# Stream Data Mining and Comparative Study of Classification Algorithms

**Ms. Madhu S. Shukla*, Mr. Kirit R. Rathod****
*(PG-CE Student, Department of Computer Engineering),
(C.U.Shah College of Engineering and Technology, Gujarat, India)
** (Assistant Professor, Department of Computer Engineering),
(C.U.Shah College of Engineering and Technology, Gujarat, India)

## ABSTRACT

**Stream Data Mining is a new emerging topic in the field of research. Today, there are number of application that generate Massive amount of stream data. Examples of such kind of systems are Sensor networks, Real time surveillance systems, telecommunication systems. Hence there is requirement of intelligent processing of such type of data that would help in proper analysis and use of this data in other task even. Mining stream data is concerned with extracting knowledge structures represented in models and patterns in non stopping streams of information [1]. Such massive data are handled with software such as MOA (Massive Online Analysis) or other open sources like Data Miner. In this paper we present some theoretical aspects of stream data mining and certain experimental results obtained on that basis with the use of MOA.**

**Keywords**— Stream, Stream Data Mining, Intelligent-processing, MOA (Massive Online Analysis), Continuous Data.

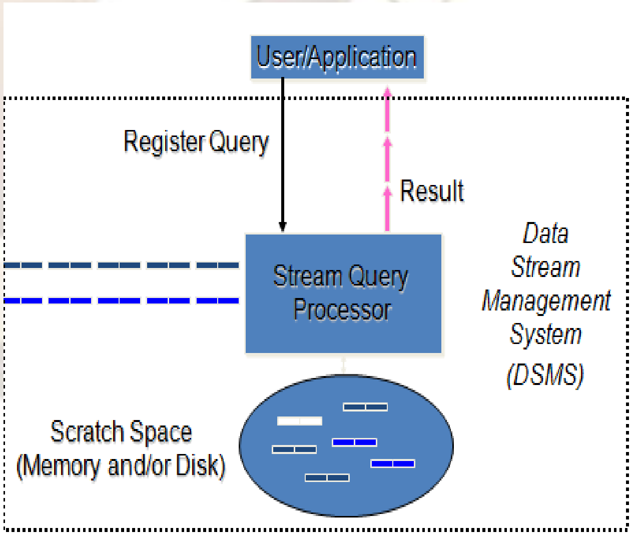
## I. INTRODUCTION

Recently new classes of applications are up-coming rapidly, that basically deal with generating stream data as output. Stream data is continuous, ordered, changing, fast, huge amount of data. Such data are so huge and continuously changing that even one look at entire data becomes difficult. Such systems are, any application that deals with Telecommunication calling records Business: credit card transactions, Network monitoring and traffic engineering, Sensor, monitoring and surveillance, Security monitoring, Web logs and page click streams. Methods with which we are trying to find knowledge or Specific patterns are called Stream Data Mining. Certain characteristics of stream data are as follows:

- Huge volumes of continuous data, possibly infinite.
- Fast changing and requires fast, real-time response.
- Data stream captures nicely our data processing needs of today.
- Random access is expensive—single linear scan algorithm (can only have one look)
- Store only the summary of the data seen so far.
- Most stream data are at pretty low-level or multi-dimensional in nature, needs multi-level and multi-dimensional processing.

As said earlier it is a huge amount of data, so in order to perform certain analysis, we need to take some sample of that data so that processing of stream data could be done with ease. These samples taken should be such that whatever data comes in the portion of sample is worth analyzing or processing, which means maximum knowledge is extracted from that sampled data.

In this paper some sampling techniques are briefed in section II. Some classification algorithms are discussed in section III. And their implementation results in section IV. Conclusions are discussed in section V and references covers section VI.


User/Application
Register Query
Result
Stream Query Processor
Data Stream Management System (DSMS)
Scratch Space (Memory and/or Disk)

**(Fig 1.1: Processing of Stream data)**

## II. FOUNDATIONS AND SAMPLING TECHNIQUES.

Foundation of stream data mining is based on statistics, complexity and computational theory

[2].Sampling refers to the process of probabilistic choice of a data item to be processed or not. Sampling is an old statistical technique that has been used for a long time. Boundaries of the error rate of the computation are given as a function of the sampling rate. Very Fast Machine Learning techniques [4] have used Hoeffding bound to measure the sample size according to some derived loss functions [3].

