Novel Method for Optimizing Performance in Resource Constrained Distributed Data Streams

Rashi Bhalla · Russel Pears · M. Asif Naeem · Farhaan Mirza

Received: date / Accepted: date

Abstract The *Big Data* Era has presented many opportunities for using data mining techniques to discover knowledge patterns across large and diverse collections of data where the volume of data is growing at an exponential rate. Recent approaches to Distributed Data Mining (DDM) have focused on addressing the heterogeneous nature of data sources. However, such approaches do not prioritize the reduction of data communication costs which could be prohibitive in large scale sensor networks where bandwidth is a limited resource. In fact, higher communication and computational costs are the two most prominent problems that have been encountered in heterogeneous distributed environments. Moreover, an effort to decrease the communications load in the distributed environment has an adverse influence on the classification accuracy. Therefore, the research challenge lies in maintaining a balance between *transmission cost*, *computational cost*, and *accuracy*. This paper proposes an algorithm Performance Optimizer in Distributed Stream Mining (PODSM) based on *Bayesian Inference* to reduce the communication volume and resource time in a heterogeneous distributed data mining environment while retaining prediction accuracy. The approach used in this work exploits the past data for calculating statistics and these statistics are then utilized for the new data. In other words, it imparts the ability to learn from experiences.

R. Bhalla
School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology
E-mail: rashi.bhalla@aut.ac.nz

R. Pears
School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology
M. A. Naeem
Department of Computer Science, National University of Computer & Emerging Sciences
F. Mirza
School of Engineering, Computer and Mathematical Sciences, Auckland University of Technology

As a result, our experimental evaluation reveals that a significant reduction in the communication load and an improvement in classification response time can be achieved across a diverse range of dataset types. Reduction of 34.66% was obtained with regard to communication overhead for one of the datasets with huge savings of nearly 27% in resource time. Importantly, instead of showing a negative effect on accuracy, this dataset observes an increment of 0.44% in accuracy.

Keywords Big Data, Bayesian inference, Distributed data stream mining, Heterogeneous distributed data

1 Introduction

The volume of data are exploding in both scientific and commercial domains. *Big Data* is a term which is used for datasets that are larger than traditional data. As the size of raw data is increasing, parallelly data mining algorithms are becoming a necessity. In many application domains data exists in a distributed fashion across different geographical locations which are networked together [14, 6]. Each physical location may have different requirements and thus the nature of the data it collects may differ from its peer nodes in its network thus giving rise to heterogeneity in the data. Such types of distributed databases require mining methods which are distinct from traditional homogeneous distributed databases where the structure is identical across different peer nodes in the network. The imperative of the *Big Data* age is to develop machine learning techniques that can be applied not only to larger static dataset but also should have the ability to process an ever-changing data stream arriving at a rapid pace. Concurrent data processing is done in two scenarios, first if there is a single, high throughput of data or if there are multiple steams of data that in turn generate huge volumes of data [31]. The first situation is dealt with by dividing the data for parallel processing across multiple machines, which is known as parallelized data mining, however, this is not always appropriate for distributed data stream mining as it is often based on the assumption that data originates in a single stream. The approach used in this research addresses the data streams as being heterogeneous and extends it to the case where data arrives continuously in streaming mode.

The advancements in information and communication technology have spurred the appearance of distributed computing environments which consist of large volumes of data and multiple computing components. The objective of Distributed Data Mining (DDM) is to obtain useful information, knowledge, and patterns from distributed data sources, which can further serve the needs of many modern applications [19, 14]. The classical data warehouse based architecture works by uploading the data into the warehouse for centralized processing, as shown in Fig. 1 but as the dataset sizes grew and the model became more complex, there was a need to replace the centralized data mining by parallel and distributed techniques. Moreover, due to the fundamental features of the aforementioned method such as long response times,

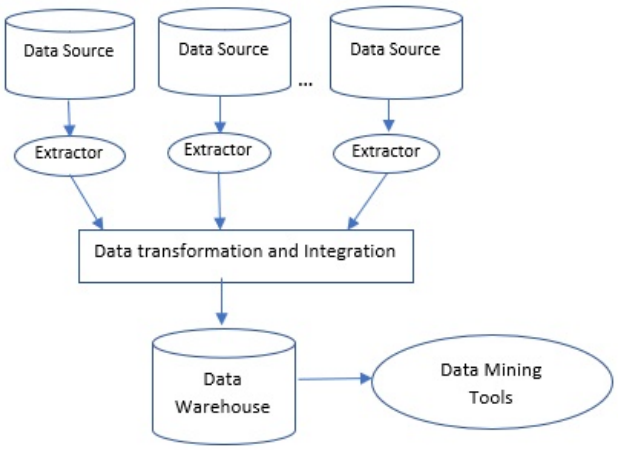


Fig. 1: Data warehouse architecture

lack of proper use of distributed resources, high communication cost, and privacy issues, traditional architectures could not achieve effective outcomes in distributed environments and therefore gave rise to distributed processing architectures. Figure 2 presents a distributed data mining framework where each individual site processes the locally available data and finally local models will be agglomerated to form a global model. The distributed approach is more effective in terms of computation, storage, privacy, and cost of communication [30]. It is being deployed in by applications like astrophysics, sensor networks, weather forecasting, and many others.

Data can be distributed in one of two ways, either they follow homogeneous schemata also known as horizontal partitioning or vertical partitioning, an alias for heterogeneous schemata [26]. Homogeneous schemata are the most common way of collecting data, according to which the features describing the data are same across each distributed site, a multinational company collecting data about its clients in different parts of the world comes under this category. On the other hand, under heterogeneous schemata the attributes present at distributed sites differ from each other for the same set of records. For example, a sensor network in which each sensor collects data related to its set of attributes. Vertically partitioned dataset is still rare but are becoming more prominent and will be discussed in this paper. It is generally believed that mining of heterogeneous distributed data is more challenging and complex [1].

The input to the system is an unbounded stream of data instances arriving at high speed. Data stream mining algorithms that can learn from continuous streams of data records have been developed; they can scan and process each record only once. This ensures that the system does not need to store large collections of records for training. These algorithms are designed so that they can adapt to changes in the underlying patterns of the data streams, also known as Concept Drift [12]. In supervised learning algorithms, it requires

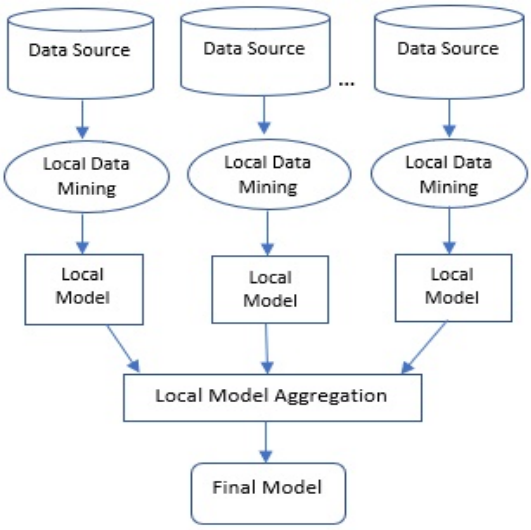


Fig. 2: Distributed data mining framework

that the prediction of the unknown target feature value is computed as soon it is received thus enabling real-time analysis, in this background cutting-edge studies are able to cater the demands of Distributed Data Stream Mining (DDSM) by developing techniques through which the local data mining models generated at each distributed site are combined into the global model while minimizing the data transmission costs and without much effect on accuracy.

A rapid growth in the number of organization operating globally has turned distributed data processing a necessity. Moreover, the broadening scale of usage of sensor networks has revived distributed mining, with an emphasis on Heterogeneous Distributed Data Stream Mining (HDDSM), since these applications analyze data streams originating from different sources. An example that can be cited here is of a Congestion Prediction Model which uses data from different data sensor nodes like weather, traffic, social media [1]. This research demonstrates how the algorithm presented here improves the performance of distributed stream mining in a number of different scenarios where resources are limited. Consequently, productivity gained by reducing communication burden and resource time would immensely benefit sensor networks, thereby increasing the life of nodes and thus, enhancing the efficiency of sensor networks.

The objective of this research is to propose a novel method, Performance Optimizer in Distributed Stream Mining (PODSM) for strengthening the performance of HDDSM. It has been developed by extending the work done in [12, 13] for distributed data stream scenarios. The main contributions made by this research are as follows:

- A method that uses Bayes' theorem for predicting the level of confidence of peer distributed nodes.
- A data transmission protocol that reduces the data transmission volume in a DDSM context while maintaining similar accuracy levels.
- We demonstrate the capability of the algorithm by achieving significant improvements in resource time.
- We augment the proficiency of local models by building a feedback system that helps the distributed nodes in making decisions.

The rest of the paper is organized as follows. The next section will elucidate the previous work in the field of distributed data mining. Section 3 describes the architecture for DDSM. In Section 4 we will illustrate the performance of our algorithm based on experimental evaluation by comparing it with a state-of-art approach. Finally, Section 5 will conclude the paper and provides pointers for future research.

