Thesis no: MGCS-2014-05

A Boosted-Window Ensemble

Haroon Elahi

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona Sweden

This thesis is submitted to Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Computer Science. The thesis is equivalent to 10 weeks of full time studies.

Contact Information:
Author:
Haroon Elahi
E-mail: hael13@student.bth.se

University advisor:
Dr. Niklas Lavesson
Associate Professor of Computer Science
Dept. Computer Science & Engineering
E-mail: niklas.lavesson@bth.se

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57

ABSTRACT

Context: The problem of obtaining predictions from stream data involves training on the labeled instances and suggesting the class values for the unseen stream instances. The nature of the data-stream environments makes this task complicated. The large number of instances, the possibility of changes in the data distribution, presence of noise and drifting concepts are just some of the factors that add complexity to the problem. Various supervised-learning algorithms have been designed by putting together efficient data-sampling, ensemble-learning, and incremental-learning methods. The performance of the algorithm is dependent on the chosen methods. This leaves an opportunity to design new supervised-learning algorithms by using different combinations of constructing methods.

Objectives: This thesis work proposes a fast and accurate supervised-learning algorithm for performing predictions on the data-streams. This algorithm is called as Boosted-Window Ensemble (BWE), which is invented using the mixture-of-experts technique. BWE uses Sliding Window, Online Boosting and incremental-learning for data-sampling, ensemble-learning, and maintaining a consistent state with the current stream data, respectively. In this regard, a sliding window method is introduced. This method uses partial-updates for sliding the window on the data-stream and is called Partially-Updating Sliding Window (PUSW). The investigation is carried out to compare two variants of sliding window and three different ensemble-learning methods for choosing the superior methods.

Methods: The thesis uses experimentation approach for evaluating the Boosted-Window Ensemble (BWE). CPU-time and the Prediction accuracy are used as performance indicators, where CPU-time is the execution time in seconds. The benchmark algorithms include: Accuracy-Updated Ensemble1 (AUE1), Accuracy-Updated Ensemble2 (AUE2), and Accuracy-Weighted Ensemble (AWE). The experiments use nine synthetic and five real-world datasets for generating performance estimates. The Asymptotic Friedman test and the Wilcoxon Signed-Rank test are used for hypothesis testing. The Wilcoxon-Nemenyi-McDonald-Thompson test is used for performing post-hoc analysis.

Results: Hypothesis testing suggests that: 1) both for the synthetic and real-world datasets, the Boosted Window Ensemble (BWE) returns significantly lower CPU-time values than two benchmark algorithms i.e., AUE1 and AWE. 2) BWE returns lower CPU-time than AUE2. However, the difference is not significant.3) BWE returns comparable prediction accuracy as AUE1 and AWE for synthetic and real-world datasets. 3) AUE2 returns better prediction accuracy than BWE in almost all cases.

Conclusions: Experimental results show that the use of Partially-Updating Sliding Window (PUSW) has resulted in lower CPU-time for BWE as compared with the chunk-based sliding window method used in AUE1, AUE2, and AWE. The results further demonstrate that the proposed algorithm can be as accurate as the state-of-the-art benchmark algorithms in most of the cases, while obtaining predictions from the stream data.

Keywords: Stream Mining, Supervised-learning by classification, Online learning algorithms, Ensemble Methods, Boosting

ACKNOWLEDGEMENTS

I would like to use this opportunity to express my special gratitude to everyone who supported me throughout the project.

I would especially like to thank my supervisor Dr. Niklas Lavesson for his aspiring guidance, invaluably constructive criticism, and non-stop feedback on the project work.

I would like to thank my brothers and my friends for their moral support throughout my master studies as well as in this project.

In the end, I am earnestly thankful to my parents and to BTH for raising me up in special ways.

CONTENTS

1 INTRODUCTION

Machine learning has made rapid progress in recent years in terms of its capabilities to solve real-world problems (Gama et al. 2013; Gaber et al. 2005; Golab et al. 2003). However, it is facing new challenges posed due to enormous growth in data volumes (Joshi and Kulkarni 2012; Zliobaite et al. 2012). This data-growth has resulted from the recent technological advances in hardware and software. As a result of these advances, it is possible to electronically capture the data in many application-domains where it was impossible earlier. Stream mining is a branch of machine learning that deals with extracting knowledge structures, represented in the models and patterns in the data-streams (Gaber et al. 2005). Stream mining solutions address the complexities introduced by different attributes of the data-streams. Large number of instances, changes in the data distribution, presence of noise and drifting concepts are some of the streaming data attributes that add complexity to the stream mining problems (Domingos et al. 2000; Domingos et al. 2001; Golab et al. 2003; Gama, João 2012; Wankhade et al., 2013; Yi Yang and Guojun Mao 2013; Bolchini et al. 2013). One of the frequent problems solved by the stream mining algorithms is obtaining predictions from the stream data. This problem involves training the algorithm on the labeled instances and suggesting the class values for unseen stream instances (Bifet et al. 2013; Wankhade et al. 2013). Many online algorithms have been designed by putting together efficient data-sampling, ensemble-learning, and incremental-learning methods. The challenge lies in designing stream mining algorithms that characterize low memory requirements, self-adaption, higher classification accuracy and lower CPU-time utilization (Li and Liu 2008; Chen et al. 2009; Kholghi et al. 2010; Wankhade et al., 2013; Yi Yang and Guojun Mao., 2013; Bifet et al. 2013).

This thesis work proposes a new algorithm, called Boosted-Window Ensemble (BWE), which would improve over state-of-the-art algorithms on CPU-time, while still keeping good classification accuracy. Experimental results are offered comparing the performance of the proposed algorithm against the related state-of-the-art benchmark algorithms.

2 BACK GROUND

Contrary to the traditional databases, data-streams can be characterized by being continuous, unbounded, time varying, and having high speed underlying data generation processes (Li and Liu 2008; Shie et al. 2012). The individual data items in the data-stream are termed as examples or instances. An instance of a data-stream consists of different attributes. The last attribute is usually termed as the class-attribute (Vilalta et al. 2002). The process of discovering knowledge from data-streams is termed as stream mining. Stream mining solutions generally perform clustering or prediction tasks (Ade et al. 2013; Wankhade et al., 2013). Prediction involves, suggesting the class value of an unseen data instance assuming the prior knowledge learned from the training-data. The supervised-learning algorithms are used for predicting the unseen stream data. Due to the specifics of the nature of data-streams, such algorithms deal with challenges that are different from those met during the conventional machine learning (Gama et al. 2013). Generally, the limited amount of memory usage, fast speed, and high prediction accuracy are used for measuring the performance of these algorithms (Zliobaite et al., 2012). It is, however, impossible to invent such algorithms that are optimal in all these efficiency indicators. A trade-off has to be made among these efficiency indicators for reaching an overall-optimized state of the algorithm (Wolpert et al. 1997).

Meta-learning is a sub-domain of machine learning that addresses most of the challenges raised in stream mining. Meta-learning methods such as incremental-learning, algorithmic-adaptation, ensemble-methods, parameter-tuning, cost-efficiency, and concept-drift handling have been the subject of stream mining research in recent years (Hansen and Jakob 1999; Kholghi et al. 2010; Hoens et al. 2012; Joshi et al. 2012; Ade et al. 2013; Bifet et al. 2013; Gama et al. 2013; Pechenizkiy and Zliobaite 2013).

A variety of techniques have been used for inventing stream mining algorithms (Hansen J.P. 1999; Gaber et al. 2005; Li and Liu 2008; Yang and Fong 2012; Bifet et al. 2013; Bolchini et al. 2013; Wankhade et al. 2013). For designing an efficient stream mining algorithm, the constraints of limited memory, CPU-time, and single-pass data have to be dealt with, effectively. Several incremental-learning algorithms have been proposed. Incremental-learning algorithms discard old model and build an up-to-date model learnt from the recent data (Masud et al. 2010; Joshi and Kulkarni 2012; Pechenizkiy and Zliobaite 2013). Sliding window technique is frequently used for efficiently sampling the data in the stream mining algorithms. Sliding-Window slides on the data-stream by discarding the old instances and receiving more recent instances. Different approaches have been proposed for implementing the sliding window method (Oza, N. C. 2005; Ferreira and António 2009; Kapp et al. 2011; Chen et al. 2012; Bifet et al. 2013; Braverman et al. 2012; Yang and Mao 2013).

Efficient data-sampling is not the only challenge faced by the stream mining algorithms. The possibility of runtime changes in the data distribution and the definition of class attributes can introduce a certain level of complexity. These changes can result from changes in the stream mining environment or can emanate from within the algorithm (Bouchachia and Nedjah 2012). Substantial efforts are needed to adapt the stream mining algorithms to their environment (Kappet al. 2011). Ensemble methods are a help in this regard. Such methods help the learning algorithms in adapting to their environments, through controlling the desired amount of bias at run-time using different techniques (Vilalta and Drissi 2002).

Mixture-of-experts is another meta-learning technique that advocates the use of diverse expert techniques in parallel. It is assumed that the techniques which are individually efficient can help in inventing algorithms that will efficiently solve computationally-complex problems (Masoudnia and Ebrahimpour 2012). Some of the recent supervised-learning algorithms implementing the meta-learning techniques include: Accuracy-Weighted Ensemble, Online Bagging and Boosting, and Accuracy-Updated Ensemble (Oza, N. C. 2005; Brzeziński and Stefanowski 2011; Brzezinski and Stefanowski 2014). This thesis uses meta-learning techniques including: Sliding Window, ensemble-learning, incremental-learning, and mixture-of-experts for inventing an efficient supervised-learning algorithm.

3 AIMS AND OBJECTIVES

The aim of conducting this thesis work is to invent an efficient supervised-learning algorithm for data-stream mining. This thesis uses mixture-of-experts technique to achieve the aim. Meta-learning techniques including: Sliding Window, ensemble-learning, and incremental-learning are investigated to meet the purpose. The study intends to propose a Partially-Updating Sliding Window method to be used for designing the new algorithm.

Following section describes the research questions used for carrying out the investigation.

3.1 Research Questions

The research questions in this thesis work are as follows:

RQ1: What is the impact of using *Partially-Updating Sliding Window* on the *CPU-time* (execution time in seconds) of the underlying stream mining algorithm?

