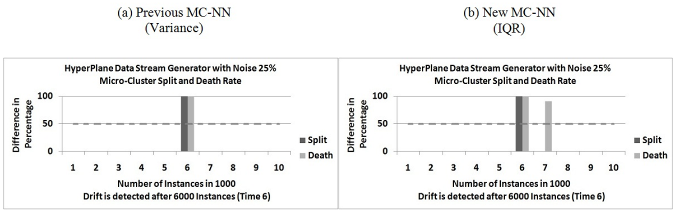
Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 1	0.00190	0	False Negative
Feature 3	0.00190	1	False Positive
Feature 2	0.00178	1	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 2	**0.00072**	**8**	True Positive
Feature 1	**0.00071**	**21**	True Positive
Feature 3	0.00070	20	

Figure 42: The results of HyperPlane data stream generator with a noise level of 15% for tracking features using Micro-Clusters.



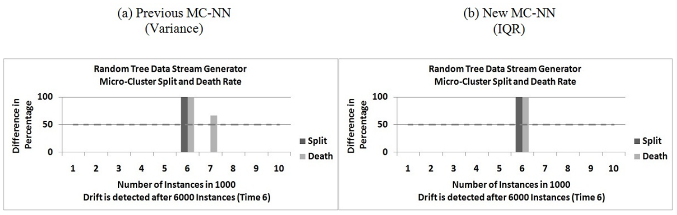
Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 2	**0.00158**	1	True Positive
Feature 1	0.00133	0	False Negative
Feature 3	0.00121	0	

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 2	**0.00029**	9	True Positive
Feature 1	**0.00027**	11	True Positive
Feature 3	0.00026	10	

Figure 43: The results of HyperPlane data stream generator with a noise level of 25% for tracking features using Micro-Clusters.



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00050	0	
Feature 2	0.00029	0	False Negative
Feature 1	0.00027	1	False Negative

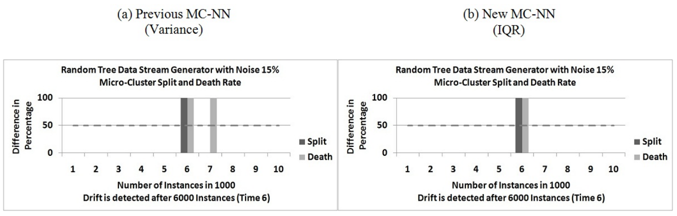
Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00030**	**5**	True Positive
Feature 2	**0.00030**	**4**	True Positive
Feature 3	0.00029	2	

Figure 44: The results of Random Tree data stream generator with a noise level of 0% for tracking features using Micro-Clusters.

tively. A random number of random feature pairs have been selected to be swapped in order to validate whether the feature tracking method can identify the changed features.

Figures 47, 48, 49, and 50 visualise the results for concept drift detection and feature tracking in the same way as Figures 29 to 37 do for the artificial data streams. Again part (a) of the figures refers to the method based on variance and part (b) of the figures refers to the method based



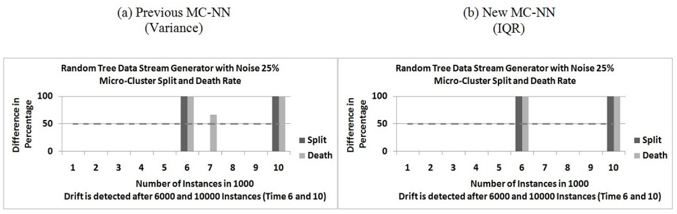
Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00051	0	
Feature 1	0.00042	0	False Negative
Feature 2	0.00041	1	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00031**	**20**	True Positive
Feature 2	**0.00023**	1	True Positive
Feature 3	0.00019	0	

Figure 45: The results of Random Tree data stream generator with a noise level of 15% for tracking features using Micro-Clusters.



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00044	0	
Feature 2	0.00032	0	False Negative
Feature 1	0.00031	1	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 1	**0.00008**	**2**	True Positive
Feature 3	0.00006	0	
Feature 2	0.00005	0	False Negative

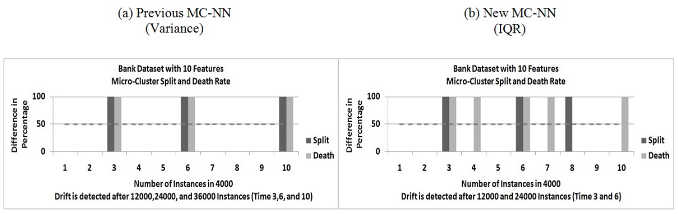
Features are ordered based on *Velocity* rate

Feature	(Time 10) Velocity Rate (Descending Order)	(Time 10) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 2)
Feature 3	0.00124	0	
Feature 2	0.00116	1	False Positive*
Feature 1	0.00079	0	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 10) Velocity Rate (Descending Order)	(Time 10) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 2	0.00068	4	False Positive*
Feature 1	0.00065	4	False Positive*
Feature 3	0.00056	0	

Figure 46: The results of Random Tree data stream generator with a noise level of 25% for tracking features using Micro-Clusters.