2 Related work

The literature abounds with research related to Distributed Data Mining (DDM); it is the field that is attracting the attention of burgeoning number of scholars. A data matrix can be divided in two ways: either by row resulting in horizontal parallelism or by column thus enabling vertical parallelism. The former approach has been used in [31], where a broadcast message is sent to all the sites for classification of an unlabelled instance, finally, a classification label with a majority prediction is selected. On the other hand, a novel method Vertical Hoeffding Tree (VHT) that parallelizes decision tree construction via vertical parallelism is proposed by Kourtellis, Morales, Bifet and Murdopo [22].

The basic procedure for creating a DDM algorithm starts with performing an analysis of the local data available at the distributed sites and then amalgamating the local models in order to obtain the global model [20]. The authors in [24] present the comparative study of four different models namely centralized database/centralized data mining, centralized database/parallel data mining, centralized database/distributed mining, and lastly their own developed method Multi-Agent Hierarchical Data Mining (MHDM). The database is centralized in the former three approaches while on the other hand, data is distributed over local sites in the fourth approach where local agents extract knowledge from the database and send the mined results to the main agent for aggregation. As discussed earlier, data can typically be distributed either in a horizontal or vertical distribution. The presence of heterogeneously partitioned data is still scarce but is becoming more prevalent and prominent [30]. Classification of vertically distributed data is the focus of this research.

The authors in [22] implemented vertical tree partitioning by splitting the data based on its attributes. The Algorithm consists of two components, the model aggregator, which holds the current model i.e. the tree produced so far

and the local statistics which contains the statistics n_{ijk} for a set of attribute-value-class triplets. In order to train the model, VHT break down the record into its constituent features, attaches class label to each and sends them independently for calculation of local statistics. The centralized classifier produced by this method requires the availability of all the attributes of a record to be classified at a single location.

In vertical distribution, each node gathers values for a different set of attributes for every record. Apparently, each site has different view of the data and it is assumed that vertical fragments of the record can be joined with the help of at least one unique key feature that all the sites share. Past approaches for classifying vertically distributed databases tend to join the models trained at local site in order to obtain a centralized classifier by merging the outputs of local sites. There is a trade-off between the accuracy and the amount of data transmitted to a central location. Chen, Sivakumar and Kargupta [9] presents an approach of learning a Bayesian Network from heterogeneous distributed data with selected data transmission. Each local site learns a Bayesian network based on the local data and identifies the records that are evidence of coupling between local and non-local variables, thereby sending only a subset of these. Subsequently, another Bayesian network is constructed at the central site by using the selective transmission by the local sites and therefore, a collective Bayesian Network is produced by mingling local and central models.

Park, Ayyagari, and Kargupta proposed a technique [27] to mine data effectively in a heterogeneous environment while distributing the computational load and reducing the communication cost. The proposed method constructs distributed decision tree learning where each site builds a local decision tree and the global site constructs its model based on instances transmitted by the local sites. Setting a confidence threshold value helps in identifying instances that contain the most information about the cross terms. Thus, records that are classified with a confidence lower than the threshold by the local sites are transmitted with the local models to the central site. This offers a way to maintain a balance between the transmission cost and accuracy.

A two stage process of creating an ensemble architecture is a more general way of mining vertically distributed data, where local classifiers are produced at the local sites in the first phase while in the following stage, the local classifications are combined to generate the final output. The methodology is capable of improving the prediction accuracy [23] as the prediction of new incoming instances is based on the combination of the individual classifier's predictions and thus can be employed in data stream environments. Ensembles of classifiers are a powerful tool which has led to substantial improvements by integrating the outputs of multiple classifiers. This framework can provide fruitful results when applied to heterogeneous distributed data mining situations because they have a pool of classifiers that differ in terms of overall performance [32]. There are several ways to merge the local classification for ensemble learning in order to obtain the final output. The most common method of merging the classification of local classifiers is Aggregation which selects the classification with maximum confidence as the final outcome. Another method for classification

depicted in [3] is based on deriving the combined probability based on utilizing the posterior probabilities computed by the individual classifiers. Skillicorn and McConnell [30] came up with a technique called attribute ensembles in which the local predictions based on the local attributes are sent to a central site where the combination take place by voting or averaging to produce a final prediction while [32] suggests a method to obtain the classification based on order statistics. A more recent approach for data stream environments [12] that does not involve centralization of data from the distributed nodes forms the basis of this research and has been discussed in detail below.

2.1 Background and problem definition

As discussed above, Park proposed a method for classifying vertically partitioned data in [27], under which the distributed sites sent only unconfident[1] records to the centralized site and subsequently, the centralized site processes only those records which are declared as unconfident by all the sites. This allows the centralized site to capture the cross-terms or relationship present between all the sites, cross-terms are the patterns which can only be detected by combining views of two or more vertical partitions. Therefore, the transmission of other records for which only a subset of distributed sites are unconfident and are not processed by the centralized site serve no purpose and results in uneconomic communication. Subsequently, Ben et al. came up with an architecture named Hierarchical Distributed Stream Miner (HDSM) [12] which has its basis in capturing the cross-terms between the subset of sites. Instead of a single centralized trouble site, it builds a hierarchy of trouble sites, where each trouble site is formed by pairing two source sites or the distributed nodes. In HDSM, each primary site constructs a local classifier based upon its view of data and identifies trouble records that is, the records which were classified with a confidence value less than the threshold. This method allows the creation of multiple trouble sites each receiving trouble records from the subset of primary sites. Every trouble site is of certain order depending upon its level on the hierarchy as such primary sites are of order 0 and the order keeps on increasing on moving down the hierarchy. The system is initialized with a minimal structure consisting of only the primary sites and the Aggregator with no trouble sites. The Aggregator site in this architecture receives the classification results from all the sites and generates the final output. The classification result consists of a classification label with a confidence value of that classification. A trouble site is created only when any two primary sites reach an agreement greater than the threshold value. Agreement measures the likelihood that all source sites agree any received record is trouble (refer [13] for more details). The architecture is equipped with three sets of monitors, namely the creation, removal, and the blacklist monitor which are responsible for creating, removing, or blacklisting a trouble site, respectively. The value of accuracy for HIG dataset under Park's experiment and the best HDSM

[1] An alias for low-confidence, signifies confidence lower than the threshold.

experiment reported in [12] were 53.40%, 53.57% respectively while the total weighted transmission were 45.18% and 48.63%. The first 1000 records of these experiment were considered a "warm-up" time and were ignored in all calculations. Moreover, the total weighted transmission considers the proportion of total data volume transmitted and was calculated in terms of weighted communication. A trouble site will process a record only if it receives the record fragment from both of its source sites on being declared as a "trouble record" by both of them, that implies the training set of a trouble site is the collection of merged record containing all the features of its source sites. The communication has narrowed down compared to Park's approach, but the trend is still similar. This can be demonstrated by an example. Suppose A and B are two distributed sites and T_{AB} is the trouble site generated for capturing the cross-terms between A and B, as such Trouble site, T_{AB} receives trouble records from A and B. While processing a record by sites A and B, there can be four cases:

- A and B are confident
- A is confident and B is unconfident
- A is unconfident and B is confident
- A and B, both are unconfident

It is worth mentioning here that the distributed sites are unaware about each other's state that is, a distributed site has knowledge about only its own view while it has no information about other sites attributes, their values, and the confidence configuration. For the first case, since both the sites are confident, there will be no transmission to the trouble site while for the second and third case, the unconfident site will send its record fragment to the trouble site while the confident site will send its classification result to Aggregator suspending communication to trouble site for this particular record. As a result, the trouble site holds record fragment of only one site which is useless to it and this results in unnecessary overhead of communication, thereby decreasing the productivity in terms of computational resource time. Hence, this research put forward a method to utilize the communication which serve no purpose in state of the art approaches.

3 PODSM

3.1 Architecture

The following section elucidates the architecture of the proposed method for classification in a vertically distributed data stream. It aims at curtailing the communication between the sites and enhancing the efficiency in terms of computational resource time, while maintaining the accuracy level. The proposed work is based on the HDSM method which was discussed in previous section. Figure 3 depicts the architecture of the system which consists of a number of primary sites, trouble sites and one aggregator site. Every primary site receives

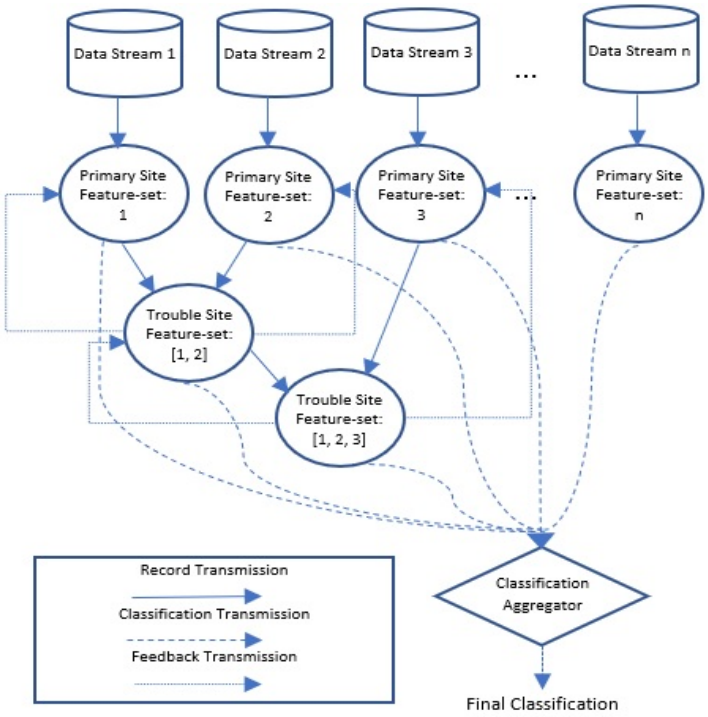


Fig. 3: Pictorial representation of the architecture

input from heterogeneous data streams which differ in terms of feature set. It is assumed that each feature set will contain a common key feature that is not used for classification but allows record fragments and classifications to be merged at trouble sites and the classification aggregator, respectively.