This research question investigates whether a faster algorithm can be designed using Partially-Updating Sliding Window approach. An alternative approach is chunk-based sliding window that reinitializes with each incremental update by forgetting all old instances and replacing them with more recent instances.

RQ2: What is the impact of using *Online Boosting* instead of the *Accuracy-Updated Ensemble*, on the *prediction accuracy* of the underlying stream mining algorithm?

Accuracy-Updated Ensemble is an ensemble-learning method that has been used in recently proposed online supervised-learning algorithms (Brzeziński and Stefanowski 2011; 2014). This research question investigates whether the use of Online Boosting, instead of Accuracy-Updated Ensemble has an impact on the prediction accuracy of an underlying learning algorithm.

RQ3: What is the impact of using *Online Boosting* instead of the *Accuracy-Weighted Ensemble* on the *prediction accuracy* of the underlying stream mining algorithm?

Accuracy-Weighted Ensemble is an ensemble-learning method that uses weights derived by estimating the expected prediction error of a classifier on the test instances (Wang et al. 2003). This research question investigates whether the use of Online Boosting, instead of Accuracy-Weighted Ensemble has an impact on the prediction accuracy of an online-learning algorithm.

3.2 Contribution

This thesis work reports the impact of the use of different data-sampling and ensemble-learning methods on the performance of supervised-learning algorithms. The study introduces a fixed-size, partially-updating sliding window method that is faster than the chunk-based sliding window method. A fast and accurate supervised-learning algorithm is proposed using this new sliding window method. As the problem of obtaining time-efficient predictions from the stream data is quite frequent, the proposed algorithm can be used in environments where execution speed is a high priority.

It is expected that additional efficient algorithms can be invented by using the proposed sliding window method.

4 RELATED WORK

A variety of techniques have been used for designing efficient stream mining algorithms (Hansen J.P. 1999; Gaber et al. 2005; Li and Liu 2008; Masud et al. 2010; Yang and Fong 2012; Bolchini et al. 2013; Wankhade et al. 2013; Bifet et al. 2013). While designing stream mining algorithm, among other, constrained memory, and CPU-time, frequent changes in the data and the inability to revisit the stream instances (single-pass) has to be dealt with, effectively. Efficient data-sampling methods, adaptation techniques, ensemble-learning methods, and incremental-learning are being used to deal with these issues.

The sliding window technique has frequently been used in stream mining environments for efficient data-sampling. Wang et al. (2003), proposed Accuracy-weighted Ensemble algorithm for stream classification that uses a chunk-based sliding window method. These chunks are re-initialized periodically by forgetting current data items and replacing them with fresh instances from the stream. Deypir et al. (2012) proposed a variable size sliding window algorithm for data-streams. Their algorithm continuously monitors the amount of change in the set of frequent patterns in stream data. The sliding window used in their solution adjusts its size in response to the observed amount of change within the incoming data-stream. For measuring the change in the stream data, they use a set of frequent patterns from the given stream that they update after each inserted pane of transactions. Deypir and Sadreddini (2012) proposed a sliding window based method and termed it as LDS. This method used a pane-based, fixed-size sliding window. This window uses three types of lists for controlling its memory usage. The adjustment is made in the window after each sliding. This method claims to have low processing-time and memory requirements. In order to forget the old data, it looks at tails of the three lists and cuts information related to outdated data from the lists. Chen et al. (2012) proposed a method for mining frequent patterns in stream data. This method uses a varying-size sliding window. The window, incrementally maintains, the contents of newly generated stream data by scanning the stream only once. It uses a decay factor for forgetting the old patterns by gradually reducing their frequencies. Braverman et al. (2012) proposed a memory-optimal method for sampling with and without replacement from fixed-size or timestamp-based windows. Bifet et al. (2013) proposed probabilistic approximate window (PAW) algorithm for performing classification on data-streams. PAW keeps a sketch window, using only a logarithmic number of instances, storing the most recent ones with higher probability. Yang and Mao (2013) proposed a self-adaptive sliding window model. This model sets the size of the Sliding Window by learning the window control parameters. This model uses an evaluation function to forget a variable number of oldest data items and keeping the most recent ones. Brzezinski and Stefanowski (2014) proposed Accuracy-Updated Ensemble1 (AUE1), and Accuracy-Updated

Ensemble2 (AUE2), which use a chunk-based sliding window method for data-sampling.

In the sliding window based algorithms, there is a probability of reduced prediction accuracy due to the frequent change in the learning dataset (Bifet et al. 2013). Ensemble methods are used to encounter this problem. Boosting is an ensemble method that is used for consistently improving the performance of a single learner in data-stream environments (García et al. 2014). AdaBoost (Adaptive Boosting) algorithm is a widely used Boosting algorithm (Freund and Schapire 1996). AdaBoost adjusts the distribution weights on the training instances, according to the performance of the previous classifiers in the ensemble. OzaBoost is another algorithm that implements an Online Boosting technique. Ozaboost uses Poisson distribution parameter (λ) values for implementing adaptation (Oza, N. C. 2005). The ensemble methods use base-learners for learning and decision making. The use of decision trees as a base-learner is closely related with their easily interpretable modeling capabilities. Very Fast Decision Tree (VFDT) is one of the earliest stream mining algorithms that implement ensemble methods and incremental-learning. VFDT was designed to obtain optimal memory and time usage (Domingos and Hulten 2000).VFDT builds decision trees using constant memory and constant time per example and uses Hoeffding-bounds to guarantee the quality of its output. The algorithm was further improved to handle the high data rate and concept-drift (Domingos and Hulten 2001).

5 PROPOSED MODEL

This thesis work proposes a supervised-learning algorithm for obtaining predictions from the data-stream instances. This algorithm is named as Boosted-Window Ensemble (BWE). A partially-updating sliding window is introduced for efficient data-sampling from the data-stream. Furthermore, BWE uses the Online Boosting method proposed by Oza, N. C. (2005), for ensemble-learning. The functioning of the model is explained underneath.

5.1 Partially-Updating Sliding Window (PUSW)

The Partially-Updating Sliding Window is introduced for efficient sampling of data from the data-stream. BWE creates a window (W) of size N on the data-stream S. During incremental-learning, the window slides forward one the data-stream by discarding n instances and receiving n new instances, in each increment. The value of n is defined by the user. Recommended values of n are: n=N/2 or n=N/4. The motivation for introducing the partial-updates is to offer a good mix of learning instances in the window.

The window slides forward in a first-in first-out (FIFO) fashion such that if n is 2; the first two instances at the tail are discarded, and the two new instances are added at the head of the window. The sliding of the window is controlled by a parameter k. k defines the intervals at which the window slides forward. In the proposed model, the number of predicted instances in the stream defines k. If the value of k is 1500, the window slides after BWE has finished obtaining predictions from 1500 instances of the stream.

```
Input : S = {s_1, s_2, ..., ∞}
        N: size of the sliding window
Output : sliding window of size N
do  Create a new window
    if (W==null)
        for i = 1 to N
            add s_i to W
        end for
    end if
```

Figure 1-Pseudo Code for Sliding Window

5.2 Ensemble-learning

The proposed model uses Online Boosting method for ensemble-learning. During ensemble-learning, the base learner generates base models on the partially-updating Sliding Window. Weights are allocated to the training instances using Poisson distribution. Poisson's probability distribution is often used to characterize the

statistics of rare events whose average number is small (Thompson, W. J. 2001). The Poisson distribution parameter (λ) associated with an instance is raised if the base model miss-classifies it otherwise it is decreased. It is replicated for all training examples. Hence, when the training set will be used for building the next model, the associated Poisson distribution parameter (λ) values can differ from what they were while building the previous model. Just like AdaBoost and OzaBoost, the proposed algorithm assigns half of the total weight to the instances miss-classified while building the previous model (Oza, N. C. 2005). The correctly classified instances get the remaining half of the weight. Each model in the ensemble is assigned an error value depending upon the number of instances correctly classified and miss-classified by that base model. Thus the ensemble continuously tries to improve its prediction accuracy. A majority-vote method is used for making predictions on unseen instances of the stream.

```
1: Input : sliding window of size N
2: Output : predictions results from ϵ using majority vote
3: do Learn an esemble ϵ on W
4:         for i = 1 to M
5:             learn model h(i) on W using L
6:                 for i = 1 to N
7:                     assign Poisson distribution weight to s_i
8:                     learn from instance s_i
9:                     test s_i on W
10:                end for
11:                adjust weights for h(i+1)
12:        end for
13:        return predictions results from ϵ using majority vote
14: do Make X predictions on stream instances
15:        for i = 1 to X
16:            return predictions results from ϵ using majority vote
17:        end for
```

Figure 2- Pseudo Code for Ensemble-learning

5.3 Base learner

The proposed model uses Hoeffding tree as the base learner in the ensemble. Hoeffding tree has been selected due to its fixed memory requirement and incremental-learning capabilities. Furthermore, the use of the Hoeffding bounds guarantees that the selected split is really the best one (Domingos and Hulten 2000).

5.4 Incremental-learning

To keep learning consistent with the recent instances of the stream, the algorithm updates itself continuously over time. Updates are performed at predefined intervals. The proposed model defines these intervals in terms of the number of predicted instances. For example, if the value of this parameter is 1000, the algorithm will

update the sliding window after predicting every 1000 instances. The ensemble is learned from the current window for updating the base models.

5.5 Discussion

The proposed supervised-learning algorithm uses a fixed-size sliding window similar to AUE1, AUE2, and AWE for the efficient data-sampling from the data-stream (Wang et al. 2003; Brzezinski and Stefanowski 2014). In contrary to AUE1, AUE2, and AWE, where the default window size is 500, the proposed model uses the window size of 1000 instances.

```
Input : Sliding window of size N
        The interval parameter k
        S = {s_1, s_2, ..., ∞}
Output : Esemble ϵ of size M
        Sliding window of size N
do  Update Learning
     Update window by k instances
        if (W!==null)
           for i = 1 to k
              remove, first instance from W
              add, a new instance to W
           end for
        end if
     Learn esemble ϵ on W
do  Make X predictions on stream instances
        for i = 1 to X
            predictions results from h(i)
        end for
repeat Update Learning
```

Figure 3- Pseudo Code for Incremental-learning

A commonly used approach for sliding the window on the data-stream is to forget old examples at constant time intervals (Wang et al. 2003). Similar approach is followed by AUE1, AUE2, and AWE, where the window is re-initialized with each incremental-learning cycle. This approach has a side effect. The model following this approach forgets even relevant information. Different approaches have been used to counter this problem, including: maintaining a store of historical models as in (Wang et al. 2003; Brzezinski and Stefanowski 2014) or by using probabilistic methods to forecast the likelihood that an instance shall be needed in the future (Bifet et al. 2013). The proposed algorithm uses partial updates to help the model in maintaining the historical information for a longer period of time. Furthermore, the proposed algorithm uses instance-intervals instead of time-intervals. The use of partial updates and instance-intervals helps in providing a good mix of training instances.