Some of the methods for stream data mining are:

1. Sampling: Idea of representing large data set by a small random sample of the data elements.
2. Sketching: Building a summary of data stream using a small amount of memory.
3. Load Shedding: Process of eliminating a batch of subsequent elements from being analyzed.
4. Synopsis Data Structures: Small space, approximate solution to massive data set. Summarizing techniques are used like Wavelet, Histogram, and Aggregation.
5. Sliding Window: Advance technique. Detailed analysis over most recent data items and over summarized versions of older ones.

1. Sampling: Idea of representing large data set by a small random sample of the data elements.

In this technique rather than dealing with entire data, sample of stream are taken at regular interval. To obtain an unbiased sampling of data we need to know the length of stream in advance, but incase if this is not possible then a little modified technique is used. It is called RESERVOIR Sampling, in this technique unbiased random sample of 's' element are taken without replacement. Basic fundamental behind this approach is, a sample of size at least 'S', called reservoir is kept from which a random sample of size s can be generated. Basic drawback of this approach is, when the reservoir is large it can be very costly to generate this sample's'.

2. Sketching: Building a summary of data stream using a small amount of memory.

Sketching [3] is the process of randomly projecting a subset of the features. It is the process of vertically sampling the incoming stream. Sketching has been applied in comparing different data streams and in aggregate queries. The major drawback of sketching is that of accuracy. It is hard to use it in the context of data stream mining. Principal Component Analysis (PCA) would be a better solution that has been applied in streaming applications [5].

3. Load Shedding: Process of eliminating a batch of subsequent elements from being analyzed.

Load shedding refers [6] to the process of dropping a sequence of data streams. Load shedding has been used successfully in querying data streams. It has the same problems of sampling. Load shedding is difficult to be used with mining algorithms because it drops chunks of data streams that could be used in the structuring of the generated models or it might represent a pattern of interest in time series analysis.

4. Synopsis Data Structures: Small space, approximate solution to massive data set.. Summarizing techniques are used like Wavelet, Histogram, and Aggregation.

Creating synopsis of data refers to the process of applying summarization techniques that are capable of summarizing the incoming stream for further analysis. Wavelet analysis [25], histograms, quintiles' and frequency moments [5] have been proposed as synopsis data structures. Since synopsis of data does not represent all the characteristics of the dataset, approximate answers are produced when using such data structures.

Aggregation is the process of computing statistical measures such as means and variance that summarize the incoming stream. This aggregated data could be taken by the mining algorithm. The problem with aggregation is that it does not perform well with highly fluctuating data distributions.

5. Sliding Window: It is an advance technique. It deals with detailed analysis over most recent data items and over summarized versions of older ones. The inspiration behind sliding window is that the user is more concerned with the analysis of most recent data streams. Thus the detailed analysis is done over the most recent data items and summarized versions of the old ones. This idea has been adopted in many techniques in the undergoing comprehensive data stream mining system.

## III. MINING TASK

Stream mining task includes task like Classification, Clustering and Mining Time-Series Data.

In this paper, we will discuss some of the algorithms that are used for classification of stream data and their comparison based on their experimental results.

Classification generally is a two step process consisting of learning or Model Construction (where a model is constructed based on class labeled tuples from training set) and classification or Model Usage (where the model is used to predict the class labels of tuples from new data sets).

Algorithms involving with classification of stream data mining are Naive Bayesian, Hoeffding tree, VFDT (Very Fast Decision Tree) and CVFDT (Concept Adapting Very Fast Decision Tree). We will discuss working of these algorithms in detail here.

**3.1Naive Bayesian:**

In simple terms, a naive Bayes classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature, given the class variable. For example, a fruit may be considered to be an apple if it is red, round, and about 4" in diameter. Even if these features depend on each other or upon the existence of the other features, a Naive Bayes classifier considers all of these properties to independently contribute to the probability that this fruit is an apple.