Features are ordered based on *Velocity* rate

Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum Variance	Detection of Swapped Features (Feature 1 and 10)
Feature 8	7.60664	2	False Positive
Feature 1	7.55771	0	False Negative
Feature 9	5.48911	0	
Feature 3	0.39137	0	
Feature 2	0.29615	1	False Positive
Feature 10	0.09573	0	False Negative
Feature 7	0.00413	0	
Feature 4	0.00000	0	
Feature 5	0.00000	0	
Feature 6	0.00000	0	

Features are ordered based on *Velocity* rate

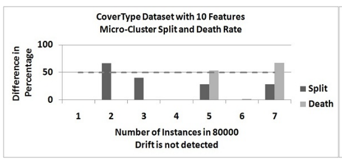
Feature	(Time 6) Velocity Rate (Descending Order)	(Time 6) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 10)
Feature 10	**0.00058**	**1**	True Positive
Feature 8	0.00032	0	
Feature 1	**0.00012**	**2**	True Positive
Feature 9	0.00007	0	
Feature 2	0.00002	0	
Feature 3	0.00001	0	
Feature 7	0.00001	0	
Feature 4	0.00000	0	
Feature 5	0.00000	0	
Feature 6	0.00000	0	

Figure 47: The results of Bank dataset with 10 features for tracking features using Micro-Clusters.

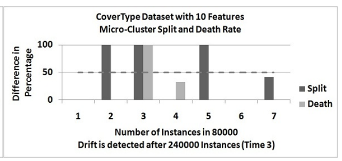
Table 8: Summary of the experimental results with artificial datasets generated with noise levels of 0%, 15% and 25%. The results are reported for Time 6 which is the point at which features had been swapped.

Generator	True Positive (*Variance*)	True Positive (*IQR*)	False Positive (*Variance*)	False Positive (*IQR*)
Sea with Noise 0%	1	2		
Sea with Noise 15%	1	2		
Sea with Noise 25%		2		
HyperPlane with Noise 0%	1	1	1	
HyperPlane with Noise 15%		2	1	
HyperPlane with Noise 25%	1	2		
RandomTree with Noise 0%		2		
RandomTree with Noise 15%		2		
RandomTree with Noise 25%		1	1*	2*
Total:	4	16	3	2

(a) Previous MC-NN (Variance)



(b) New MC-NN (IQR)



Features are ordered based on *Velocity* rate

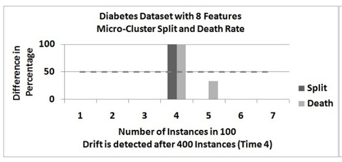
Feature	(Time 3) Velocity Rate (Descending Order)	(Time 3) History of Maximum Variance	Detection of Swapped Features (Features 6,7,8 and 9)
Feature 9	0.54267	0	False Negative
Feature 1	0.38065	0	
Feature 7	0.31810	0	False Negative
Feature 8	0.30382	0	False Negative
Feature 2	0.21634	0	
Feature 5	0.15249	0	
Feature 10	0.15043	0	
Feature 3	0.09466	0	
Feature 4	0.08274	0	
Feature 6	0.01658	0	False Negative

Features are ordered based on *Velocity* rate

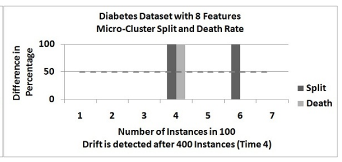
Feature	(Time 3) Velocity Rate (Descending Order)	(Time 3) History of Maximum IQR	Detection of Swapped Features (Features 6,7,8 and 9)
Feature 9	0.13627	0	False Negative
Feature 1	0.08728	0	
Feature 8	**0.07995**	**1**	True Positive
Feature 2	0.07212	82	False Positive
Feature 7	**0.04696**	**23**	True Positive
Feature 10	0.04182	0	
Feature 5	0.04157	0	
Feature 3	0.04001	0	
Feature 4	0.01528	2	
Feature 6	0.00392	0	False Negative

Figure 48: The results of CoverType dataset with 10 features for tracking features using Micro-Clusters.