Ensemble-learning has been applied to improve the prediction capabilities. The proposed algorithm learns an ensemble on the sliding window instances using Online Boosting method (Oza, N. C. 2005). The default ensemble-size used in the proposed model is 10. The ensemble is re-learnt from the current partially-updating Sliding Window. During ensemble-learning, weights are allocated to the training instances using Poisson distribution co-efficient. When a base model misclassifies a training example, the Poisson distribution parameter (weight), associated with that particular instance is increased for the next base model. Keeping track of the total weights of each base model's correctly classified and miss-classified training instances does this. These weights are used to update each base model's error (Oza, N. C. 2005).

5.6 Comparison of Learning Behavior

Decision models in the stream-mining algorithms evolve over time (Gama et al. 2013). This implies that the models used for obtaining the predictions from the first and the nth instances are different. A learning curve can be used to graphically visualize the progress in the learning behavior of an algorithm (Duane, J. T. 1964; Meir and Fontanari 1992; Gama et al. 2013). As the proposed and the benchmark algorithms use similar construction methods, if same benchmark dataset is used for producing performance estimates in the prediction tasks, learning curves can be drawn to perform comparisons.

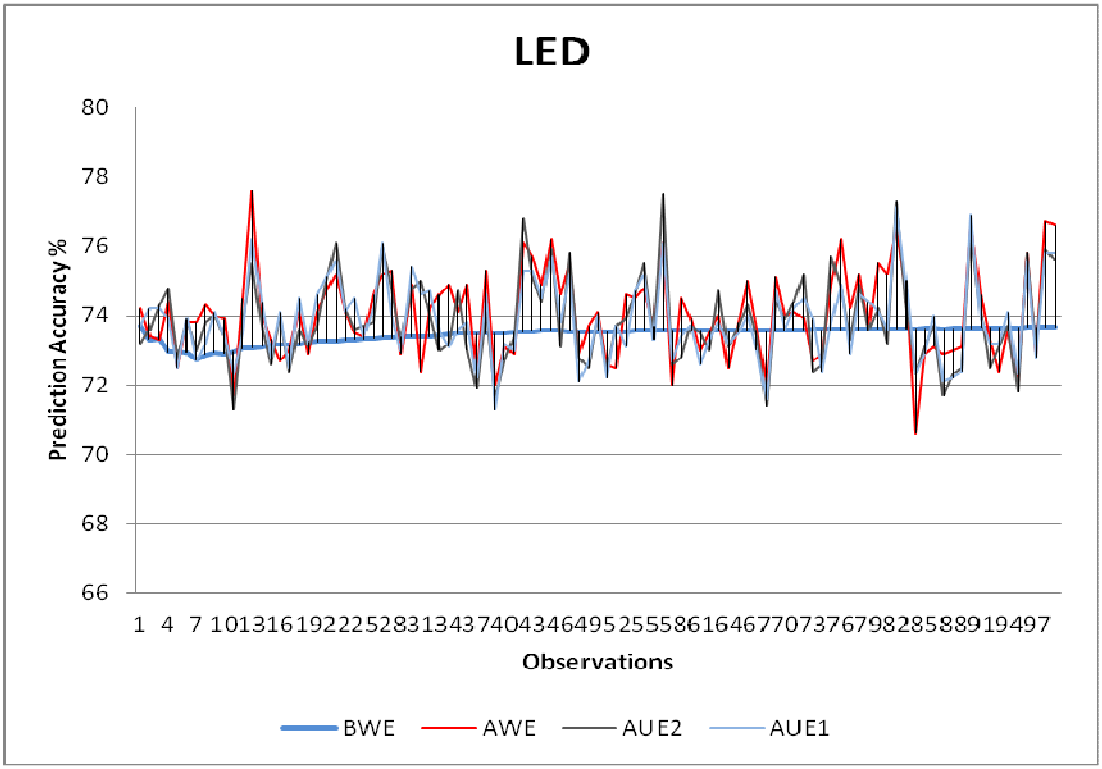

Figure 4- Learning Curves for Boosted-Window Ensemble (BWE), Accuracy-Weighted Ensemble (AWE), Accuracy-Updated Ensemble2 (AUE2), and Accuracy-Updated Ensbmle1 (AUE1) using LED dataset (Synthetic Dataset)

Figure 4 and 5 show the learning curves drawn using 100 observations of the prediction accuracy obtained at equal intervals by using BWE, AWE, AUE1, and AUE2. These intervals are plotted along the x-axis and the prediction accuracy is plotted along the y-axis for each algorithm. The LED and the Poker-lsn datasets are used in these prediction tasks as the benchmark datasets. In this case, these are selected as the exemplary Synthetic and Real-World datasets respectively. Whereas, the learning curves of the benchmark algorithms i.e., AWE, AUE1, and AUE2 show obvious interruptions (i.e., deviations from respective average values), BWE has a comparatively smoother learning-curve. Interruptions in the learning curve depict a learning loss during re-learning (Yelle, L. E. 1979). Similar learning curves are observed for the given algorithms in case of all the chosen synthetic and the real-world datasets.

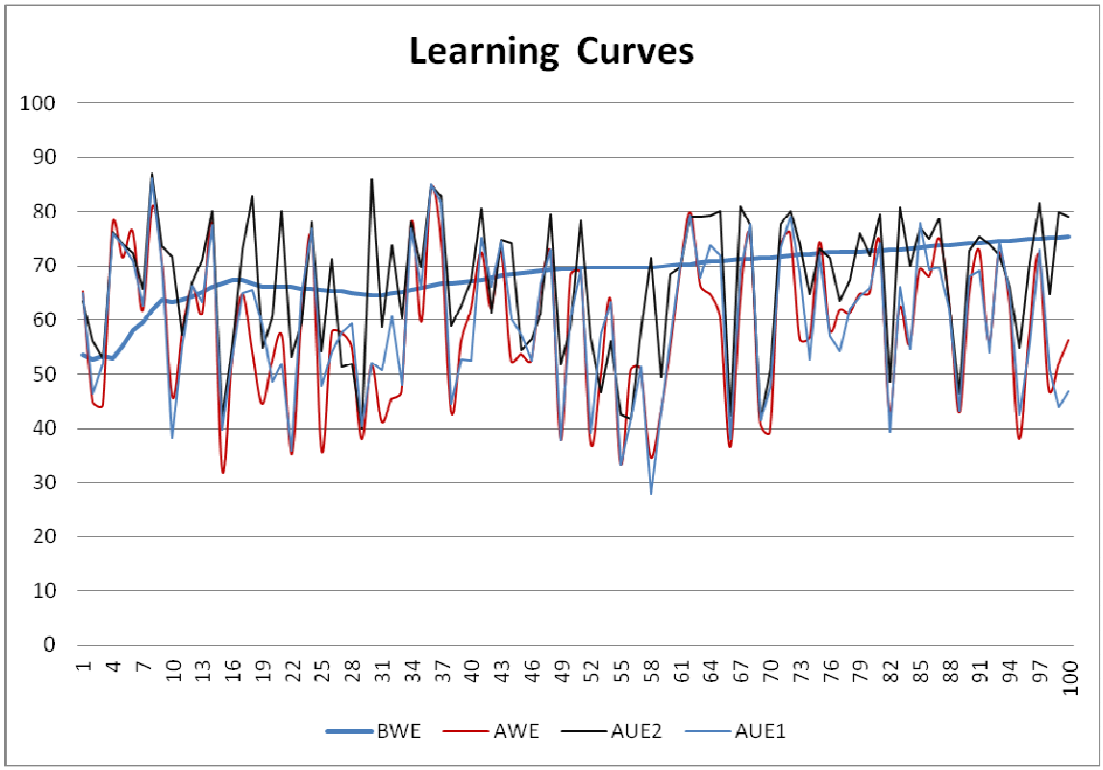

Figure 5- Learning Curves for Boosted-Window Ensemble (BWE), Accuracy-Weighted Ensemble (AWE), Accuracy-Updated Ensemble2 (AUE2), and Accuracy-Updated Ensbmle1 (AUE1) using Poker-lsn dataset (Real-world Dataset)

6 METHOD

This thesis work follows an empirical investigation method, i.e., experimentation approach. Experimentation is an established technique frequently used for the comparison of different machine-learning algorithms for random learning problems (Hothorn et al. 2005; Sjøberg et al. 2005). The purpose of conducting these experiments is to assess the performance of the proposed algorithm.

6.1 Problem statement

The problem of obtaining predictions on data-streams can be described as training of the supervised-learning algorithm on the labeled-instances of a stream and predicting the class-values of unseen instances (Salzberg 1997). The nature of the stream data makes this task complicated (Hoens et al. 2012). In contrast to the traditional machine learning tasks, where a complete training-data set is expected to be available, in stream mining, this is not the case (Brzeziński and Stefanowski 2011). The large number of data instances makes it a resource-constrained problem. The possibility of changes in the stream data adds further complexity to the problem. Various supervised-learning algorithms have been suggested by putting together efficient data-sampling, ensemble-learning, and incremental-learning methods (Wang et al. 2003; Brzeziński and Stefanowski 2011; Ade et al. 2013; Bifet et al. 2013; Brzezinski and Stefanowski 2014). Mixtures-of-experts technique has frequently been used in many of these algorithms to take advantage of methods that individually perform well (Masoudnia and Ebrahimpour 2012). Considering tradeoffs involved in achieving combinatorial optimization as explained Wolpert and Macready (1997), the problem is formulated as follows:

Given that Y_1 is a supervised-learning algorithm, using sliding window mechanism W_1, an ensemble-learning method E_1 and a base learner L; there can be a possibility to design a supervised-learning algorithm Y_2, using a sliding window mechanism W_2, an ensemble-learning method E_2 and a base learner L, such that the resulting algorithm can achieve faster speed and higher prediction accuracy than Y_1.