The ultimate motive of refining the productivity in terms of communication and computation has been addressed by targeting the records which are sent by only one unconfident source site. These records have no significance in generating the classifier at the trouble site and result in wastage of communication. The basic working scheme says that if a source site is unconfident for a record, it predicts the probability of confidence of paired source site. If the predicted probability value is greater than the threshold, the unconfident site would assume that the other site is confident and therefore makes a decision not to send the record to the trouble site while in the other case when the prediction value is less than the threshold the unconfident site would send the record. Supposing that the unconfident site decides not to send the record depending upon the prediction value, on the occasion when this assumption made by the unconfident site turns out to be correct would fruitfully results in reduction of communication otherwise it may affect the accuracy.

For calculating the probability by a source site, the algorithm exploits the Bayes' formula, according to which the conditional probability can be

calculated as:

$$P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B \mid A)P(A) + P(B \mid A^{c})P(A^{c})}$$

The formula is based on the expression $P(B) = P(B \mid A)P(A) + P(B \mid A^{c})P(A^{c})$, which simply states that the probability of event B is the sum of the conditional probabilities of event B given that event A has or has not occurred. On relating the above equation in our system shown in Fig. 3. Suppose, if A, B are the two primary sites and A is unconfident for a record, the formula used by A in that case to predict the probability of confidence of B will be as follow:

$$P(B\,conf \mid X = x\ A \sim conf) = \frac{P(X = x\ A \sim conf \mid B\,conf)\,P(B\,conf)}{P(X = x\ A \sim conf \mid B\ conf)\,P(B\,conf) + P(X = x\ A \sim conf \mid B \sim conf)\,P(B \sim conf)}$$

where the terms on the left hand side will give the probability of B being confident when A is unconfident for X = x, where X is the set of attributes at site A and x contribute to values of attributes for a record while the terms on the right side are calculated at the trouble site using the historic data (i.e. a batch of seen records). At the end of this batch of records, the trouble site calculates statistics for the terms mentioned on the right side of the above formula and send those values to the source sites. Thus, the trouble sites are responsible for calculating and transmitting these statistics to their respective source sites. Likewise, when B being unconfident for a record predicts the probability of confidence of A using the above formula but in that event the positions of A and B in the formula gets interchanged. The statistics or feedback from all the trouble sites is then utilized by the source sites when they encounter a trouble record, for predicting the level of confidence of their peer site. To achieve dynamism for the data stream environment, the statistics after being calculated at the end of first batch, are regularly updated on incoming of every record. This makes algorithm adaptive for situations when there is any change in the structure, expressly if there is any addition or removal of trouble site and further adds the flexibility on account of concept drifts in data streams.

3.2 Algorithm

The primary sites generate its local classifier and correspondingly have their local "confidence threshold" value. When a new unlabelled record arrives, it calls a procedure *processRecord* in Algorithm 1. If the local classification confidence of the record is less than the confidence threshold and it belongs to *batch*-1[2] then that particular record is forwarded to the trouble site for calculating historic statistics. On the other hand, if the record does not belong to *batch*-1 and there exists sufficient statistics (received from trouble site as explained above), then the unconfident local site estimates the probability of

[2] First set of records that is used by the trouble site as the historic data for calculation of the conditional probabilities.

the other site being confident. If the computed value is greater than the peer confidence threshold which is set at **0.5** then it holds back the record and does not send it to trouble site, but if peer confidence is less than **0.5**, it then forwards the record to the trouble site. While, if sufficient statistics do not exist, then the record is forwarded to trouble site for calculating statistics.

The reason for setting the peer confidence at **0.5**, is that a site that estimates that if a peer site can make a correct prediction above random chance, then it should allow that peer site's prediction to be forwarded to the final aggregation site. The aggregation site will make a final determination of the class value for such a record based on the site's confident prediction and its perception of a confident prediction from its sister peer site. In case the site's perception of its peer was incorrect then the site's prediction will prevail, else the class value will be determined on the basis of the maximum of the confident predictions received by the site in question and its peer site. Ultimately, setting peer confidence at a modest value of **0.5** ensures that communications overhead is minimized by preventing transmissions involving assessments with a confidence greater than random chance rather than requiring a higher hurdle such as a value in the range [0.5, 1.0). One consequence of setting it at the random chance value is that there is a risk of not identifying cross-terms since such records are not dealt with at the trouble site. In effect, the setting of the threshold represents a trade-off between *classification accuracy* and *communications overhead*.

Each trouble site runs a procedure *forwardTroubleRecord* of Algorithm 1, which shows that if it receives record fragments from all source sites expressly as a result of all its source sites agreeing that a record is trouble, then it processes that record, otherwise when it receives just a fragment of a record from one site then it uses that particular record for updating existing statistics or computes new values in case they do not exist.

As a result, the calculated feedback data are transmitted to the source sites initially at the end of *batch*-1 and then incrementally updated on a record by record basis in order to keep statistics current.

All the sites in the network must send their local classification confidence to the Aggregator which in turn will compute the final prediction depending upon highest confidence in our case, check procedure notifyAggregator for reference. The role of a source site is summarized in Fig. 4.

3.2.1 Illustration of transmission protocol

The Aggregator has all the information required for the classification of every record through the transmission mechanism described in Algorithm 1. The completeness of this transmission protocol can be demonstrated by considering a hierarchy, where p_1 and p_2 are the two primary sites which classify their record segments r_1 and r_2 of record r with key k and produces c_1 and c_2 as their respective classification, and t is the trouble site with c_t representing its classification. There can be four possible cases to elucidate the transmission process. The first possible case is when both the primary sites are confident.

Algorithm 1: Procedures for primary, trouble and aggregator site

```
Procedure processRecord(site, record):
    classify record at site to get the classification-result;
    aggregateClassification(site, record-key, classification-result,
    t-site-forwarding);
    foreach t-site that site may forward to do
        if classification-result confidence < threshold from site to t-site then
            if record belongs to batch-1 then
                forwardTroubleRecord(t-site, record);
            else if feedback data received from t-site for (X = x1, x2, ..., xn) != 0
            then
                calculate the conditional probability of another paired site being
                confident given the present site is unconfident for (X = x1, x2, ...,
                xn);
                if the above estimated classification-result confidence of other site
                < 0.5 then
                    forwardTroubleRecord(t-site, record);
                else
                    quit processing the current record and resume with the next
                    record;
                end
            else
                forwardTroubleRecord(t-site, record);
            end
        end
    end
Procedure forwardTroubleRecord(t-site, record-fragment):
    if t-site has received a record-fragment from all source sites then
        processRecord(t-site, merged-record);
    else if record belongs to batch-1 then
        add record-fragment to a size-limited buffer at t-site;
        calculate historic statistics when (X = x1, x2, ..., xn) where X is the
        feature set;
        transmit it to the source sites as feedback data;
    else
        update the existing historic statistics if they exist otherwise calculate new
        values and transmit it to the source sites as feedback data;
    end
Procedure aggregateClassification(site, record-key, classification-result,
t-site-forwarding):
    log classification-result as received from site for record-key;
    foreach t-site in t-site-forwarding do
        if site forwarded record-key to t-site AND record-key has not been logged as
        unexpected from t-site then
            log that a result for record-key may be expected from t-site;
        else
            log that a result for record-key is unexpected from t-site;
        end
    end
    if all p-sites and expected t-sites have provided a classification-result then
        aggregate all classifications for record-key into a final-classification;
    end
```

In this situation, both the sites classify their record segment with confidence

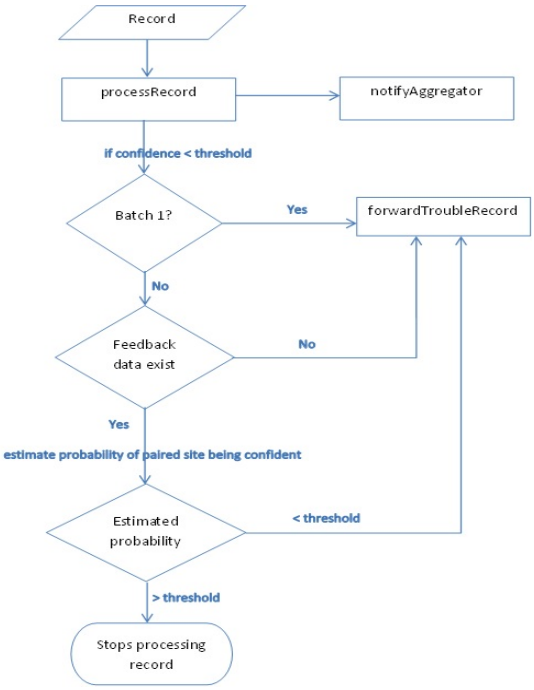


Fig. 4: Flowchart summarizing role of primary site

values greater than the threshold value, in this case both the sites p_1 and p_2 send their classification c_1 and c_2 to the aggregator without forwarding their record fragments to the trouble site. The sites notify the aggregator that they are not forwarding the record to the trouble site by setting t-site-forwarding $= \{t : 0\}$ in the call to *notifyAggregator* in Algorithm 1. When the aggregator starts processing each site's classification, the condition is evaluated to be false for trouble site t as both the sites p_1 and p_2 sent $\{t : 0\}$ in the call so therefore the record is logged as unexpected from t. Ultimately, the aggregator will process the classification received from sites (i.e. from both the primary sites and not expected from trouble site) and produce the final classification. Algorithm 2 outlines the high-level procedure calls involved in this case.