(a) Previous MC-NN (Variance)



(b) New MC-NN (IQR)



Features are ordered based on *Velocity* rate

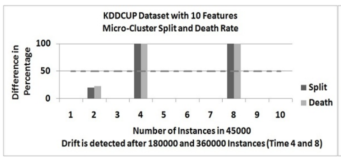
Feature	(Time 4) Velocity Rate (Descending Order)	(Time 4) History of Maximum Variance	Detection of Swapped Features (Features 1 and 2)
Feature 2	1.32981	0	False Negative
Feature 3	1.04979	0	
Feature 8	0.98365	0	
Feature 6	0.96037	0	
Feature 1	0.77272	1	False Negative
Feature 4	0.63246	0	
Feature 7	0.52421	0	
Feature 5	0.09601	0	

Features are ordered based on *Velocity* rate

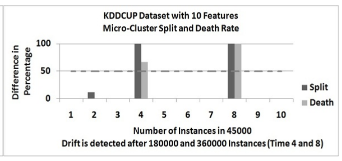
Feature	(Time 4) Velocity Rate (Descending Order)	(Time 4) History of Maximum IQR	Detection of Swapped Features (Feature 1 and 2)
Feature 2	**0.35153**	**12**	True Positive
Feature 8	0.21042	0	
Feature 3	0.20740	0	
Feature 6	0.20328	0	
Feature 4	0.16891	0	
Feature 7	0.16261	0	
Feature 1	0.09023	1	False Negative
Feature 5	0.01541	0	

Figure 49: The results of Diabetes dataset with 8 features for tracking features using Micro-Clusters.

(a) Previous MC-NN (Variance)



(b) New MC-NN (IQR)



Features are ordered based on *Velocity* rate

Feature	(Time 4) Velocity Rate (Descending Order)	(Time 4) History of Maximum Variance	Detection of Swapped Features (Features 2,3,4,8,9 and 10)
Feature 9	**0.00037**	**2**	True Positive
Feature 4	**0.00011**	**6**	True Positive
Feature 10	**0.00003**	**1**	True Positive
Feature 7	0.00002	0	
Feature 1	0.00001	2	False Positive
Feature 8	0.00001	0	False Negative
Feature 3	0.00001	0	
Feature 5	0.00001	0	
Feature 2	0.00001	0	
Feature 6	0.00000	0	

Features are ordered based on *Velocity* rate

Feature	(Time 4) Velocity Rate (Descending Order)	(Time 4) History of Maximum IQR	Detection of Swapped Features (Features 2,3,4,8,9 and 10)
Feature 9	**0.00005**	**13**	True Positive
Feature 10	**0.00002**	**1**	True Positive
Feature 7	0.00001	0	
Feature 3	**0.00001**	**76**	True Positive
Feature 4	**0.00001**	**414**	True Positive
Feature 8	0.00001	0	False Negative
Feature 1	0.00001	4	
Feature 5	0.00001	0	
Feature 2	0.00001	0	False Negative
Feature 6	0.00000	0	

Features are ordered based on *Velocity* rate

Feature	(Time 8) Velocity Rate (Descending Order)	(Time 8) History of Maximum Variance	Detection of Swapped Features (Features 2,3,4,8,9 and 10)
Feature 9	0.00012	0	False Negative
Feature 5	0.00010	0	
Feature 10	0.00006	3	False Positive*
Feature 1	0.00002	1	False Positive
Feature 4	0.00001	0	False Negative
Feature 3	0.00001	0	False Negative
Feature 7	0.00001	0	
Feature 2	0.00001	0	False Negative
Feature 6	0.00000	0	
Feature 8	0.00000	0	False Negative

Features are ordered based on *Velocity* rate

Feature	(Time 8) Velocity Rate (Descending Order)	(Time 8) History of Maximum IQR	Detection of Swapped Features (Features 2,3,4,8,9 and 10)
Feature 5	0.00007	0	
Feature 9	0.00006	45	False Positive*
Feature 10	0.00002	56	False Positive*
Feature 1	0.00001	0	
Feature 3	0.00001	0	False Negative
Feature 4	0.00001	0	False Negative
Feature 7	0.00001	0	
Feature 2	0.00001	0	False Negative
Feature 6	0.00000	0	
Feature 8	0.00000	0	False Negative

Figure 50: The results of KDDCUP dataset with 10 features for tracking features using Micro-Clusters.

on IQR. Regarding drift detection in Figures 47, 48, 49, and 50, it can be seen that the *Split* and *Death* rates at the time of concept drift increase (i.e., time when the features are swapped), indicating that the current set of Micro-Clusters does not fit the concept encoded in the data anymore and a concept drift is detected correctly for every dataset and method examined in the figures. Please note there are further concept drifts detected at different times than the features were swapped, which could be due to the fact that real data was used and thus there may be concept drifts that are not known. This is one of the reasons why the method was also evaluated on artificial datasets earlier in this section, as the ground truth for the artificial datasets was known. At this stage it is assumed that high *Split* and *Death* rates other than during the time when features were swapped are due to natural concept drift. As the ground truth for the concept drift is known only at the time of feature swapping, the evaluation of correct feature tracking was focussed on the time of feature swapping only.