6.2 Problem Definition

Let $S = \{s_1, s_2, ..., s_n\}$ be a time ordered stream of *n* instances. If Y is a supervised-learning algorithm designed for performing the prediction on instances of S. If W_1 and W_2 are two different sliding window methods, E_1 and E_2 are two different ensemble-learning methods and L is a base learner, the performance of Y, i.e., PY can be determined by:

1. The sliding window method used by Y, i.e., W_1 or W_2 ; and
2. The ensemble-learning method used by Y, i.e., E_1 or E_2.

Given that data input (S) and base-learner L is same for the both methods, the associated hypothesis under test shall be as follows:

Hypothesis 0: *The prediction accuracy and CPU-time (in seconds) of the two learning algorithms do not change with the usage of different data-sampling and ensemble-learning methods.*

$$H0: PY((W_1, E1)|S, L) = PY((W_2, E2)|S, L)$$

6.3 Experimental Design

Controlled experiment is an established method for comparing the impact of two or more treatments on machine learning algorithms and has been used in similar studies (Prechelt at al. 2002). Quasi-experiment is a form of controlled experiment in which experimental units are assigned to treatments non-randomly (Sjøberg et al. 2005; Kampenes et al. 2009). This study uses quasi-experiments. The changes in performance (prediction accuracy and CPU-time) of the underlying supervised-learning algorithm are observed by changing the data sampling method (sliding window) and the underlying ensemble method while keeping other variables constant. Variables involved in this study are explained underneath.

The study follows an approach for comparing stream mining algorithm recommended by Salzberg, S. L. (1997). In this regard, the benchmark algorithms have been selected taking special care that the selected algorithms use similar construction techniques and method as the proposed algorithm. The benchmark datasets are selected for illustrating the strengths of the proposed algorithm. The datasets are prequentially evaluated by the proposed and the benchmark algorithms to produce performance estimates for each instance. Finally, statistical tests have been used to assess the significance of difference in the CPU-time and prediction accuracy estimates produced by the proposed and the benchmark algorithms.

6.3.1 Independent Variables

An independent variable is a quantity that is directly controlled by the observer or experimenter. In this experiment, the sliding window updation method and ensemble-learning method are the independent variables. It is assumed that changing either of the three can have an impact on the performance of the algorithm. It is worth noting that one of these variables is changed at a particular time and other variables are kept constant.

6.3.2 Dependent Variables

The dependent variable, as the name suggests, depends upon the independent variable. The ability of an underlying learning algorithm to predict unseen data as a consequence of using a particular data-sampling and ensemble method is dependent variable. The ability of a learning algorithm to predict unseen data cannot be measured directly. Different parameters are used for this purpose. Normally it is measured by observing the changes in prediction accuracy, CPU-time (execution time in seconds), and memory requirements of the learning algorithms (Gama et al. 2013). In this thesis work, prediction accuracy, and CPU-time (execution time in seconds) are used to measure the effect of the treatments.

6.3.3 Constants

Constants are the factors whose values remain same throughout the experiment. The datasets and the base learner are set as constants in the experiment.

6.3.4 Experiments

The performance estimates of the proposed and the benchmark algorithms are generated by obtaining predictions on a learning sample consisting of nine synthetic and five real-world datasets.

Following section provides the details of experimental settings used in our experiments.

6.4 Experimental settings

The purpose of the experiment is to evaluate the performance of proposed Partially-Updating Sliding Window (PUSW) and Online Boosting, in BWE. This thesis demonstrates PUSW working together with the Online Boosting algorithm, using the PUSW as its data-sampling method. The class label prediction for each instance is obtained by considering the majority of the votes of base learners in the ensemble in favor of a particular class. Online Boosting has been shown to perform well against a variety of other methods in the data-stream context (Oza, N. C. 2005). The experiments compare the performance of BWE with the state-of-the-art online-learning algorithms including: AUE1, AUE2, and AWE that use tree ensembles and Sliding Window. These methods and their parameters are given in Table 1 and Table 2.

Table 1- Experimental Treatments

Algorithm	**Sliding Window**	**Ensemble method**	**Ensemble-size**	**Base Learner**
BWE	Partially-Updating Sliding Window	Online Boosting	10	Hoeffding Tree
AUE1	Chunk-based Sliding Window	Accuracy-updated Ensemble	15	Hoeffding Tree
AUE2	Chunk-based Sliding Window	Accuracy-updated Ensemble	15	Hoeffding Tree
AWE	Chunk-based Sliding Window	Accuracy-weighted Ensemble	15	Hoeffding Tree

All of the algorithms evaluated in this thesis work are implemented in Java by extending the MOA software (Bifet et al.2010).

The operating system and hardware specifications of the machine used for running the experiments are as follows:

Windows 7 Home Premium, 64 bit operating system
6 GB main memory
AMD E2-1800 APU with Radeon (tn) HD Graphics 1.70 GHz

Table 2- Sliding Window Settings

Window Method	Window Size	Update Size	Update Method
PUSW	1000	250	# instances
PUSW	1000	500	# instances
Chunk-based Sliding Window	500	500	Time

Following section provides the details of the datasets used in the experiments:

6.5 Datasets

Two types of datasets have been used to generate performance estimates of the proposed and the benchmark supervised-learning algorithms. The synthetic datasets have been generated by using the stream generators in Massive Online Analysis (MOA) and Waikato Environment for Knowledge Analysis (WEKA) (Hall et al. 2009; Bifet et al. 2010). The real-world datasets have been downloaded from the UCI repository (Asuncion and Newman 2007). Brief details of the selected datasets are as following:

Table 3- Synthetic Datasets

Dataset	#Attributes	#Nominal	#Numerical	#Classes	#Instances
Hyperplane	10	0	10	2	10^6
LED	24	0	24	10	10^6
LEDDrift	24	0	24	10	10^6
RandomRBF	10	0	10	2	10^6
RandomTree	10	5	5	2	10^6
SEA	3	0	3	2	10^6
Waveform	40	0	40	3	10^6
WaveformDrift	40	0	40	3	10^6
WekaRDG	10	10	0	2	10^6

The *hyperplane dataset* represents the geometric problem of predicting class of a rotating hyperplane. The rotation of the hyperplane induces the concept-drift in the data (Tsymbal et al. 2008). The classification algorithm predicts the target attribute by looking at 10 feature attribute values. To generate the dataset, the default seed value for random generation of instances is used. The number of attributes with drift is set to two. The noise percentage is to 5% and the percentage of the probability that the direction of change is reversed is placed at 10%.

The *LED* dataset contains attribute values for seven-segment LED display, and the classification algorithm predicts the digit displayed (Breiman et al. 1984). Seed value for random generation of instances has been set to one. And the noise percentage is 10%.

The *LED Drift* dataset represents the problem of predicting the digit display on a seven-segment LED display. Concept-drift is introduced in this dataset (Breiman et al. 1984). The number of attributes with concept-drift has been set to one. And the noise percentage is placed at 10%.

The *RandomRBF* dataset is generated using a radial basis function (RBF) that generates real-numbers (Bifet et al. 2010). The values of attributes depend only on the distance from the origin. The supervised-learning algorithm approximates the

function that generated the data. While generating the dataset, random seed value is one, instance random seed value is one and number of centroids is 50. For this dataset, the number of feature attributes is 10 and the number of class values is two.

The *RandomTree dataset* contains five nominal and five numeric-attributes. The supervised-learning algorithm predicts one of the two class values for a particular instance. While generating the dataset, the value of seed for random generation of tree and seed for random generation of instances is placed at one. The number of values to generate each nominal attribute is placed at five. The maximum depth for the tree concept is placed at five. The leaf level or first level of the tree above the maximum depth, which can have leaves, is three, and the fraction of leaves per level from first leaf level onwards is 0.15.

SEA dataset contains instances generated by *"simple ensemble algorithms concepts"* functions (Street and Kim2001). The dataset has three numerical attributes and a nominal class attribute. A single function is used to generate the dataset, with single seed for random generation of instances. Percentage of noise to be added to data is set to 10%.

In the *Waveform dataset*, each instance contains attribute values representing one of the three waveforms (Asuncion and Newman 2007). This dataset helps solving electronics/electrical problems. The dataset is generated using a single seed for random generation of instances.

WaveformDrift dataset has similar instances as Waveform dataset. In this case, concept-drift is introduced. Number of attributes with drift is set to one. Noise is added for a total of 40 attributes.

WekaRDG dataset contains a randomly generated decision list. These rules are generated by a function in WEKA (Hall et al.2009). And each instance of the list represents a rule. The function generating the rules observes that if a decision list fails to classify the current instance, a new rule according to this current instance is generated. While generating the dataset, maximum number of tests in the rules is 10 and minimum number of tests in rules is one. The seed value for the random number generator is 1 and the voting is turned off.

Table 4- Real-World Datasets

Dataset	**#Attributes**	**#Nominal**	**#Numerical**	**#Classes**	**#Instances**
Airlines	7	4	3	18	539,383
CoverTypeNorm	54	44	10	7	581,012
ElecNormNew	8	2	6	2	45,312
Imdb-e	1001	0	1001	2	120,919
Poker-lsn	10	5	5	4	829,201

The *airlines dataset* contains statistics of arrivals and departures of flights. Instances include attributes such as the airline name, flight number, departing airport, destination airport, and associated time attributes. The supervised-learning algorithm detects delays.

CoverTypeNorm dataset represents the problem of predicting forest cover type, from cartographic variables. The dataset has 54 attributes and 7 possible class values. Depending on the values of attributes the supervised-learning algorithm predicts the

target attribute value for a particular instance. Individual attributes contain cartographic values and the target attribute (class) contains wilderness type.

ElecNormNew (Electricity) dataset contains data collected from the Australian New South Wales Electricity Market. In this market, the prices of electricity are determined by the demand and supply and are set every five minutes. The dataset contains 45, 312 instances. The supervised-learning algorithm predicts the increase or decrease in the per unit price of electricity at a particular time.

IMDB (Internet Movie Database) dataset is a subset of the Internet movie database. *Imdb-E* dataset has large number of attributes. The preference of the user, while selecting a movie, is predicted depending upon values in 1001 attributes. This dataset represents binary sentiment classification problem.