Algorithm 2: Procedure calls for case when both p_1 and p_2 are confident

```
processRecord(p1, r1)
    ≻ aggregatorClassification(p1, k, c1, {t : 0})
processRecord(p2, r2)
    ≻ aggregatorClassification(p2, k, c2, {t : 0})
```

In the second case, p_1 confidently classifies the record while p_2 is not confident in its classification, so in this case if the record r belongs to *batch*-1, then t-site-forwarding will be $\{t : 1\}$ for p_2 and $\{t : 0\}$ for p_1 while if the record does not belongs to *batch*-1 and the estimated probability of confidence of p_1 by p_2 is less than the threshold, the outcome will be again $\{t : 1\}$ for p_2 and $\{t : 0\}$ for p_1, but if the estimated probability is greater than the threshold, the t-site-forwarding will be $\{t : 0\}$ for both p_1 and p_2 (i.e. p_2 will estimate that p_1 is confident and will not transmit the record fragment, thus saving communication). If t-site-forwarding is $\{t : 0\}$ for both the sites, the situation will be same as in case first while if p_2 sends the record fragment to t, the trouble site t will then receive the record segment from only one site making the condition false and therefore t will not be processing the record. While if both the primary sites differ in their decision, the record will be added to the buffer and will be used for calculation of the conditional probabilities, thus logging the classification to be unexpected from t. The Aggregator will produce the final classification from the classification of both the sites and will not wait for t. Algorithm 3 outlines the procedure calls made when record r belongs to *batch*-1 or if the estimated probability of p_1 being confident by p_2 is less than threshold, in other cases Algorithm 2 will be used.

Algorithm 3: Procedure calls for case when p_1 is confident and p_2 is not confident

```
processRecord(p1, r1)
    ≻ aggregatorClassification(p1, k, c1, {t : 0})
processRecord(p2, r2)
    ≻ aggregatorClassification(p2, k, c2, {t : 1})
      forwardTroubleRecord(t, r2)
```

The third possible case is when the primary site p_1 is unconfident and p_2 is confident, which is similar to second case except that the role of the primary site gets interchanged, therefore this case does not need further discussion.

The final and fourth case is when both the primary sites are unconfident. If the record belongs to *batch*-1 then both the sites will transfer their respective record fragment to the trouble site and t-site-forwarding will be $\{t : 1\}$ for both thus making the condition true. The trouble site after receiving the record from both the sites will process the merged record, accordingly, and will declare t-site to be expected. Finally, the aggregator will generate the final classification based on the classification received from both the primary sites p_1, p_2 plus it will also wait for the classification from t making the condition true, Algorithm 4 demonstrates the procedure calls used in this particular case. on the other hand, if the record doesn't belong to *batch*-1, both the primary sites will estimate the probability of confidence of the other paired site using the historic statistics, if the estimated probability value is greater than the threshold, the transmission of the record will be skipped by that unconfident primary site.

If both the sites wrongly estimate their paired site to be confident, as a result both the unconfident primary sites will not send the record to the trouble site following Algorithm 2, whilst if one of the sites made a wrong estimation and the other made the correct prediction, the second and the third case will be used depending on the roles of the primary sites. The last scenario would be if both the sites correctly predict the low confidence of each other, in this case Algorithm 4 will be used, the former two cases will have an effect on accuracy.

Algorithm 4: Procedure calls for case when both p_1 and p_2 are not confident

```
processRecord(p1, r1)
    ≻ aggregatorClassification(p1, k, c1, {t : 1})
      forwardTroubleRecord(t, r1)
processRecord(p2, r2)
    ≻ aggregatorClassification(p2, k, c2, {t : 1})
      forwardTroubleRecord(t, r2)
      /* t has received fragments from all source sites, and therefore processes r */
     ≻ processRecord(t, (r 1 + r 2))
        ≻ aggregatorClassification(t, k, ct, {})
```

Table 1 highlights the comparison of PODSM with recent approaches [27], [13] in terms of communication convention. It summarizes the nature of action under each method for different states (i.e. confident or unconfident) of two distributed sites, A and B in this case. It is important to recall that in HDSM and PODSM, A and B would be paired into a trouble site and transmission will be to that trouble site while in Park's approach all the communication takes place between the distributed sites and a single central trouble site.

3.2.2 Aggregation methods

The final classification outcome by the aggregator can be produced based on several methods. The first one being maximum confidence which has been adopted as the standard aggregation method in the experiment assessment section. Under maximum aggregation, a single site whose classification confidence is highest is the victor [27] and the aggregator chooses the winner site's class as the final selection. The other method of aggregation is Voting, under this approach the aggregator's outclass is based on majority votes from all the primary sites and the final confidence of classification is the mean confidence of all of them [30]. The third method of aggregation is stacking [28], where the classification results from all the primary sites are used as input to other stream classifier, however these stream classifiers may or may not use the same algorithm as the base classifiers. These stream classifiers generate a new classification with a confidence value. The impact of using different aggregation methods is depicted in the next section.

Table 1: Transmission comparison for different states of sites

Site Status		Transmission Status		
Site A	Site B	Parks' Algorithm	HDSM	PODSM
Conf	Conf	Yes	Yes	Yes
Conf	Unconf	Yes	Yes	Maybe
Unconf	Conf	Yes	Yes	Maybe
Unconf	Unconf	No	No	No

4 Experimental evaluation

All experiments were performed with OpenJDK 1.8.0 on a 64-bit Windows running on a 4 × 3.40 GHz Intel Core i5 CPU with 16 GB of memory. We have used Adaptive Random forest (ARF) [17] classifiers at trouble sites while Naïve Bayes classifiers are used at the primary sites and the final classification is selected based on highest confidence aggregation method. There were several configuration parameters that were kept fixed throughout the implementation phase in HDSM [12]. They were size-limit for all windows, all threshold values for creation and removal monitors, Hoeffding bound (δ), and smoothing factor for agreement threshold (θ). Additionally, in PODSM an invariable known as *batch*-1 was given a constant value depending upon the size of the dataset such as, the value of *batch*-1 for HIG dataset was 5000. Trouble factor is a user-configured parameter in the HDSM architecture that determines the proportion of the data stream that will be transmitted to a trouble site in the form of trouble records [13]. The experiments were configured for trouble factor equal to 1, 1.5 and 2. The experiments were performed on 11 datasets out of which nine datasets are real namely, Occupancy (OCC), NASA FLTz (FLT), HIGGS (HIG), Gas Sensor Array Drift (GAS), Gesture Master (GES), Skin NonSkin (SKIN), Parkinson (PARK), Fog release (FOG), Sensorless Drive Diagnosis (SDD)and two are synthetic; RBF-M, SEA-G, these are based on the Radial Basis Function generator (RBF) and SEA generator. Table 2 describes the datasets in terms of feature count, number of primary sites, classes and records count and also list the number of features available per site. The column stream? provides streaming information about the dataset, in some cases the dataset consists of long periods of single class, where shuffling is needed for meaningful assessment while for SDD dataset, there are 11 classes producing several streams, so these streams are interleaved for preserving time-series progression within them. Interleaving was done by drawing records in order from each stream until all records from all the streams are exhausted and the decision for the selection of next stream was made randomly. FLTz data was retrieved from NASA's data portal, and rest of the real world data were retrieved from UCI Machine Learning Repository while the synthetic datasets SEA-G and RBF-M were generated with MOA [5].