Regarding the results for the *Bank* dataset in Figure 47, both methods identify a concept drift correctly at the time the features were swapped (in this case 2 features). The method based on *IQR* is superior to the method based on *Variance*, as it identified both swapped features correctly, whereas the method based on *Variance* did not identify any correctly. A similar result can be observed for the *Diabetes* dataset (see Figure 49), both methods identify a concept drift correctly at the time the features were swapped. Again the method based *IQR* is superior to the method based on *Variance*, as it identified 1 of the swapped features correctly, whereas the method based on *Variance* did not identify any correctly. Regarding the results for the *CoverType* dataset (see Figure 48), again the method based on *IQR* was superior by identifying 2 of the swapped features (in this case 4 features) correctly whereas the method based on *Variance* did not identify any correctly, because a drift was not detected. For the dataset based on *KDDCUP* (see Figure 50) 6 features were swapped and again the method based on *IQR* was superior by identifying 4 features correctly and the method based on *Variance* only identifying 3 features correctly.

These results are summarised in Table 9. It can be seen that the method based on *IQR* generally achieves a much higher number of *true positive* identification of changed features compared with the method based on *Variance*. Also the method using *IQR* has a lower *false positive* number than the method using *Variance*.

Numbers indicated with a * in Table 9 indicate potentially false positives detections of features that changed their relevance. Potentially means in this case that there was also an unexpected concept drift, hence it is not possible to verify with absolute certainty if the detection was indeed a false positive.

Table 9: Summary of the experimental results with real datasets. The results are reported for the time of drift onset (Table 3) which is the time at which features were swapped.

Dataset	True Positive (*Variance*)	True Positive (*IQR*)	False Positive (*Variance*)	False Positive (*IQR*)
Bank		2	2	
CoverType		2		1
Diabetes		1		
KDDCUP	2	4	2 + 2*	2*
Total:	2	9	6	3

Experiments have been conduced in a controlled environment on artificial datasets and in a less controlled environment on real datasets. Both methods based on *IQR* and *Variance* have been evaluated and compared on all test cases. It was observed that the method based on *IQR* identified all known concept drifts correctly whereas the method based on *Variance* did miss one known concept drift of a real data stream. Furthermore, it was found that the method based on *IQR* identified more changed features correctly compared with the method based on *Variance*. Loosely speaking the method based on *IQR* has shown to be superior to the method based on Variance in all respects. Thus the method based on *IQR* has been chosen to evaluate the real-time feature selection strategy in the next section.

7.2.3. Real-Time Feature Selection

In the previous sections the paper examined the capability of the developed method to identify concept drifts and to track whether features have undergone a significant change. The purpose was to use this method to re-examine the relevance of the features for the classification task. It has been found that the method based on *IQR* preformed best in all respects. Thus the experiments in this section are conducted with the method based on *IQR*. This section evaluates the use of the method as feature selection technique as described in Section 6. The method is applied on a couple of real datasets with a larger number of features than in the previous more controlled experiments, with no known ground truth, to evaluate the effect of the methods in real scenarios. Information about the datasets can be found in Table 4 in Section 7.1.2. It should be noted that the *CoverType* dataset has been re-used for the experiments here under uncontrolled conditions. This time all 54 features were included. It is expected that this original version of *CoverType* is more challenging for data mining algorithms to induce good models compared with the reduced version used earlier in the controlled experiments in Section 7.2.2. The data stream classifier chosen to be used with the developed method was the Hoeffding Tree [13]. The reason for choosing the Hoeffding Tree was due to its popularity and the fact that it is considered to be one of the most accurate data stream classifiers [55, 52]. The Hoeffding Tree classifier is training and updating incrementally instance by instance. From the start of the experiments all features are considered relevant, and the proposed method may at any time during the experiment detect a concept drift and flag features as relevant or irrelevant after a drift has been detected. Thus at any time during the experiment the Hoeffding Tree classifier selects only the feature values of relevant features and accordingly uses only the currently relevant features for training and incrementally updating the model. Figure 51 shows the accuracy differences over time the method achieved using *IQR* for real-time feature selection in comparison with applying the Hoeffding Tree classifier as standalone method. As it can be seen in the figure, Hoeffding Trees using the proposed method generally achieved a better accuracy over time. Only during a few time windows the method achieved a marginally lower accuracy.