Poker-lsn dataset has 10 attributes and 10 possible, class-attribute values. Each instance of this dataset represents a hand consisting of five playing cards drawn from a deck of 52 cards. Each card is described using two attributes, suit, and rank. The target attribute or the class can have one of 10 possible values.

6.6 Software Platform

Waikato Environment for Knowledge Analysis (WEKA) and Massive Online Analysis (MOA) (Hall et al. 2009; Bifet et al. 2010) are used for performing the experiments. WEKA and MOA are open source platforms for machine learning and data mining. Same platforms are supported by AUE1, AUE2, and AWE. We have used these platforms to have like environment. Using like platforms has both its strengths and limitations. This makes the comparative evaluation fair. However, it induces the inability to control impact of the environment in which the experiment is being conducted. WEKA and MOA are implemented in Java. Matching programming language has been used in our code level implementation.

6.7 Validity Threats

The probability of rejecting a null hypothesis when it is true (Type I error) or chances of failing to reject a null hypothesis when it is false (Type II error) is always there (Rothman, K. J. 2010). These errors can result from incorrect assumptions about data, showing favorable results through fishing or incorrect measurement of effect size. Comparisons of heterogeneous units, lack of standardization and inability to control impact of the environment in which the experiment is being conducted, can also lead to erroneous inferences (Christenfeld et al. 2004).

Following sub-chapters provide details of the measures taken for mitigating the possible validity threats.

6.7.1 Construct Validity

Construct validity describes how well the devised measurements stand in, for target scientific concepts (Güleşir et al. 2009). For example, this thesis is studying the impact of certain treatment on the efficiency of underlying supervised-learning algorithms. In this regard, classification accuracy and CPU-time are chosen as performance indicators. The selection of performance indicators takes care of the ‘missing explanatory variables’ problem as well (Jones 1996).

6.7.2 External Validity

External validity is synonymous with scalability (Sjøberg et al. 2005). The threat mainly lies in using lab settings and convenience data samples in experiments and extending the results to real-world problems. Synthetic and real-world datasets with and without concept-drift have been selected for generating the performance estimates. A complete description of the used method, including: problem statement, problem definition, experimental design, experimental settings, and the learning samples have been supplied in Chapter 7. Performance of the proposed algorithm should be evaluated against any new dataset, before inferring any additional conclusions.

6.7.3 Internal Validity

Internal validity of an experiment deals with whether the reflected changes are caused by the presumed treatment or from an alternative (Sjøberg et al. 2005). The use of open source APIs (MOA and WEKA) provides a solid platform for the experiments in this thesis. The selection has been made to avoid efforts involved in re-doing basic tasks. However, such open source platforms are subject to continuous development. Enhanced features, or bug fixes in the basic functionality, can affect the results of experiment. Same versions of the software have been used to evaluate the original and the proposed algorithms.

7 EXPERIMENTAL RESULTS

The experimental results are summarized in the given tables. Table-5 shows the Prediction accuracy of the proposed and the benchmark algorithms for the chosen datasets. Table-6 presents, the CPU-time (in seconds) spent for completing the prediction task by the proposed and the benchmark algorithms for selected datasets.

7.1 Synthetic Datasets

Table 5- Prediction Accuracy (in percentage)

Dataset	AUE1	AUE2	AWE	BWE
Hyperplane	91.27	90.69	93.41	87.14
LED	73.84	73.85	74.03	73.45
LEDDrift	73.84	73.85	74.03	73.45
RandomRBF	94.81	94.78	73.02	92.80
RandomTree	96.88	96.83	79.65	91.11
SEA	89.73	89.76	87.74	88.80
Waveform	83.48	84.37	81.55	83.55
WaveformDrift	83.10	83.75	81.51	83.11
WekaRDG	100.00	99.63	86.37	98.19

Looking at Table 5, we can see that in most of the cases, the prediction accuracies of all the four algorithms are comparable. While observing pair-wise values, we discover that an individual algorithm might out-perform one or more algorithms for certain datasets. However, there is no single supervised-learning algorithm among the four that outperforms all the rest, for all the datasets.

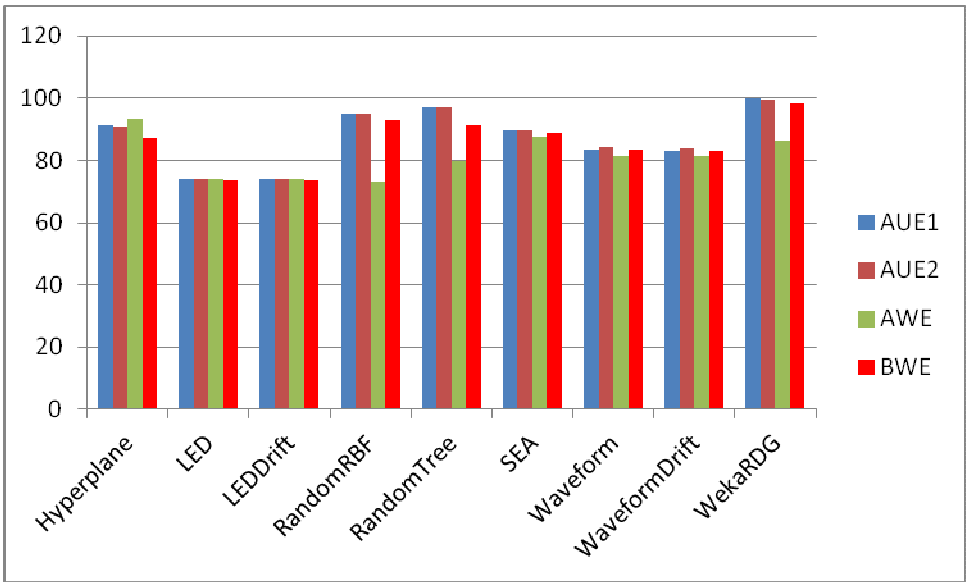

Figure 6- Prediction Accuracy for Synthetic Datasets

Table 6 - CPU-time (execution time in seconds)

Dataset	AUE1	AUE2	AWE	BWE
Hyperplane	694	356	506	265
LED	2379	961	1771	441
LEDDrift	2378	1000	1759	453
RandomRBF	611	314	438	247
RandomTree	714	225	562	180
SEA	324	122	186	100
Waveform	1780	693	1428	479
WaveformDrift	3195	1244	2430	790
WekaRDG	362	117	283	48

Table-6 presents the CPU-time (in seconds) required to carry out the prediction task by the proposed and the benchmark algorithms, for the selected synthetic datasets. Looking at the values, as measured during the experiments, it can be observed that the CPU-time spent by all the four supervised-learning algorithms to perform the prediction task on a given dataset is different. However, we can clearly observe that, for all datasets, the proposed algorithm has the least CPU-time values.

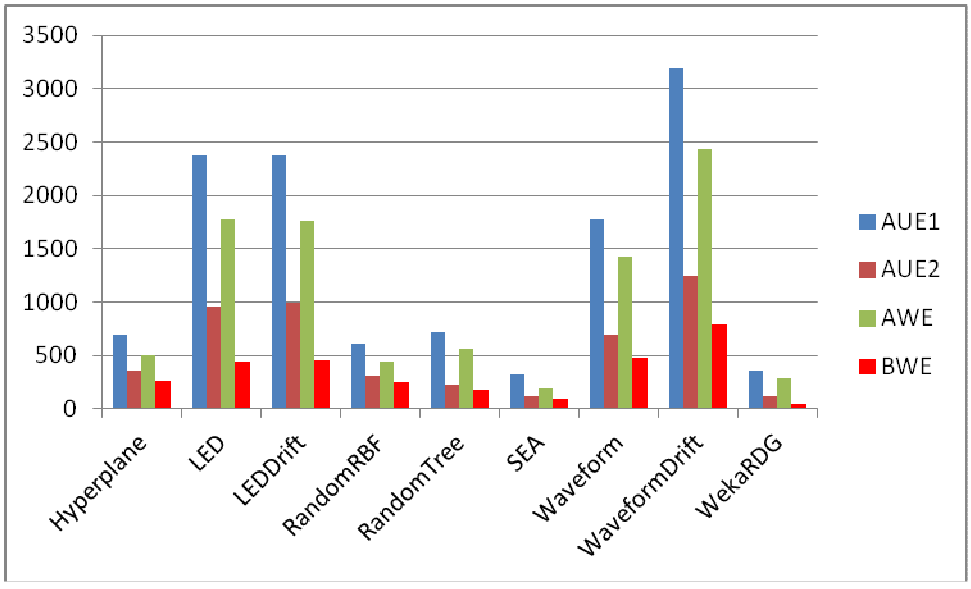

Figure 7- CPU-time for Synthetic Datasets

7.2 Real-World Datasets

Table 7 and 8 show the evaluation results of the proposed and benchmark algorithms for real-world datasets. Similar to the synthetic datasets, in a majority of the cases, the prediction accuracy of the proposed and the benchmark algorithms is comparable. However, the CPU-time for the proposed algorithm is lower than that of benchmark algorithms for all the selected datasets.

Table 7- Prediction Accuracy (in percentage)

Dataset	AUE1	AUE2	AWE	BWE
Airlines	62.89	66.89	62.11	60.32
CoverTypeNorm	80.78	87.63	80.78	80.07
ElecNormNew	72.84	77.48	70.80	76.11
Imdb-e	72.38	72.50	72.95	71.14
Poker-lsn	59.73	67.05	58.65	68.60

As is the case of synthetic datasets, the prediction accuracies of the benchmark and the proposed algorithms are comparable in the case of real-life datasets. For the majority of the datasets, AUE2 has higher accuracy values, but for two datasets (i.e. imdb-e and poker-lsn), AWE, and BWE slightly outperform AUE2. Moreover, in multiple cases the proposed algorithm outperforms at least one of the three benchmark-algorithms in terms of the prediction accuracy.