Table 2: Properties of datasets used for performance evaluation

Dataset	Feature count	Feature per site	P-site count	Class count	Record count	Stream?
FLT	20	1-3	9	2	25034	Shuffled
OCC	3	1	3	2	20560	Shuffled
HIG	19	3-4	5	2	10000	Stream
GAS	128	8	16	6	13910	Stream
GES	8	1	8	6	63196	Shuffled
SKIN	3	1	3	2	50000	Shuffled
PARK	6	1	6	2	26731	Shuffled
FOG	9	1	9	3	38774	Shuffled
SEN	48	4	12	11	58509	Interleaved
SEA-G	3	1	3	2	100,000	Gradual drift
RBF-M	10	1	10	5	50,000	Moderate Incremental Drift

4.1 Performance assessment metrics

The accuracy and the total volume of transmission reported in terms of percentage (%) are used as metrics to evaluate the performance of the method discussed in this study. The accuracy was evaluated using the confusion matrix produced as part of the classification process. Total transmission (TT) is reported as the record transmission between all the sites and may exceed 100% in cases when record fragments are transmitted through multiple layers of trouble sites. Transmission of site classifications to the Aggregator is not considered, as transmission of class labels and confidence values from the sites is same in both the cases, while being a much less significant cost in comparison to feature transmission. In HDSM, the author reported resource time in terms of Mean Time on Critical Path (MTCP) and Mean Time Between Completions (MTBC). MTCP measures the classification interval for a simple record on average and it is the cumulative CPU time taken to process a record, over the longest running sequences of classifiers, averaged over all the records. While, MTBC measures the mean difference between the execution times of consecutive records, provided each classifier can start processing a record only after it has finished processing the previous record and also after all its source sites have processed the record. Both MTCP and MTBC are expressed in nanoseconds, for more details refer [13]. Table 3 provides a summarised definitions for all the assessment units.

Table 3: Explanation of metrics used for performance assessment.

Metric	Explanation
Accuracy	Classification accuracy.
TT	Total data transmission to trouble sites.
MTCP	Mean Resource Time (CPU) to generate final classification on critical Path.
MTBC	Mean Resource Time (CPU) between successive classification completions.

4.2 Performance comparison

The following section presents the comparative study between HDSM [12] and PODSM. The results obtained by applying Algorithm 1 demonstrate its ability to achieve a substantial drop in communication while maintaining nearly the same accuracy levels. The values for these metrics were computed without taking *batch*-1 into account as *batch*-1 was only used for calculating the first historic statistics and in actual there was no effect of applying this algorithm on *batch*-1. Tables 4 and 5 provide a comparative study for accuracy and total transmission between HDSM and PODSM which clearly depicts decline in communication while obtaining nearly same accuracy except two adverse cases: *Gesture Master and Sensorless drive*. In case of SDD, the drop in accuracy is much higher when compared to savings in terms of communication while in case of GES when trouble factor equal 1.5, a contradictory effect on communication can be seen after applying algorithm, instead of decreasing, the communication is increasing.

Table 4: Accuracy comparison

Dataset	HDSM			PODSM		
	TF 1x	TF 1.5x	TF 2x	TF 1x	TF 1.5x	TF 2x
FLT	**81.18**	87.37	87.62	**80.54**	87.01	87.62
OCC	88.67	**88.78**	92.69	82.34	**88.33**	91.19
HIG	53.90	54.00	**54.48**	54.24	53.58	**54.72**
GAS	**63.04**	73.05	81.49	**60.95**	70.80	81.40
GES	**63.08**	63.01	63.29	**62.97**	63.07	63.29
SKIN	**97.78**	96.68	94.83	**95.97**	95.90	94.83
PARK	**79.53**	92.52	99.59	**77.46**	92.52	99.59
FOG	**85.38**	88.27	89.20	**84.40**	87.90	89.28
SEN	86.34	95.83	**98.00**	71.60	87.86	**97.68**
SEA-G	**75.95**	75.76	75.93	**75.60**	75.76	75.93
RBF-M	**43.11**	45.72	52.76	**41.86**	47.23	52.69

For Occupancy dataset, when trouble factor equals 1, the accuracy drops by 7% but on the other hand the algorithm achieves a huge reduction in communication of approximately 42%. Table 6 summarizes the accuracy for the Occupancy dataset in terms of other metrics; precision, which measures the exactness and recall, which is the measurement for completeness.

However, it can be witnessed that in some cases (SEA-G when TF is 1.5x, 2x) even after applying algorithm PODSM, communication and accuracy remains unchanged. Figure 5 highlights the efficiency of the best case of algorithm which reveals a significant cost reduction can be seen in terms of communication with marginal effect on accuracy, Fig. 5a shows the comparison of accuracy between HDSM and PODSM, while Fig. 5b presents the communication observation.

Table 5: Communication comparison

Dataset	HDSM			PODSM		
	TF 1x	TF 1.5x	TF 2x	TF 1x	TF 1.5x	TF 2x
FLT	**425.31**	632.46	669.66	**369.47**	616.74	669.66
OCC	99.31	**149.68**	307.20	57.70	**108.12**	220.78
HIG	218.12	342.14	**525.52**	181.24	307.36	**343.38**
GAS	**543.31**	850.77	952.87	**419.05**	707.98	892.85
GES	**297.97**	276.86	267.28	**279.56**	353.11	267.28
SKIN	**100.65**	76.35	13.76	**36.95**	57.37	13.76
PARK	**204.99**	311.04	398.46	**150.34**	311.04	398.46
FOG	**341.09**	586.54	769.69	**230.65**	522.34	766.43
SEN	334.88	421.13	**493.83**	294.28	378.64	**463.17**
SEA-G	**51.34**	16.38	6.74	**32.21**	16.38	6.74
RBF-M	**476.33**	688.02	1026.22	**327.35**	684.55	1023.76

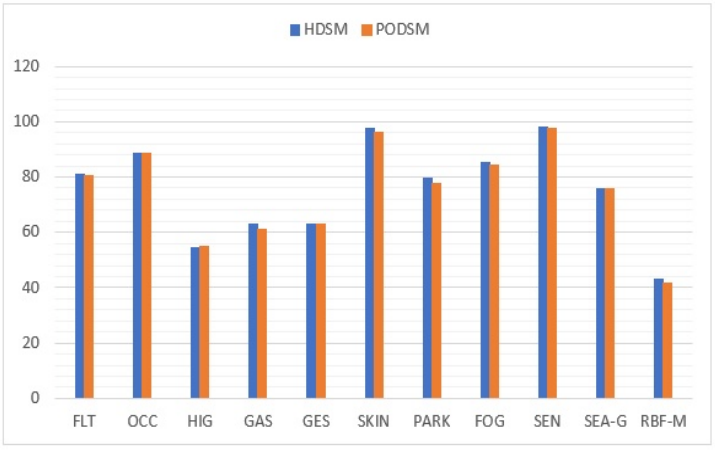
Table 6: Other Accuracy metrics for OCC dataset

TF	HDSM				PODSM			
	Precision		Recall		Precision		Recall	
	Class 0	Class 1	Class 0	Class 1	Class 0	Class 1	Class 0	Class 1
1x	0.93	0.74	0.92	0.77	0.85	0.67	0.93	0.46
1.5x	0.93	0.75	0.93	0.76	0.93	0.74	0.92	0.75
2x	0.95	0.85	0.96	0.82	0.93	0.85	0.96	0.75

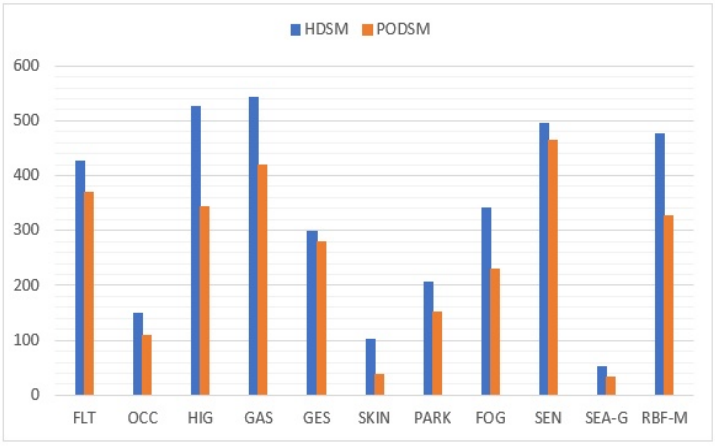
Table 7: Resource time comparison

Dataset	HDSM		PODSM	
	MTCP	MTBC	MTCP	MTBC
FLT	1.80E5	7.05E4	1.67E5	7.06E4
OCC	7.63E4	6.63E4	6.34E4	3.84E4
HIG	1.45E5	6.43E4	1.40E5	4.69E4
GAS	1.85E5	2.63E4	1.67E5	2.17E4
GES	1.33E5	1.31E4	1.38E5	1.50E4
SKIN	3.12E4	2.49E4	2.01E4	8.11E3
PARK	9.47E4	5.34E4	7.86E4	2.23E4
FOG	1.47E5	7.92E4	1.23E5	6.21E4
SEN	1.43E5	6.61E4	1.20E5	1.71E4
SEA-G	4.58E4	2.96E4	4.15E4	2.54E4
RBF-M	2.08E5	6.35E4	1.65E5	1.74E4

Table 7 draws the comparison between HDSM algorithm and PODSM algorithm in terms of resource time when TF equals 1, in almost all the cases an improvement in resource time is observed except for GES, and FLT (only for MTBC). It can be observed that a considerable reduction in resource time has been obtained while maintaining the same accuracy levels, with SEN and RBF-M showing remarkable efficiency.