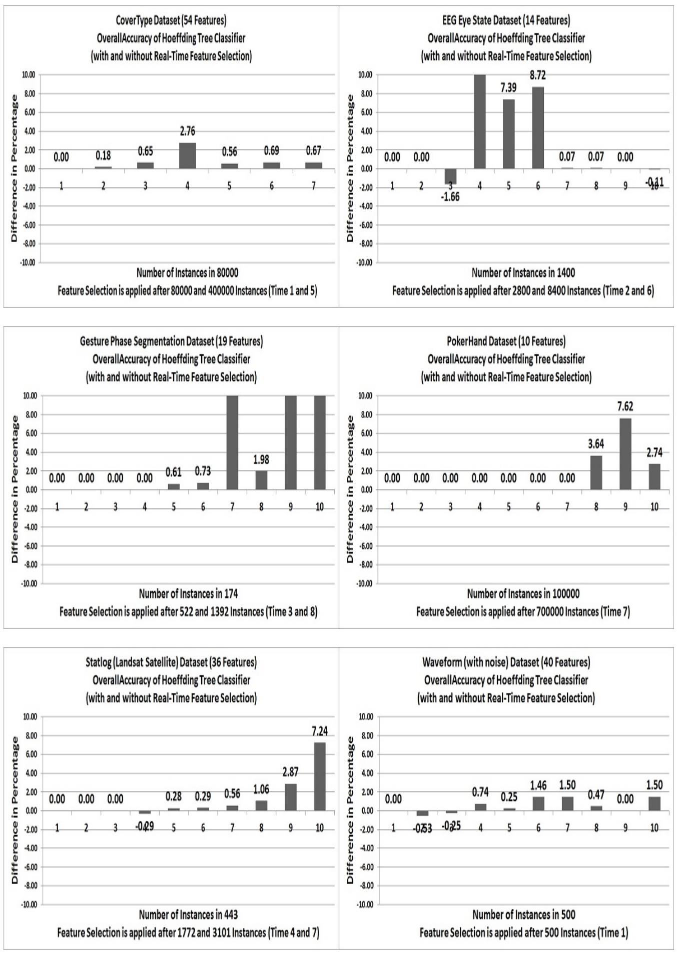


Figure 51: The results of real-time feature selection with uncontrolled datasets using Micro-Clusters.

Table 10 summarises the results for the experiments depicted in Figure 51. It states the average accuracy achieved with and without using the proposed real-time feature selection method. It also indicates at which time window features and how many features have been re-evaluated for inclusion or removal from the current feature set considered by the Hoeffding Tree. As can be seen the method actively re-evaluated features for inclusion in the tree at various times and achieved on average a higher accuracy compared with standalone Hoeffding Trees (not employing the proposed real-time feature selection method).

Table 10: Summary of the results for the experiments using real datasets with Hoeffding tree classifier.

Real Dataset	OverallAccuracy Average (with Real-Time Feature Selection)	OverallAccuracy Average (without Real-Time Feature Selection)	Time of Applying Real-Time Feature Selection	Number of Features Flagged
CoverType	**79.81**	79.19	(2 to 4) and (6 to 7)	1
EEG Eye State	**83.68**	81.49	(3 to 5) and (7 to 10)	1
Gesture Phase Segmentation	**80.23**	72.18	(4 to 7) and (9 to 10)	1
Poker Hand	**51.75**	51.02	8 and 10	5
Statlog (Landsat Satellite)	**81.63**	80.68	(5 to 6) and (8 to 10)	5
Waveform (with noise)	**80.68**	80.26	2 to 10	1

Loosely speaking the results show that the proposed real-time feature selection method indeed improves the accuracy of data stream classifiers.