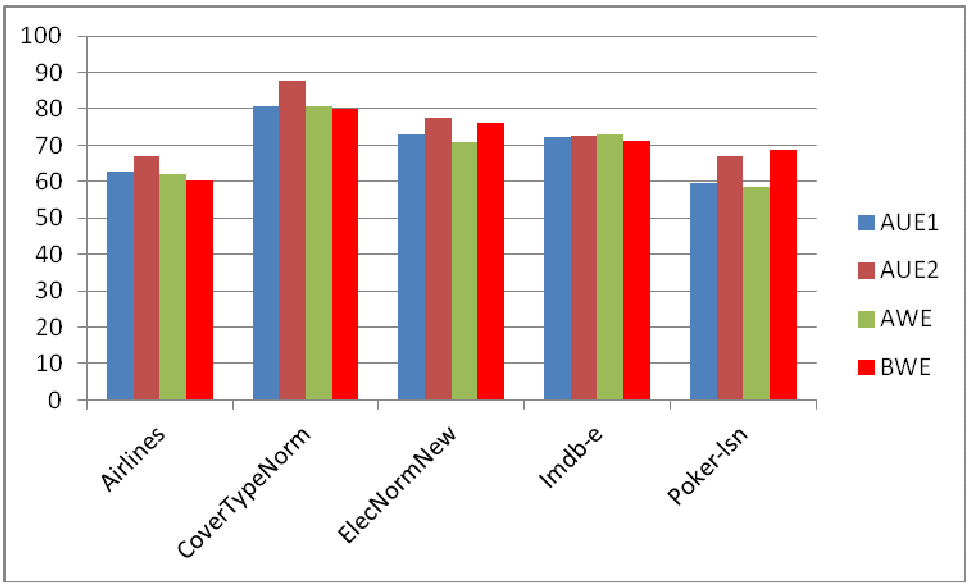

Figure 8- Prediction Accuracy for Real-world Datasets

Table 8- CPU-time (execution time in seconds)

Dataset	AUE1	AUE2	AWE	BWE
Airlines	915	733	944	112
CoverTypeNorm	1226	620	1210	317
ElecNormNew	12	10	11	8
Imdb-e	3438	3170	5127	1448
Poker-lsn	276	228	249	134

It can be observed in Table 8, that the CPU-time requirement for the proposed algorithm is consistently lower than all the three benchmark algorithms, for all datasets. In most of the cases, the difference is obvious. Only case where the difference is minor is the case where the number of instances in the dataset is small.

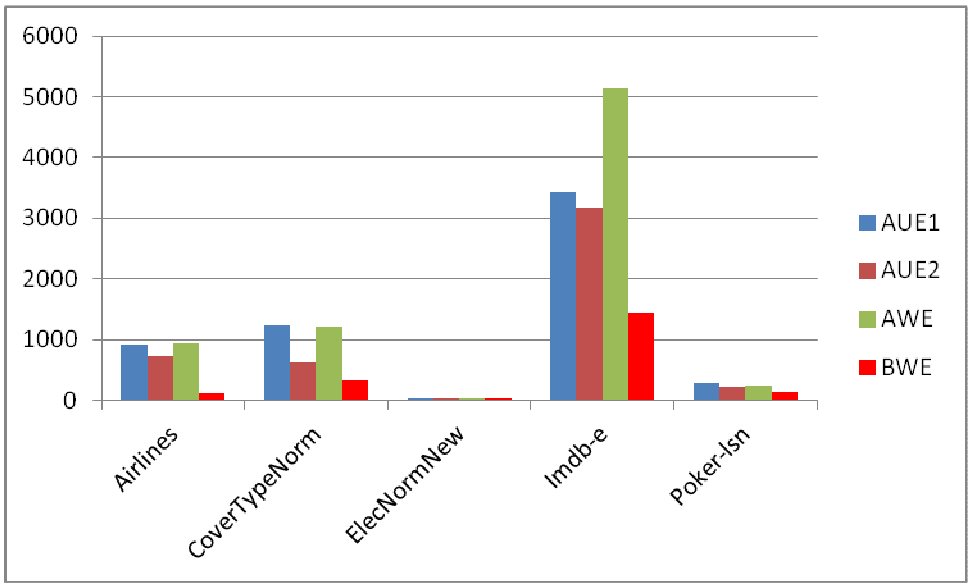

Figure 9- CPU-time for Real-World Datasets

8 STATISTICAL ANALYSIS

As this thesis is comparing multiple methods, we have used multiple-comparison tests. Multiple-comparison tests help in drawing several statistical inferences simultaneously (Derrac et al. 2011). Furthermore, to account for the multiplicative effect, in order to control the Family-Wise-Error-Rate, nonparametric tests are more suitable for use. Such tests help controlling of the family wise error, which is the probability of making one or more false discoveries. Pair wise tests do not control the error propagation of making more than one comparison and therefore, should not be used in multiple comparisons (García et al. 2010). The Asymptotic Friedman test has been used in this regard. It is used to detect differences in treatments across multiple tests. Multiplicative effect of the given treatment is measured, and post-hoc analysis is carried out to detect outliers. The Wilcoxon Signed-Rank test is used to test the hypothesis for investigating RQ3.

For measuring the significance of the impact of selected treatments, the null hypothesis stated in sub-chapter 6.2 needs to be tested. It is a high level hypothesis formulated for testing a composite effect of the changes made in the two independent variables at the same time. For clarity reasons, the testing proceeds by assessing the performance estimates generated by changes caused in one variable at a time. A cause-effect relationship has been established assuming that the sliding window method determines the CPU-time and the selected ensemble method determines the prediction accuracy of an algorithm. The null and the alternative hypotheses are formulated in accordance with the research questions.

Following hypotheses are formulated such that the conclusions drawn by testing them can in effect, imply whether the high level null hypothesis can be accepted or rejected.

In the following sub-chapters, hypothesis testing is reported separately for the performance estimates obtained from the synthetic and the real-world datasets.

8.1 Synthetic data

RQ1: What is the impact of using Partially-Updating Sliding Window on the CPU-time (execution time in seconds) of the supervised-learning algorithm?

H0: The use of Partially-Updating Sliding Window does not change the CPU-time of the underlying supervised-learning algorithms.

Conclusion: At confidence level of 0.05, the Asymptotic Friedman test returns a P-value of 5.887e-06 indicating that the null hypothesis can be rejected. We conclude that use of Partially-Updating Sliding Window technique changes the CPU-time of given supervised-learning algorithm. To further investigate the impact of individual

treatments on the speed of given algorithms Wilcoxon-Nemenyi-McDonald-Thompson test has been used. The test results are as follows:

Table 9- Pair-wise Comparison (CPU-time)

Pair	**P-Value**
AUE2 – AUE 1	5.53e-03
AWE - AUE1	3.541e-01
BWE – AUE1	1 5.23e-06
AWE - AUE2	3.54e-01
BWE - AUE2	3.54e-01
BWE - AWE	5.28e-03

Looking at Table 9, the results of the post-hoc test indicate that:

1. For the first pair, the null hypothesis is rejected indicating a significant difference in CPU-time values of AUE2 and AUE1.
2. For the second pair, the null hypothesis is rejected indicating a significant difference in CPU-time values of AWE and AUE1.
3. For the third pair, the null hypothesis is rejected indicating a significant difference in CPU-time values of BWE and AUE1.
4. For the fourth pair, the null hypothesis cannot be rejected indicating no significant difference in CPU-time values of AWE and AUE2.
5. For the fifth pair, the null hypothesis cannot be rejected indicating no significant difference in CPU-time values of BWE and AUE2.
6. For the last pair, the null hypothesis is rejected indicating significant difference in CPU-time values of BWE and AWE.

RQ2: What is the impact of using Online Boosting instead of Accuracy-Updated ensemble on the Prediction accuracy of the underlying learning algorithm?

H0: The use of Online Boosting instead of Accuracy-Updated Ensemble does not change the prediction accuracy of the underlying learning algorithm.

Conclusion: At a confidence level of 0.05, the Asymptotic Friedman test returns p-value of 0.013, indicating that the null hypotheses can be rejected. We conclude that use of different treatments changes the prediction accuracy of the underlying learning algorithm. Post-hoc analysis is performed using Wilcoxon-Nemenyi-McDonald-Thompson test. The test results are as follows:

Table 10- Pair-wise Comparison (BWE vs AUE)

Pair	**P-Value**
BWE - AUE1	0.085
BWE - AUE2	0.013
AUE2 - AUE1	0.759

The returned p-values show that in case of BWE and AUE1, the null hypothesis cannot be rejected. The data appear to be consistent with the null hypothesis.

However, in the case of the BWE and AUE2, the null hypothesis can be rejected, indicating a difference in the prediction accuracies of the two algorithms. It can be implied that AUE2 has significantly better accuracy than BWE.

RQ3: What is the impact of using Online Boosting instead of accuracy-weighted ensemble on the Prediction accuracy of the underlying supervised-learning algorithm?

H0: The use of Online Boosting instead of accuracy-weighted ensemble does not affect the prediction accuracy of the underlying supervised-learning algorithm.

Conclusion: The null hypothesis has been tested using Wilcoxon signed rank test at a confidence level of 0.05. The returned p-value 0.12 provides no evidence against the null hypothesis. We conclude that the use of different given treatments did not affect the prediction accuracy of the given supervised-learning algorithm.

8.2 Real-World datasets

RQ1: What is the impact of using Partially-Updating Sliding Window on the CPU-time of the underlying supervised-learning algorithm?

H0: The use of Partially-Updating Sliding Window does not change the CPU-time of the underlying supervised-learning algorithm.

Conclusion: At the confidence level of 0.05, the Friedman test returns a p-value of 0.00357, indicating that the null hypothesis can be rejected. We conclude that the use of Partially-Updating Sliding Window technique changes the CPU-time of the underlying supervised-learning algorithm. To investigate the impact of individual treatments on the CPU-time of the given algorithms, Wilcoxon-Nemenyi-McDonald-Thompson test has been used. The results of the test are as follows:

Table 11- Pair-wise Comparison (CPU-time)

Pair	**P-Value**
AUE2 – AUE 1	0.20
AWE - AUE1	0.99
BWE – AUE1	0.008
AWE - AUE2	0.310
BWE - AUE2	0.61
BWE - AWE	0.017

Looking at Table 10, the results of the post-hoc test indicate that:

1. For the first pair, the null hypothesis cannot be rejected indicating not significant difference in CPU-time values of AUE2 and AUE1.
2. For the second pair, the null hypothesis cannot be rejected indicating no significant difference in CPU-time values of AWE and AUE1.

3. For the third pair, the null hypothesis is rejected indicating a significant difference in CPU-time values of BWE and AUE1.
4. For the fourth pair, the null hypothesis cannot be rejected indicating no significant difference in CPU-time values of AWE and AUE2.
5. For the fifth pair, the null hypothesis cannot be rejected indicating no significant difference in CPU-time values of BWE and AUE2.
6. For the last pair, the null hypothesis is rejected indicating significant difference in CPU-time values of BWE and AWE.