(a) Accuracy



(b) Communication

Fig. 5: Significance of PODSM

As discussed in previous section, there can be different ways of combining the classification results from all the sites at the aggregator site. The effect of choice of method to produce the final classification on accuracy and total transmission for the SKIN dataset has been depicted in Fig. 6, which has three subgraphs one for each case of trouble factor. In subgraph for TF equals to 1, a major decline in transmission has been achieved with marginal effects on accuracy for all the methods. When TF is 1.5x and aggregation method is Maximum Confidence, transmission witnesses a drop of 24.86% with 0.81% decrease in accuracy, on the other hand, for other two methods there seem to be less or no effect. Similarly for TF 2x, voting shows a small decline in transmission with a minor effect of 0.17% on accuracy while the other two

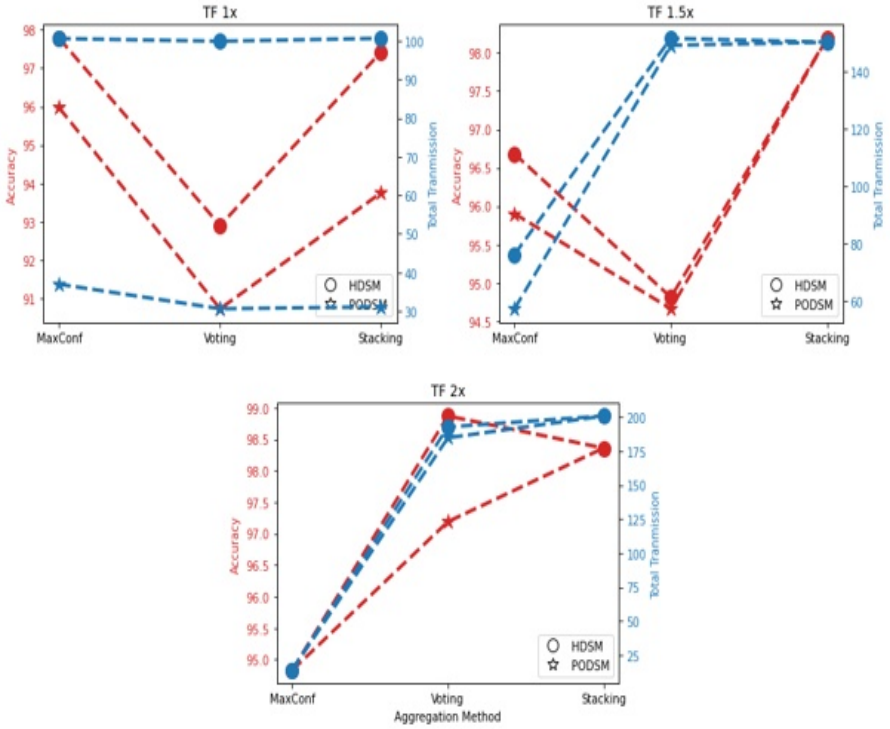


Fig. 6: Accuracy and transmission comparison for other aggregation methods

Table 8: Resource time comparison for different methods

TF	HDSM						PODSM					
	MaxConf		Voting		Stacking		MaxConf		Voting		Stacking	
	MCTP	MTBC	MCTP	MTBC	MCTP	MTBC	MCTP	MTBC	MCTP	MTBC	MCTP	MTBC
1x	**3.12E4**	**2.49E4**	**2.88E4**	**2.19E4**	1.07E5	9.25E4	**2.01E4**	**8.11E3**	**1.74E4**	**7.35E3**	1.00E5	8.87E4
1.5x	2.54E4	1.86E4	5.25E4	4.52E4	1.14E5	9.56E4	2.21E4	1.54E4	4.91E4	4.28E4	1.12E5	9.46E4
2x	1.36E4	5.69E3	5.46E4	4.78E4	**1.36E5**	**1.06E5**	1.28E4	5.50E3	5.14E4	4.48E4	**1.25E5**	**9.63E4**

method witnesses no change. The impact of using different aggregation methods on resource time has been summarized in Table 8, which shows substantial improvement in performance in terms of resource time, bold values represent the best case. This relative analysis portrays the efficiency of the algorithm in achieving substantial decline in communication and resource time while maintaining nearly same level of accuracy for all the methods.

A relative view has been drawn among HDSM and PODSM in Figs. 7 and 8. It displays the timeline for accuracy, data transmission volume and resource time. Data transmission volume is the record feature communication and is reported as the proportion of the total volume of data in the dataset. Total transmission represents the volume of transmission that occur between all sites

while Maximum transmission to one trouble site expresses the maximal volume of transmission to any trouble site. Figure 7 presents the timeline for HDSM, it can be seen that after processing approximately 9000 records a trouble site [2, 3] is generated. The addition of trouble site [2, 3] into the system captures the cross terms between sites 2 and 3, thereby rising accuracy levels while at the same time transmission levels also shows a spike, thus higher accuracy is achieved at the expense of data transmission. Another trouble site [1, [2, 3]] gets created due to high agreement between the sites and thereupon capturing the cross terms. After a while, the recently added trouble site is removed and is blacklisted because the cross terms for which it was created disappeared. Throughout the experiment, the trouble site [2, 3] remains active revealing the presence of cross terms and plays a consistent role in improving the accuracy. The total transmission and resource time shows a spike whenever a trouble site is created while maximum transmission to a trouble site maintains a steady trend.

Meanwhile, a comparison drawn between the above experimentation timeline for HDSM and the visualizations obtained after applying algorithm PODSM is represented in Fig. 8 reveals that the same trouble site [2, 3] is created at the same scales, giving rise to accuracy and total transmission but in this case, maximum transmission to trouble site shows a declining trend after some time because at this stage the source sites initiate using the statistics sent by the trouble sites to predict the probability of confidence of peer site, hence reducing communication and resource time. The trouble site [2, 3] gets removed due to absence of cross terms and thus the agreement between the sites decreases, from this point trouble site transmission turns out to be null till the time a new trouble site is created. The same recurring behaviour can be witnessed as maximum transmission after reaching a height maintains a relatively balanced level for some time because the transmission during this period is used for calculating the statistics and when these statistics or feedback are used, they led to decreasing trend in transmission. Trouble site [1, 2] gets removed since the utilisation of trouble site [1, 2] gets dropped due to absence of cross-terms, so this trouble site gets removed. During the ending phase, presence of cross-terms triggers the creation of a trouble site [1, 3].

A final distinction has been drawn in Fig. 9 between centralized experiment and PODSM with highest confidence as the aggregation method for all cases of trouble factor. Relative ranks have been calculated for accuracy and communication averaged across eleven datasets. Although, centralized experiment is the victor in the race of accuracy but lags behind in maintaining the trade-off between accuracy and communication (calculated as distance from origin) whereas all cases of HDSM shows better performance in building up the balance between the two.

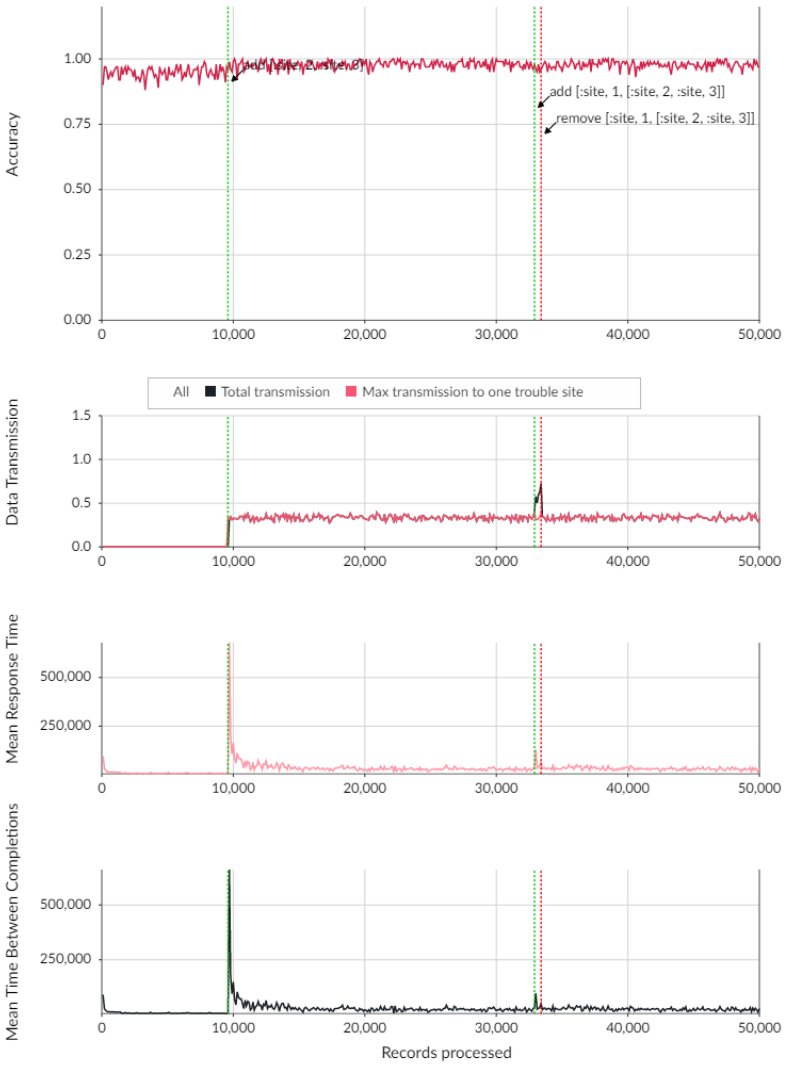


Fig. 7: Experimental timeline for HDSM

4.3 Parameter sensitivity analysis of PODSM

Along with configuring the experiments to different values of trouble factors, the sensitivity of PODSM was evaluated on two other parameters; size of *batch*-1 and periodicity of updating the statistics. Differing values were applied on both of these parameters, while changing value of one parameter the other parameters were stabilized to their standard value (value used in all the