8. Discussion and Conclusions

This paper has investigated the problem of real-time feature selection. At present the focus of data stream mining lies in the development of data mining algorithms rather than on pre-processing methods. Thus at present there are no developments for truly real-time feature selection given a data streaming input space. This is important as features may potentially change their relevance for data mining tasks based on certain measures of relevance such as Information Gain. Thus the three objectives of this paper were to develop a real-time pre-processing method that could (a) detect a concept drift, (b) identify features that were involved in a concept drift and thus potentially changed their relevance and (c) build and validate a real-time feature selection method based on the aforementioned developments. The starting point for this research was the MC-NN classifier. This research was not concerned with the classification capabilities of MC-NN but in the behaviour of its underlying model during concept drift. The MC-NN model is based on adaptive statistical Micro-Cluster summaries of the absorbed data stream instances that can *Split* into new Micro-Clusters, change their size/position in the feature space or be removed (Micro-Cluster death) in order to adapt to concept drift. The research proposed in this work was based on two hypotheses on the behaviour of MC-NN in this regard:

(1) The *Split* and *Death* rates are expected to increase during a concept drift and thus could be used as a measure to detect concept drift in real-time; and (2) the direction of the Micro-Cluster movement in feature space during concept drift and their *Velocity* indicate which features are involved in the concept drift. This could be used as a measure if the relevance of a feature for a data mining task has changed.

MC-NN originally used variance as a statistical measure to *Split* the Micro-Cluster. During this research it was expected that variance as an indirect measure for concept drift adaptation would be susceptible to potential outliers and noise. Thus an alternative method has been explored for adapting micro-clusters based on *IQR*. Both methods have been evaluated with respect to hypothesis 1 and 2 on artificial data streams and real datasets. In addition for both methods a Low Pass Filter (*LPF*) was also incorporated to filter out noise and normalisation to reduce feature bias. The original MC-NN did not make use of *LPF* or normalisation. The evaluation of the proposed method initially focused on the approach based on the already available MC-NN statistical measure, namely Variance. Thus the method was evaluated on artificial data streams and real datasets with artificially induced concept drifts. It was observed that the method did detect concept drifts very for the artificial data streams compared with competing alternative concept drift detection methods. It achieved a very high *true positive* detection number and resulted in only one *false positive* detection. For the real datasets the method detected all but one artificially induced concept drift correctly but also detected further concept drifts. These drifts may have been natural and thus previously unknown concept drifts. Thus in the next step of the evaluation *LPF*, *Normalisation* and *IQR* were used as an alternative measure for feature splitting and concept drift detection. To allow a fair comparison, *LPF* and Normalisation were applied on both methods based on variance and the method based on *IQR*. Both versions of the method were compared with regards to concept drift detection and feature tracking for the artificial data streams and the real datasets (with induced concept drift). For the artificial data streams all induced concept drifts were detected correctly by both versions of the method and for the real datasets one concept drift was missed but only by the method based on variance. Regarding feature tracking, the method based on *IQR* clearly outperformed the method based on Variance. The method based on *IQR* achieved a high true positive identification of features that were actually features involved in concept drift and a low number of *false positive* identifications. Thus the method based on variance was considered not suitable for feature tracking and the next step of the evaluation focussed only on the method based on *IQR* for feature selection. Here the method for feature selection was tested based on feature tracking using *IQR*. The evaluation was conducted in an uncontrolled environment on 6 case studies with data streams based on real datasets. Thus the ground truth for feature relevance and concept drift were not known. Hence the impact of the method on the classification accuracy over time was measured and also the extent to which the method actually identified features with changing relevance. The data stream classification method chosen was the popular Hoeffding Tree algorithm. The results showed that the method detected various concept drifts throughout the streams and identified features for re-evaluation for their relevance. It was also shown that the classifier achieved a higher average accuracy when using the proposed method compared with not using the method. Overall the research represents a first attempt to resolve real-time feature selection for data stream mining tasks. It has been shown that the method can indeed identify concept drift, track features and identify features that may have changed their relevance for the data mining task in real-time. It has also been shown that the proposed method can improve the accuracy of data stream classification tasks.

References

[1] M. Ebbers, A. Abdel-Gayed, V. B. Budhi, F. Dolot, V. Kamat, R. Picone, J. Trevelin, et al., Addressing Data Volume, Velocity, and Variety with IBM InfoSphere Streams V3. 0, IBM Redbooks, 2013.

[2] B. Babcock, S. Babu, M. Datar, R. Motwani, J. Widom, Models and issues in data stream systems, in: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, ACM, 2002, pp. 1–16.

[3] P. Kadlec, B. Gabrys, S. Strandt, Data-driven soft sensors in the process industry, Computers & Chemical Engineering 33 (4) (2009) 795–814.

[4] A. Jadhav, P. Jadhav, P. Kulkarni, A novel approach for the design of network intrusion detection system (nids), in: Sensor Network Security Technology and Privacy Communication System (SNS & PCS), 2013 International Conference on, IEEE, 2013, pp. 22–27.

[5] G. Hulten, L. Spencer, P. Domingos, Mining time-changing data streams, in: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2001, pp. 97–106.