RQ2: What is the impact of using Online Boosting instead of Accuracy-Updated Ensemble on the prediction accuracy of the underlying supervised-learning algorithm?

H0: The use of Online Boosting instead of the Accuracy-Updated Ensemble does not affect the prediction accuracy of the underlying supervised-learning algorithm.

Conclusion: Running Asymptotic Friedman test on the relevant groups of data at a significance level of 0.05 returns a p-value of 0.09. The high p-value indicates that the null hypothesis cannot be rejected. Hence, it can be concluded that different treatments did not have an impact on the prediction accuracy of the underlying supervised-learning algorithm.

RQ3: What is the impact of using the Online Boosting instead of the Accuracy-Weighted Ensemble on the prediction accuracy of the underlying supervised-learning algorithm?

H0: The use of Online Boosting instead of Accuracy-Weighted Ensemble does not affect the prediction accuracy of the underlying supervised-learning algorithm.

Conclusion: At a confidence level of 0.05 running the Wilcoxon Signed Rank test on the relevant groups of results returned a p-value 0.81 indicating that the null hypothesis cannot be rejected. We conclude that the use of the given treatment did not have change the prediction accuracy of the underlying supervised-learning algorithm.

Figure 10- Boxplots of the Differences (Real-world Datasets CPU-time)

8.3 Synthesis

Based upon the observations and the results of hypotheses testing the null hypothesis in sub-chapter 6.2 can be rejected and an overall conclusion can be drawn that:

The prediction accuracy and CPU-time (in seconds) of the two learning algorithms change with the usage of different data-sampling and ensemble-learning methods.

8.4 Parallel Coordinates Plots

Parallel coordinates plot (PCP) is a tool used for multivariate visualization and analysis. Multivariate statistics is a form of statistics that involves the simultaneous observation and analysis of more than one outcome variable. PCP is simple in concept: Coordinates of multivariate observations are plotted on parallel axes and connected by line segments (Huh and Park 2008). Each axis corresponds to a variable. Observations are shown in a PCP as a series of unbroken line segments. These lines are termed as polylines. Each polyline passes through the axis at a location that indicates the observation's value relative to all other values. The ends of the axis represent the maximum and minimum values of the axis variable for all observations under consideration (Edsall 2003). PCPs can reveal and highlight hidden trend and correlation information of polylines through their patterns (Siirtola and Räihä 2006; Zhao and Kaufman 2012). Small number of observations and ease of understanding make PCP a preferred choice for this study.

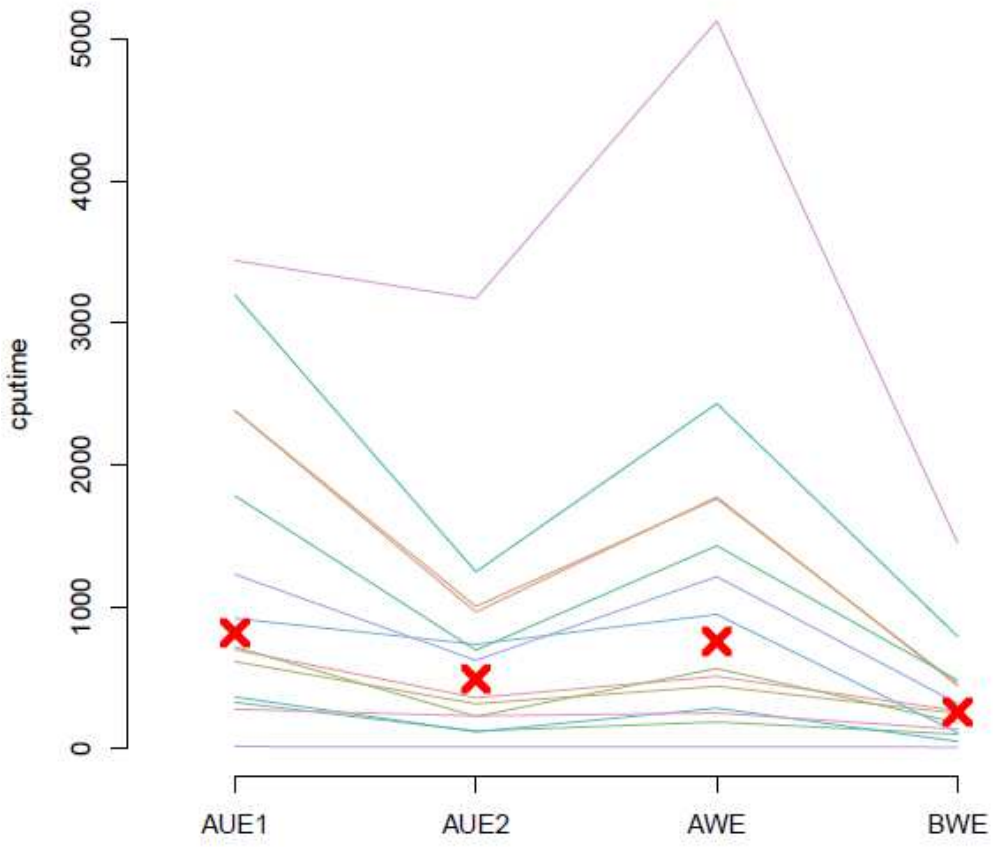

Figure 11- PCP for CPU-time values shows that BWE has an overall lower CPU-time while obtaining predictions from the 14 chosen datasets. PCP is drawn using experimental results given in Tables 6 and 8.

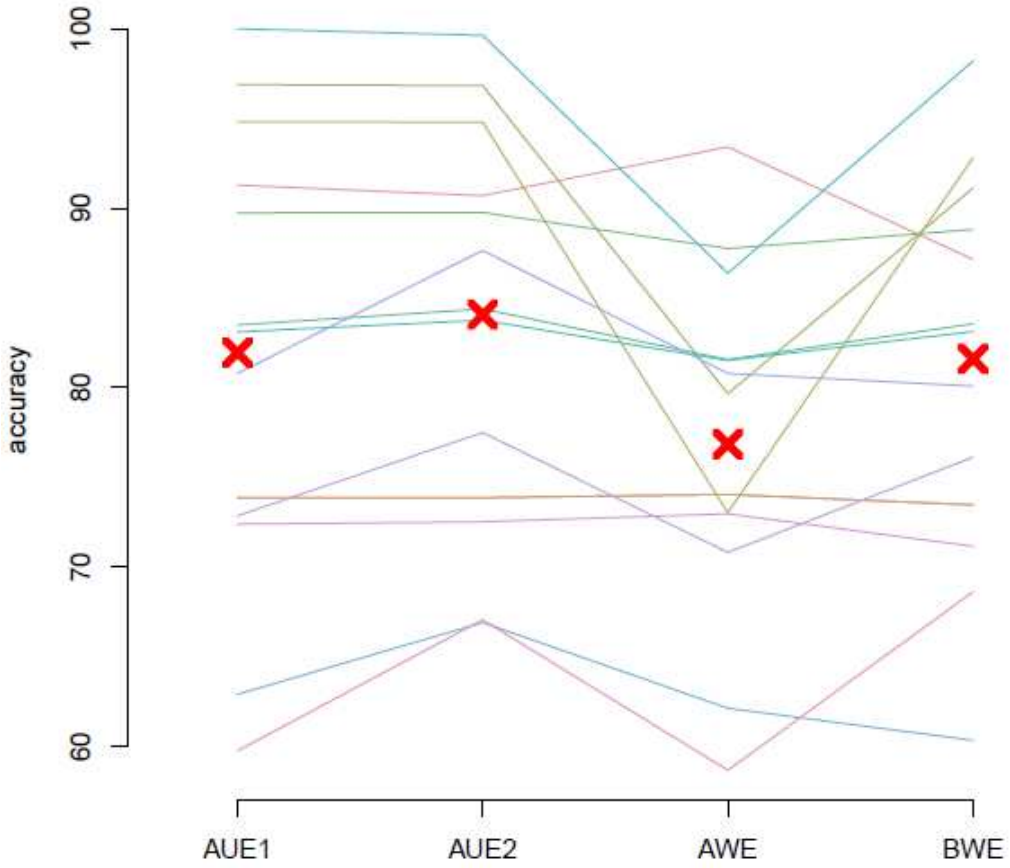

Figure 12- PCP for the prediction accuracy values shows that BWE has exhibited an overall accuracy better than AWE and similar to AUE1. However, AUE2 has better prediction accuracy than BWE. Plotted prediction accuracy values have been produced by the four algorithms while obtaining predictions from the 14 chosen datasets. Prediction accuracy values given in Tables 5 and 7 are used for this purpose.

9 CONCLUSIONS AND FUTURE WORK

9.1 Conclusions

It is presented that changing the update mechanism of the sliding window can result in lower CPU-time of the stream mining algorithms. A new method called Partially-Updating Sliding Window (PUSW) is proposed, which is simple in construction and can be used for designing innovative stream mining algorithms. By using Online Boosting and incremental-learning on the proposed sliding window method, a new algorithm is proposed and is called as Boosted-Window Ensemble (BWE). BWE solves the problem of obtaining predictions from the data-streams. The performance of BWE is evaluated against state-of-the-art data mining algorithms. The results suggest that the proposed technique improves the performance of BWE in most of the cases by reducing the CPU-time and achieving the prediction accuracy equivalent to the state-of-the-art stream mining algorithms.

9.2 Future Work

In future, using the proposed sliding window technique with different ensemble methods such as accuracy-weighted and accuracy-updated ensembles can extend the research. Furthermore, the possibility of increasing the prediction accuracy of the BWE can be investigated. In addition, it could be interesting to use the proposed algorithm in the industry for measuring its practical significance.

.

REFERENCES

A. Asuncion and D. Newman. (2007) *UCI Machine Learning Repository*. http://www.ics.uci.edu/~mlearn/MLRepository.html.

Ade, M. R., GHRIET, P., Deshmukh, P. R., & SCOE&T, A. (2013). Methods for Incremental-learning: a survey. *International Journal of Data Mining & Knowledge Management Process*, *3*(4).