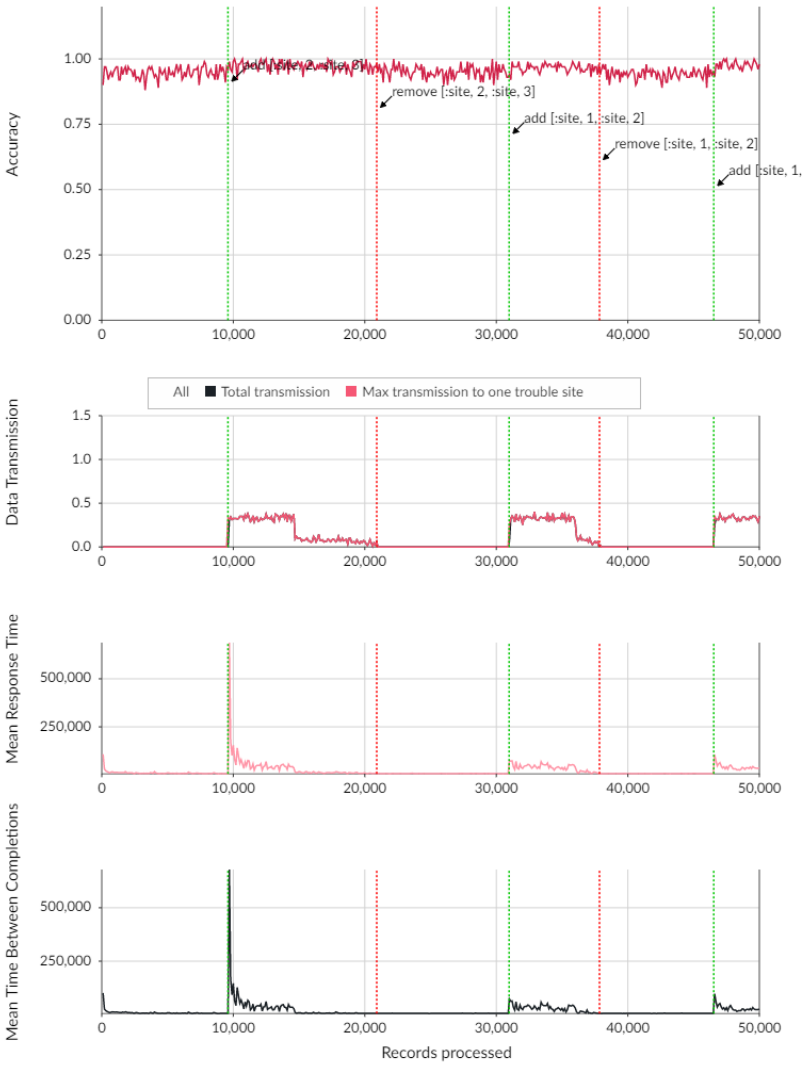


Fig. 8: Timeline visualizations for PODSM

previous experiments). The size of *batch*-1 was increased for some datasets while for others it was decreased, for instance in case of HIG dataset the standard size of *batch*-1 used for all the previous experiments was 5000, which was decreased to 2500 for examining the influence of size of *batch*-1 on accuracy and communication. The results in Table 9 shows the impact of changing the size of *batch*-1 **(Increased / Decreased)** on accuracy and communication. A direct relationship between batch size and communication can be witnessed

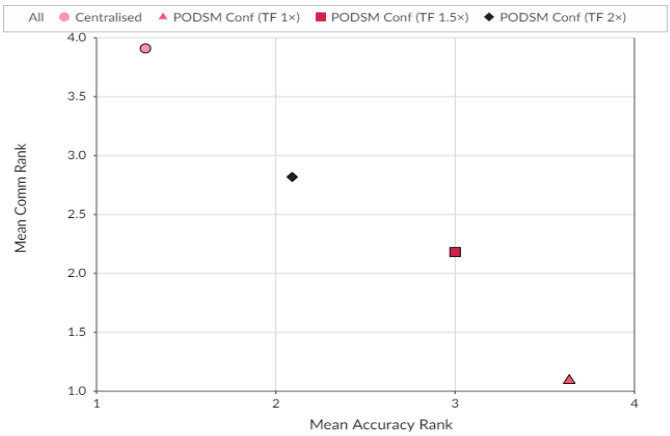


Fig. 9: Comparison between centralized experiment and PODSM

from the results. Once the statistics gets calculated at the end of *batch*-1, they are updated on record basis (as mentioned above). The frequency of refreshing these statistics was switched from record basis to situation when there is any change in the hierarchical structure that is when there is an addition or removal of a trouble site. The readings in Tables 4 and 5 were obtained when the probabilities are updated on record basis after *batch*-1 while Table 10 presents the results when the probabilities are updated only when there is some change in the structure, that is only when a trouble site is added or removed. It can be seen a decrease in communication is achieved compromising accuracy levels marginally.

Results in these tables clearly highlight the trade-off between classification accuracy and communication overhead, therefore the size of *batch*-1 and periodicity of updating probabilities can be adjusted in order to maintain a balance between the two.

4.4 Practical implications and limitations of PODSM

The implication is that the PODSM is a flexible algorithm that can provide fast and accurate classification using lessen communication and computation. The backbone of the algorithm to learn heuristics using historic data and employ them for new data, thus saving resources in reference to communication and time. This probability calculation method has its major application in resource restricted heterogeneous frameworks. The development in Internet of Things (IoT) applications obliges the conjoint servicing of heterogeneous data streams like traffic congestion model, video surveillance systems involving numerous video cameras. The experiment evaluation revealed the potential of PODSM targeted towards enhancing performance of mining of distributed

Table 9: Impact of change in *batch*-1 size on Accuracy and Communication

Dataset	*batch*-1 size	Accuracy			Communication		
		TF 1x	TF 1.5x	TF 2x	TF 1x	TF 1.5x	TF 2x
FLT	Dec	79.81	86.36	86.95	356.62	598.03	676.31
OCC	Dec	80.61	84.38	90.31	39.65	85.28	223.52
HIG	Dec	53.93	54.90	54.04	113.39	277.37	345.49
GAS	Dec	63.23	72.14	80.28	357.34	552.14	739.71
GES	Inc	63.18	63.09	63.24	295.38	253.09	186.49
SKIN	Dec	95.21	95.26	94.86	18.74	32.78	14.24
PARK	Inc	77.40	93.37	99.61	163.91	311.66	398.56
FOG	Dec	82.61	87.14	89.09	200.07	460.79	747.77
SEN	Inc	72.06	88.36	98.07	375.33	626.71	478.94
SEA-G	Dec	75.90	78.46	78.54	43.27	54.54	60.61
RBF-M	Inc	42.88	47.45	53.11	382.47	691.28	1036.11

Table 10: Effect of changing the periodicity of updating the probabilities

Dataset	Trouble Factor					
	1x		1.5x		2x	
	Accuracy	Comm	Accuracy	Comm	Accuracy	Comm
FLT	80.80	390.83	87.20	615.24	87.62	669.66
OCC	84.46	38.41	88.57	120.08	92.31	259.13
HIG	54.32	219.70	53.48	399.16	55.24	445.56
GAS	60.49	474.16	65.58	719.24	80.90	980.16
GES	62.97	279.58	63.07	353.13	63.29	267.28
SKIN	97.62	39.94	96.55	69.14	94.83	13.76
PARK	75.39	170.24	92.52	311.04	99.59	398.46
FOG	84.59	240.67	88.17	561.52	89.33	768.16
SEN	77.40	329.29	92.01	423.65	97.91	497.47
SEA-G	78.24	55.52	75.76	16.38	75.93	6.74
RBF-M	40.78	345.99	45.80	691.91	52.25	1023.48

data streams while maintaining a balance between accuracy, communication, and computation resource time. As mentioned before, Naive Bayes classifiers which are light weight algorithms were used at the primary sites and moreover the focus of this paper has been on decreasing the communication and resource time, therefore, this algorithm can strengthen the sensor networks which consist of distributed sensor nodes with limited capability such as the one used to predict seismic activities like an earthquake or tsunami early warning system. However, it is a flexible algorithm which can be used with many datasets and with other classification methods or may even be applied to other machine learning scenarios such as numeric regression.

It is worth discussing the current constraints and relevance of this algorithm and the scenarios where PODSM may not be suitable. Table 5 shows that in some of the cases even after implementing algorithm, the communication factor remains unchanged, it may be particularly due to the inefficiency of the dataset to learn sufficient statistics that can be applied to estimate the

probability of confidence of paired site and the other case can be when there is substantial fall in accuracy. One way to combat these cases can be to switch to other trouble factor values. The focus of PODSM was on calculating the probability of confidence of paired distributed node and comparing it to static value of 0.5, however some mechanism can be developed to store the past confidence values of sites and thereupon using those values to make a decision about the confidence status of peer distributed mode.