[6] T. Le, F. Stahl, J. B. Gomes, M. M. Gaber, G. Di Fatta, Computationally efficient rule-based classification for continuous streaming data, in: Research and Development in Intelligent Systems XXXI, Springer, 2014, pp. 21–34.

[7] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Computing Surveys (CSUR) 46 (4) (2014) 44.

[8] A. Bifet, R. Gavalda, Learning from time-changing data with adaptive windowing., in: SDM, Vol. 7, SIAM, 2007, p. 2007.

[9] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Advances in artificial intelligence–SBIA 2004, Springer, 2004, pp. 286–295.

[10] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, R. Morales-Bueno, Early drift detection method, in: Fourth international workshop on knowledge discovery from data streams, Vol. 6, 2006, pp. 77–86.

[11] G. Lee, A. Singanamalli, H. Wang, M. D. Feldman, S. R. Master, N. N. Shih, E. Spangler, T. Rebbeck, J. E. Tomaszewski, A. Madabhushi, Supervised multi-view canonical correlation analysis (smvcca): integrating histologic and proteomic features for predicting recurrent prostate cancer, IEEE transactions on medical imaging 34 (1) (2015) 284–297.

[12] U. Ahsan, I. Essa, Clustering social event images using kernel canonical correlation analysis, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2014, pp. 800–805.

[13] P. Domingos, G. Hulten, Mining high-speed data streams, in: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2000, pp. 71–80.

[14] M. Hammoodi, F. Stahl, M. Tennant, A. Badii, Towards realtime feature tracking technique using adaptive micro-clusters, in: Proceedings of the BCS SGAI Workshop on Data Stream Mining Techniques and Applications, 2016.

[15] R. J. C. Bose, W. M. Van Der Aalst, I. Žliobaitė, M. Pechenizkiy, Dealing with concept drifts in process mining, IEEE transactions on neural networks and learning systems 25 (1) (2014) 154–171.

[16] D. Brzeziński, Mining data streams with concept drift, Ph.D. thesis, Masters thesis, Poznan University of Technology (2010).

[17] C. C. Aggarwal, P. S. Yu, Outlier detection for high dimensional data, in: ACM Sigmod Record, Vol. 30, ACM, 2001, pp. 37–46.

[18] E. Page, Continuous inspection schemes, Biometrika 41 (1/2) (1954) 100–115.

[19] P. B. Dongre, L. G. Malik, Stream data classification and adapting to gradual concept drift, International Journal 2 (3).

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

[21] J. Tang, S. Alelyani, H. Liu, Feature selection for classification: A review, Data Classification: Algorithms and Applications (2014) 37.

[22] D. Lavanya, D. K. U. Rani, Analysis of feature selection with classification: Breast cancer datasets, Indian Journal of Computer Science and Engineering (IJCSE) 2 (5) (2011) 756–763.

[23] J. Han, J. Pei, M. Kamber, Data mining: concepts and techniques, Elsevier, 2011.

[24] W. Yi, F. Teng, J. Xu, Noval stream data mining framework under the background of big data, Cybernetics and Information Technologies 16 (5) (2016) 69–77.

[25] M. E. Houle, M. Nett, Rank-based similarity search: Reducing the dimensional dependence, IEEE transactions on pattern analysis and machine intelligence 37 (1) (2015) 136–150.

[26] H. Loo, M. Marsono, Online data stream classification with incremental semi-supervised learning, in: Proceedings of the Second ACM IKDD Conference on Data Sciences, ACM, 2015, pp. 132–133.

[27] J. Shao, Z. Ahmadi, S. Kramer, Prototype-based learning on concept-drifting data streams, in: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2014, pp. 412–421.

[28] Y.-N. Law, C. Zaniolo, An adaptive nearest neighbor classification algorithm for data streams, in: Knowledge Discovery in Databases: PKDD 2005, Springer, 2005, pp. 108–120.

[29] H. Wang, W. Fan, P. S. Yu, J. Han, Mining concept-drifting data streams using ensemble classifiers, in: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2003, pp. 226–235.

[30] J. Gama, P. Kosina, et al., Learning decision rules from data streams, in: IJCAI Proceedings-International Joint Conference on Artificial Intelligence, Vol. 22, 2011, p. 1255.

[31] M. A. Khan, A. Khan, M. N. Khan, S. Anwar, A novel learning method to classify data streams in the internet of things, in: Software Engineering Conference (NSEC), 2014 National, IEEE, 2014, pp. 61–66.

[32] J. P. Barddal, H. M. Gomes, F. Enembreck, Sfnclassifier: a scale-free social network method to handle concept drift, in: Proceedings of the 29th Annual ACM Symposium on Applied Computing, ACM, 2014, pp. 786–791.