This is very common approach used for classification. Hence forth the name is used as Naive Bayesian.

Algorithm:

**Step 1: Assumption**

**D -** Training set of tuples and class labels

**X**=($x_1$,$x_2$,…,$x_n$) , **Attributes** – $A_1$,$A_2$,…,$A_n$ ,

**Classes** – $C_1$,$C_2$,…,$C_m$

**Step 2: Maximum Posteriori Hypothesis**

$P(Ci|X) > P(Cj|X)$ for $1 \leq j \leq m, j \neq i$

$P(Ci|X)= P(X|Ci)P(Ci)/P(X)$ (By Bayes' Theorem)

**Step 3: Prior Probability of X**

**Step 4: Class Conditional Independence** (Naive's Assumption)

$P(X|Ci) = \prod P(x_k|Ci)$ (k=1 to n) $= P(x_1|Ci) * P(x_2|Ci) * \ldots * P(x_n|Ci)$

**Step 5: Classifier Prediction**For tuple X, the class is Ci if and only if

$P(X|Ci)P(Ci) > P(X|Cj)P(Cj)$ for $1 \leq j \leq m, j \neq i$ .

**Example: Naive Bayesian Classifier**

| ROLLNO | BRANCH | BUYS_COMPUTER |
|---|---|---|
| A1 | CE | YES |
| B1 | CE | YES |
| A2 | CE | YES |
| A3 | CE | YES |
| B2 | IT | NO |
| B3 | IT | NO |

**(Fig 1.2: Naive Bayesian Example)**

**Performance Analysis**

**<u>Advantages</u>**

- Minimum error rate
- High accuracy and speed when applied to large databases.
- Incremental

**<u>Limitations</u>**

- Can't handle concept drift

**3.2 Decision Tree Approach: Hoeffding Tree Classifier**

In Hoeffding algorithm, classification problem must be defined. Classification problem is a set of training examples of the form (a, b), where 'a' is a vector of d attributes and 'b' is a discrete class label. Our goal is to produce a model b= f (a) such that it provides and predicts the classes y for future examples x with high accuracy. Decision tree learning is considered one of the most effective classification methods. By recursively replacing leaf node with test nodes, starting at the root we can learn a Decision trees. In decision tree each node has a test on the attributes and each branch gives possible outcome of the test and each leaf contain a class prediction. Before processing starts, data is first stored into main memory. After starting learning process for complex trees it is expensive to repeatedly read data from secondary memory. So our aim is to design decision tree learners than read each example at most once, and use a small amount time to process it. First key role is to find the best attribute at a node and for that consider only some training examples that pass through that nodes. Second choose the root and then expensive examples are checked down to the corresponding leaves and used to choose the attribute there, and so on. How many examples are required at each node is decided by Hoeffding bound after continuous use. Taken a random variable a and its range is R. We have n observation of a. Now find mean of a (), so Hoeffding tree bound states that with probability 1-δ, the true mean of a is at least - ε. where Hoeffding bound ε [7] .

$$\varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}$$

Algorithm:

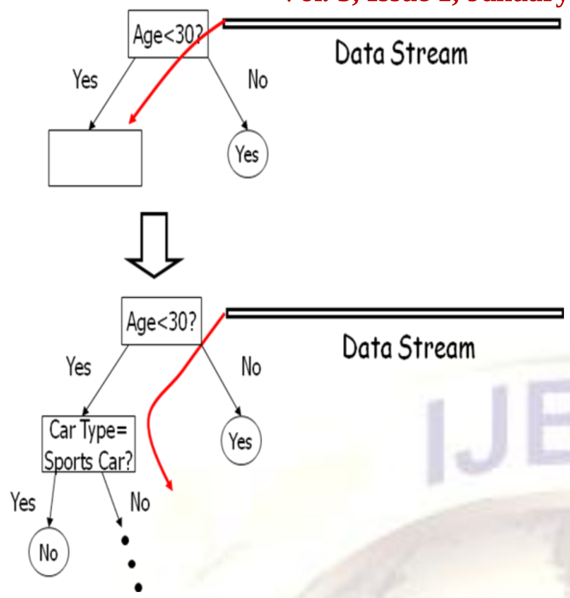
**Step 1:** Calculate the information gain for the attributes and determine the best two attributes.