5 Conclusion and future directions

In this paper, we focused on Distributed Data Mining (DDM), particularly in the environment where the distributed sources are heterogeneous in nature, very little research has been done in the heterogeneous arena and still lesser in the streaming scenario. Indeed, mining knowledge from this architecture incurs huge communication cost, therefore, we have presented a method to optimize communication overhead and resource time in vertically distributed data streams while maintaining the similar accuracy level. Our experimentation showed a significant drop in communication (HIG, with communication drop of 34.66% with an increase of 0.44% in accuracy) and an improvement in resource time with minimal effect on accuracy except for two cases (Gesture Master and Sensorless Drive).

However, there may still be opportunities to enhance the performance of mining knowledge from distributed data streams. One approach can be to include confidence values with feature value to calculate the historic statistics. This may help to estimate the confidence value instead of estimating probability of confidence of paired unconfident site. Furthermore, in the current work the algorithm is used for the classification problem, however, it can be adapted to other machine learning techniques as well.

References

1. Akbar A, Kousiouris G, Pervaiz H, Sancho J, Ta-Shma P, Carrez F, Moessner K (2018) Real-time probabilistic data fusion for large-scale iot applications. IEEE Access 6:10015–10027, DOI 10.1109/ACCESS.2018.2804623, URL `https://doi.org/10.1109/ACCESS.2018.2804623`
2. Aronis JM, Kolluri V, Provost FJ, Buchanan BG (1997) The world: Knowledge discovery from multiple distributed databases. In: In Proceedings of Florida Arti Intelligence Research Symposium (FLAIRS-97), pp 337–341
3. Basak J, Kothari R (2004) A classification paradigm for distributed vertically partitioned data. Neural Comput 16(7):1525–44, DOI 10.1162/089976604323057470
4. Bifet A, Gavaldà R (2007) Learning from time-changing data with adaptive windowing. In: Proceedings of the Seventh SIAM International Conference on Data Mining, SIAM, Minneapolis, Minnesota, USA

5. Bifet A, Holmes G, Kirkby R, Pfahringer B (2010) Moa: Massive online analysis. Journal of Machine Learning Research 11:1601–1604, DOI 10.1145/1219092.1219093, URL http://portal.acm.org/citation.cfm?id=1859903
6. Boutaba R, Salahuddin MA, Limam N, Ayoubi S, Shahriar N, Solano FE, Rendon O (2018) A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. Journal of Internet Services and Applications 9:1–99
7. Chan PK, Stolfo SJ (1993) Toward parallel and distributed learning by meta-learning. In: Proceedings of the 2nd International Conference on Knowledge Discovery in Databases, AAAI Press, Washington, DC, AAAIWS'93, p 227–240
8. Chan PK, Stolfo SJ (1995) Learning arbiter and combiner trees from partitioned data for scaling machine learning. In: KDD:Proceedings of the First International Conference on Knowledge Discovery and Data Mining, AAAI, Montréal, Québec, Canada, pp 39–44
9. Chen R, Sivakumar K, Kargupta H (2001) An approach to online bayesian learning from multiple data streams. In: In Proceedings of Workshop on Mobile and Distributed Data Mining, PKDD '01, pp 31–45
10. Chen X, Rowe NC (2011) An energy-efficient communication scheme in wireless cable sensor networks. In: 2011 IEEE International Conference on Communications (ICC), Calhoun, pp 1–5, DOI 10.1109/icc.2011.5963077
11. Cheng J, Ke Y, Ng WK (2008) A survey on algorithms for mining frequent itemsets over data streams. Knowledge and Information Systems 16(1):1–27, DOI 10.1007/s10115-007-0092-4, URL https://doi.org/10.1007/s10115-007-0092-4
12. Denham B (2019) Novel methods for distributed and privacy-preserving data stream mining. Master's thesis, Auckland University of Technology, Auckland, NZ
13. Denham B, Pears R, Naeem MA (2020) Hdsm: A distributed data mining approach to classifying vertically distributed data streams. Knowledge-Based Systems 189, DOI https://doi.org/10.1016/j.knosys.2019.105114, URL https://www.sciencedirect.com/science/article/pii/S0950705119304836
14. Devi SG (2014) A survey on distributed data mining and its trends. International Journal of Research in Engineering & Technology 2(3)
15. Dua D, Taniskidou EK (2017) UCI machine learning repository. URL http://archive.ics.uci.edu/ml
16. Friedman N, Geiger D, Goldszmidt M (1997) Bayesian network classifiers. Machine Learning 29:131–163, DOI https://doi.org/10.1023/A:1007465528199, URL https://doi.org/10.1023/A:1007465528199
17. Gomes HM, Bifet A, Read J, Barddal JP, Enembreck F, Pfharinger B, Holmes G, Abdessalem T (2017) Adaptive random forests for evolving data stream classification. Mach Learn 106(9-10):1469–1495, URL https://doi.org/10.1007/s10994-017-5642-8

18. Hulten G, Spencer L, Domingos P (2001) Mining time-changing data streams. In: In Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, San Francisco California, p 97–106, DOI 10.1145/502512.502529
19. ikhale RC (2016) Study of distributed data mining algorithm and trends. IOSR Journal of Computer Engineering 8:41–47
20. Kargupta H, Byung-Hoon, Hershberger D, Johnson E (1999) Collective data mining: A new perspective toward distributed data analysis. In: Advances in Distributed and Parallel Knowledge Discovery, MIT Press, Cambridge, MA, United States, pp 133–184
21. Khedo KK, Perseedoss R, Mungur A (2010) A wireless sensor network air pollution monitoring system. International Journal of Wireless & Mobile Networks 2(2):31–45, DOI 10.5121/ijwmn.2010.2203, URL https://doi.org/10.5121/ijwmn.2010.2203
22. Kourtellis N, Morales GDF, Bifet A, Murdopo A (2016) Vht: Vertical hoeffding tree. In: 2016 IEEE International Conference on Big Data (Big Data), IEEE Computer Society, Los Alamitos, CA, USA, pp 915–922, DOI 10.1109/BigData.2016.7840687
23. Krawczyk B, Minku LL, Gama J, Stefanowski J, Woźniak M (2017) Ensemble learning for data stream analysis: A survey. Information Fusion 37:132–156, DOI https://doi.org/10.1016/j.inffus.2017.02.004, URL https://www.sciencedirect.com/science/article/pii/S1566253516302329
24. Moghadam AN, Ravanmehr R (2017) Multi-agent distributed data mining approach for classifying meteorology data: case study on iran's synoptic weather stations. International Journal of Environmental Science and Technology 15:149–158
25. Pal V, Yogita, Singh G, Yadav RP (2015) Effect of heterogeneous nodes location on the performance of clustering algorithms for wireless sensor networks. Procedia Computer Science 57:1042–1048, DOI https://doi.org/10.1016/j.procs.2015.07.376, URL https://www.sciencedirect.com/science/article/pii/S1877050915019055, 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015)
26. Park BH, Kargupta H (2003) Distributed data mining: Algorithms, systems, and applications. In: The Handbook of Data Mining, Lawrence Erlbaum Associates, Mahwah, United States, pp 341–358
27. Park BH, Ayyagari R, Kargupta H (2001) A fourier analysis based approach to learning decision trees in a distributed environment. In: Proceedings of the 2001 SIAM International Conference on Data Mining, SIAM, Chicago, IL, USA, pp 1–22, DOI 10.1137/1.9781611972719.19, URL https://epubs.siam.org/doi/abs/10.1137/1.9781611972719.19
28. Parker B, Mustafa AM, Khan L (2012) Novel class detection and feature via a tiered ensemble approach for stream mining. In: 2012 IEEE 24th International Conference on Tools with Artificial Intelligence, vol 1, pp 1171–1178, DOI 10.1109/ICTAI.2012.168

29. Rokach L (2010) Ensemble-based classifiers. Artificial Intelligence Review 33(1-2):1–39, DOI 10.1007/s10462-009-9124-7, URL https://doi.org/10.1007/s10462-009-9124-7
30. Skillicorn D, Mcconnell S (2008) Distributed prediction from vertically partitioned data. Journal of Parallel and Distributed Computing 68(1):16–36, DOI https://doi.org/10.1016/j.jpdc.2007.07.009, URL https://www.sciencedirect.com/science/article/pii/S0743731507001396, parallel Techniques for Information Extraction
31. Tennant M, Stahl F, Rana O, Gomes JB (2017) Scalable real-time classification of data streams with concept drift. Future Generation Computer Systems 75:187–199, DOI https://doi.org/10.1016/j.future.2017.03.026, URL https://www.sciencedirect.com/science/article/pii/S0167739X17304685
32. Tumer K, Ghosh J (2000) Robust order statistics based ensembles for distributed data mining. In: Kargupta H, Chan P (eds) Advances in Distributed and Parallel Knowledge Discovery, AAAI/MIT Press, pp 185–210
33. Urmela S, Malaiyappan NM (2017) Approaches and techniques of distributed data mining : A comprehensive study. International journal of engineering and technology 9:63–76
34. Weinberg AI, Last M (2017) Interpretable decision-tree induction in a big data parallel framework. Int J Appl Math Comput Sci 27(4):737–748, DOI 10.1515/amcs-2017-0051, URL https://doi.org/10.1515/amcs-2017-0051