[33] T. Zhang, R. Ramakrishnan, M. Livny, Birch: an efficient data clustering method for very large databases, in: ACM Sigmod Record, Vol. 25, ACM, 1996, pp. 103–114.

[34] C. C. Aggarwal, J. Han, J. Wang, P. S. Yu, A framework for clustering evolving data streams, in: Proceedings of the 29th international conference on Very large data bases-Volume 29, VLDB Endowment, 2003, pp. 81–92.

[35] C. C. Aggarwal, J. Han, J. Wang, P. S. Yu, A framework for projected clustering of high dimensional data streams, in: Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, VLDB Endowment, 2004, pp. 852–863.

[36] K. Udommanetanakit, T. Rakthanmanon, K. Waiyamai, E-stream: Evolution-based technique for stream clustering, in: Advanced Data Mining and Applications, Springer, 2007, pp. 605–615.

[37] P. Kranen, I. Assent, C. Baldauf, T. Seidl, Self-adaptive anytime stream clustering, in: Data Mining, 2009. ICDM'09. Ninth IEEE International Conference on, IEEE, 2009, pp. 249–258.

[38] W. Meesuksabai, T. Kangkachit, K. Waiyamai, Hue-stream: Evolution-based clustering technique for heterogeneous data streams with uncertainty, in: Advanced Data Mining and Applications, Springer, 2011, pp. 27–40.

[39] M. Tennant, F. Stahl, O. Rana, J. B. Gomes, Scalable real-time classification of data streams with concept drift, Future Generation Computer Systems.

[40] E. Ogasawara, L. C. Martinez, D. De Oliveira, G. Zimbrão, G. L. Pappa, M. Mattoso, Adaptive normalization: A novel data normalization approach for non-stationary time series, in: The 2010 International Joint Conference on Neural Networks (IJCNN), IEEE, 2010, pp. 1–8.

[41] O. Schall, A. Belyaev, H.-P. Seidel, Robust filtering of noisy scattered point data, in: Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005., IEEE, 2005, pp. 71–144.

[42] R. E. Rosenholtz, A. Zakhor, Iterative procedures for reduction of blocking effects in transform image coding, in: Electronic Imaging'91, San Jose, CA, International Society for Optics and Photonics, 1991, pp. 116–126.

[43] M. S. Arulampalam, S. Maskell, N. Gordon, T. Clapp, A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking, IEEE Transactions on signal processing 50 (2) (2002) 174–188.

[44] L. Sunitha, M. BalRaju, J. Sasikiran, E. V. Ramana, Automatic outlier identification in data mining using iqr in real-time data, International Journal of Advanced Research in Computer and Communication Engineering 3 (6) (2014) 7255–7257.

[45] C. Leys, C. Ley, O. Klein, P. Bernard, L. Licata, Detecting outliers: Do not use standard deviation around the mean, use absolute deviation around the median, Journal of Experimental Social Psychology 49 (4) (2013) 764–766.

[46] D. De Gregorio, L. Di Stefano, Skimap: An efficient mapping framework for robot navigation, arXiv preprint arXiv:1704.05832.

[47] N. Shavit, I. Lotan, Skiplist-based concurrent priority queues, in: Parallel and Distributed Processing Symposium, 2000. IPDPS 2000. Proceedings. 14th International, IEEE, 2000, pp. 263–268.

[48] Y.-C. Hu, A. Perrig, D. B. Johnson, Efficient security mechanisms for routing protocolsa., in: NDSS, 2003.

[49] A. Bifet, G. Holmes, R. Kirkby, B. Pfahringer, Moa: Massive online analysis, Journal of Machine Learning Research 11 (May) (2010) 1601–1604.

[50] W. N. Street, Y. Kim, A streaming ensemble algorithm (sea) for large-scale classification, in: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2001, pp. 377–382.

[51] D. Brzezinski, J. Stefanowski, Reacting to different types of concept drift: The accuracy updated ensemble algorithm, IEEE Transactions on Neural Networks and Learning Systems 25 (1) (2014) 81–94.

[52] A. Bifet, E. Frank, Sentiment knowledge discovery in twitter streaming data, in: International conference on discovery science, Springer, 2010, pp. 1–15.

[53] M. Lichman, UCI machine learning repository (2013).
URL `http://archive.ics.uci.edu/ml`

[54] D. Marrón, J. Read, A. Bifet, N. Navarro, Data stream classification using random feature functions and novel method combinations, Journal of Systems and Software 127 (2017) 195–204.

[55] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, R. Gavaldà, New ensemble methods for evolving data streams, in: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, ACM, 2009, pp. 139–148.