Bifet, A., Holmes, G., Pfahringer, B., Kirkby, R., & Gavaldà, R. (2009). New ensemble methods for evolving data streams. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 139-148).

Bifet, A., Holmes, G., Kirkby, R., & Pfahringer, B. (2010). Moa: Massive online analysis. *The Journal of Machine Learning Research*, *11*, 1601-1604.

Bifet, A., Pfahringer, B., Read, J., & Holmes, G. (2013, March). Efficient data-stream classification via probabilistic adaptive windows. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing* (pp. 801-806)..

Bolchini, C., Quintarelli, E., Salice, F., & Garza, P. (2013). A data mining approach to incremental adaptive functional diagnosis. In *IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), 2013* (pp. 13-18).

Bouchachia, A., & Nedjah, N. (2012). Introduction to the special section on self-adaptive systems: models and algorithms. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, *7*(1), 13.

Braverman, V., Ostrovsky, R., & Zaniolo, C. (2009). Optimal sampling from Sliding Windows. In *Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (pp. 147-156).

Breiman, L., Friedman, J., Stone, C. J., & Olshen, R. A. (1984). Classification and Regression Trees. *CRC press*.

Brzeziński, D., & Stefanowski, J. (2011). Accuracy-updated ensemble for data-streams with concept-drift. In *Hybrid Artificial Intelligent Systems* (pp. 155-163). Springer Berlin Heidelberg.

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

Chen, H., Shu, L., Xia, J., & Deng, Q. (2012). Mining frequent patterns in a varying-size Sliding Window of online transactional data-streams. *Information Sciences*, *215*, 15-36.

Chen, Y. L., Wu, C. C., & Tang, K. (2009). Building a cost-constrained decision tree with multiple condition attributes. *Information Sciences*, *179*(7), 967-979.

Christenfeld, N. J., Sloan, R. P., Carroll, D., & Greenland, S. (2004). Risk factors, confounding, and the illusion of statistical control. *Psychosomatic Medicine*, *66(*6), 868-875.

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, *1*(1), 3-18.

Deypir, M., & Sadreddini, M. H. (2012). A dynamic layout of Sliding Window for frequent itemset mining over data-streams. *Journal of Systems and Software*, *85*(3), 746-759.

Deypir, M., Sadreddini, M. H., & Hashemi, S. (2012). Towards a variable size Sliding Window model for frequent itemset mining over data-streams. *Computers & Industrial Engineering*, *63*(1), 161-172.

Hulten, G., & Domingos, P. (2001). Catching up with the data: research issues in mining data streams. In *Proceedings of Workshop on Research Issues in Data Mining and Knowledge Discovery*.

.Domingos, P., & Hulten, G. (2000). Mining high-speed data-streams. In *Proceedings of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 71-80).

Duane, J. T. (1964). Learning curve approach to reliability monitoring. *IEEE Transactions on Aerospace*, *2*(2), 563-566.

Edsall, R. M. (2003). The parallel coordinate plot in action: design and use for geographic visualization. *Computational Statistics & Data Analysis*, *43*(4), 605-619.

Ferreira, P. M., & Ruano, A. E. (2009). Online sliding-window methods for process model adaptation. *IEEE Transactions on Instrumentation and Measurement*, *58*(9), 3012-3020.

Freund, Y., & Schapire, R. E. (1996). Experiments with a new boosting algorithm. In *Proceedings of the Thirteenth International Conference on Machine Learning (ICML '96)*, (pp. 148-156).

Gaber, M. M., Zaslavsky, A., & Krishnaswamy, S. (2005). Mining data-streams: a review. *ACM Sigmod Record*, *34*(2), 18-26.

Gama, J., Sebastião, R., & Rodrigues, P. P. (2013). On evaluating stream learning algorithms. *Machine Learning*, *90*(3), 317-346.

Gama, J. (2012). A survey on learning from data-streams: current and future trends. *Progress in Artificial Intelligence*, *1*(1), 45-55.

García, S., Fernández, A., Luengo, J., & Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: experimental analysis of power. *Information Sciences*, *180*(10), 2044-2064.

Golab, L., & Özsu, M. T. (2003). Issues in data-stream management. *ACM Sigmod Record*, *32*(2), 5-14.

Güleşir, G., van den Berg, K., Bergmans, L., & Akşit, M. (2009). Experimental evaluation of a tool for the verification and transformation of source code in event-driven systems. *Empirical Software Engineering*, *14*(6), 720-777.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, *11*(1), 10-18.

Hansen, J. V. (1999). Combining predictors: comparison of five meta machine learning methods. *Information Sciences*, *119*(1), 91-105.

Hoens, T. R., Polikar, R., & Chawla, N. V. (2012). Learning from streaming data with concept-drift and imbalance: an overview. *Progress in Artificial Intelligence*,*1*(1), 89-101.

Hothorn, T., Leisch, F., Zeileis, A., & Hornik, K. (2005). The design and analysis of benchmark experiments. *Journal of Computational and Graphical Statistics*,*14*(3), 675-699.

Huh, M. H., & Park, D. Y. (2008). Enhancing parallel coordinate plots. *Journal of the Korean Statistical Society*, *37*(2), 129-133.

Jessica Gurevitch and S. T. Chester Jr.,(1986). Analysis of Repeated Measures Experiments. *Ecology*, *67*(1), 251-255.

Jones, M. P. (1996). Indicator and stratification methods for missing explanatory variables in multiple linear regression. *Journal of the American Statistical Association*, *91*(433), 222-230.

Joshi, P., & Kulkarni, P. (2012). Incremental-learning: areas and methods—a survey. *International Journal of Data Mining & Knowledge Management Process*, *2*(5), 43-51.

Kampenes, V. B., Dybå, T., Hannay, J. E., & K Sjøberg, D. I. (2009). A systematic review of quasi-experiments in software engineering. *Information and Software Technology*, *51*(1), 71-82.

Kapp, M. N., Sabourin, R., & Maupin, P. (2011). A dynamic optimization approach for adaptive incremental-learning. *International Journal of Intelligent Systems*, *26*(11), 1101-1124.

Kholghi, M., Hassanzadeh, H., & Keyvanpour, M. (2010). Classification and evaluation of data mining techniques for data-stream requirements. In, *International Symposium on Computer Communication Control and Automation (3CA) 2010* (Vol. 1, pp. 474-478).

Li, F., & Liu, Q. (2008). An improved algorithm of decision trees for streaming data based on VFDT. In *International Symposium on Information Science and Engineering, 2008. ISISE'08.* (Vol. 1, pp. 597-600).

Masoudnia, S., & Ebrahimpour, R. (2012). Mixture of experts: a literature survey. *Artificial Intelligence Review*, 1-19.

Masud, M. M., Gao, J., Khan, L., Han, J., & Thuraisingham, B. (2010). Classification and novel class detection in data-streams with active mining. In *Advances in Knowledge Discovery and Data Mining* (pp. 311-324). Springer Berlin Heidelberg.

Meir, R., & Fontanari, J. F. (1992). Calculation of learning curves for inconsistent algorithms. *Physical Review A*, *45*(12), 8874.

Oza, N. C. (2005). Online bagging and Boosting. In *IEEE International Conference on Systems, Man and Cybernetics*, *2005* (Vol. 3, pp. 2340-2345).

Pechenizkiy, M., & Zliobaite, I. (2013). Introduction to the special issue on handling concept-drift in adaptive information systems. *Evolving Systems, 4*(1), 1-2.

Prechelt, L., Unger-Lamprecht, B., Philippsen, M., & Tichy, W. F. (2002). Two controlled experiments assessing the usefulness of design pattern documentation in program maintenance. *IEEE Transactions on Software Engineering*, 28(6), 595-606.

Rothman, K. J. (2010). Curbing type I and type II errors. *European Journal of Epidemiology*, *25*(4), 223-224.

Salzberg, S. L. (1997). On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, *1*(3), 317-328.

Shie, B. E., Yu, P. S., & Tseng, V. S. (2012). Efficient algorithms for mining maximal high utility itemsets from data-streams with different models. *Expert Systems with Applications*, *39*(17), 12947-12960.

Siirtola, H., & Räihä, K. J. (2006). Interacting with parallel coordinates. *Interacting with Computers*, *18*(6), 1278-1309.

Street, W. N., & Kim, Y. (2001). A streaming ensemble algorithm (SEA) for large-scale classification. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 377-382).

Sjøberg, D. I., Hannay, J. E., Hansen, O., Kampenes, V. B., Karahasanovic, A., Liborg, N. K., & Rekdal, A. C. (2005). A survey of controlled experiments in software engineering. *IEEE Transactions on Software Engineering, 31*(9), 733-753.

Thompson, W. J. (2001). Poisson distributions. *Computing in Science & Engineering, 3(3), 78-82*.

Tsymbal, A., Pechenizkiy, M., Cunningham, P., & Puuronen, S. (2008). Dynamic integration of classifiers for handling concept drift. *Information Fusion*, *9*(1), 56-68.

Vilalta, R., & Drissi, Y. (2002). A perspective view and survey of meta-learning. *Artificial Intelligence Review*, *18*(2), 77-95.

Wang, H., Fan, W., Yu, P. S., & Han, J. (2003). Mining concept-drifting data-streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 226-235).

Wankhade, K., Hasan, T., & Thool, R. (2013). A Survey: Approaches for Handling Evolving Data-streams. In *International Conference on Communication Systems and Network Technologies (CSNT), 2013* (pp. 621-625).

Wolpert, D. H., & Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1*(1), 67-82.

Yang, H., & Fong, S. (2012). Incrementally optimized decision tree for noisy big data. In *Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications* (pp. 36-44).

Yang, Y., & Mao, G. (2013). A Self-Adaptive Sliding Window Technique for Mining Data-streams. In *Intelligence Computation and Evolutionary Computation* (pp. 689-697). Springer Berlin Heidelberg.

Yelle, L. E. (1979). The learning curve: Historical review and comprehensive survey. *Decision Sciences*, *10*(2), 302-328.

Zhao, X., & Kaufman, A. (2012). Structure revealing techniques based on parallel coordinates plot. *The Visual Computer*, *28*(6-8), 541-551.

Zliobaite, I., Bifet, A., Gaber, M., Gabrys, B., Gama, J., Minku, L., & Musial, K. (2012). Next challenges for adaptive learning systems. *ACM SIGKDD Explorations Newsletter*, *14*(1), 48-55.