Pre-pruning: consider a "null" attribute that consists of not splitting the node.

**Step 2:** At each node, check for the condition.

**Step 3:** If condition is satisfied, create child nodes based on the test at the node.

**Step 4:** If not, stream-in more examples and perform calculations till condition is satisfied.



(Fig:1.3 Hoeffding Tree Example)

**Performance Analysis**

**Advantages**

- High accuracy with small sample
- Multiple scans of the same data never performed
- Incremental
- Can classify the data while growing

**Limitations**

- Time consuming for splitting attribute selection
- Can't handle concept drift

**3.3 VFDT Algorithm**

Algorithm:
Input: δ desired probability level
Output: τ a Decision Tree.
In it: τ ← Empty Leaf (Root)

**Step** 1.While (TRUE)
**Step** 2. Read next example.
**Step** 3. Propagate Example through the tree from the root till a leaf.
**Step** 4. Update sufficient statistics at leaf .
**Step** 5. If leaf (number of examples) > Nmin.
**Step** 6. Evaluate the merit of each attribute.
**Step** 7. Let A1 the best attribute and A2 the second best.
**Step** 8. Let ε be the Hoeffding Bound.
**Step** 9. If G (A1)-G (A2) > ε.
**Step** 10. Install a splitting test based on A1.
**Step** 11. Expand the tree with two descendant leaves

**Explanation**:

Sometimes more than one attributes have similar attribute values. In that case choosing best attribute is quite critical. We have to decide appropriate value between them with high confidence.VFDT can decide and solve this problem. When there is effectively a tie and split on the current best attribute if difference between the best split candidate all others is less than G (.) and G (.) < τ. where τ is a user-defined threshold.

G computation: Inefficient to calculate G for every new data set, because at specific point it is hard to split best attributes. Users specify minimum amount of new data examples, 'Nmin' that must be calculated at a leaf before G is computed. This mechanism incrementally takes less amount of global time which was spent on G computations. VFDT making learning work as fast as classify dat-sets .

**Performance Analysis**

**Advantages**

- VFDT is better than Hoeffding tree in terms of time, memory and accuracy.
- So VFDT takes an advantage after 100k to greatly improve accuracy.

**Limitations**

- Concept of drift is not handled in VFDT.

**3.4CVFDT**

Algorithm:

**Step** 1.Alternate trees for each node in HT started as empty.
**Step** 2. Process Examples from the stream indefinitely.
**Step** 3. For Each Example (x, y),
**Step** 4. Pass (x, y) down to a set of leaves using HT and all alternate trees of the nodes (x, y) pass Through.
**Step** 5. Add(x, y) to the sliding window of examples.
**Step** 6.Remove and forget the effect of the oldest Examples, if the sliding window over flows.
**Step** 7. CVFDT Grows.
**Step** 8. Check Split Validity if f examples seen since last checking of alternate trees.
**Step** 9. Return HT.

**Explanation:**

- CVFDT is an extended version of VFDT which provides same speed and accuracy advantages but if any changes occur in example generating process, it provides the ability to detect and respond.
- CVFDT uses sliding window of various dataset to keep its model consistent.
- Most of systems need to learn a new model from scratch after arrival of new data. Instead, CVFDT continuously monitors the quality of new data and adjusts those that are no longer correct. Whenever new data arrives, CVFDT incrementing counts for new data and decrements counts for oldest data in the window.
- However If the concept is changing, some splits examples that will no longer appear

best because new data provides more gain than previous one. Whenever this phenomenon occurs, CVFDT creates alternative sub-tree to find best attribute at root. Each time new best tree replaces old sub tree and it is more accurate on new data.

## IV.COMPARISION:

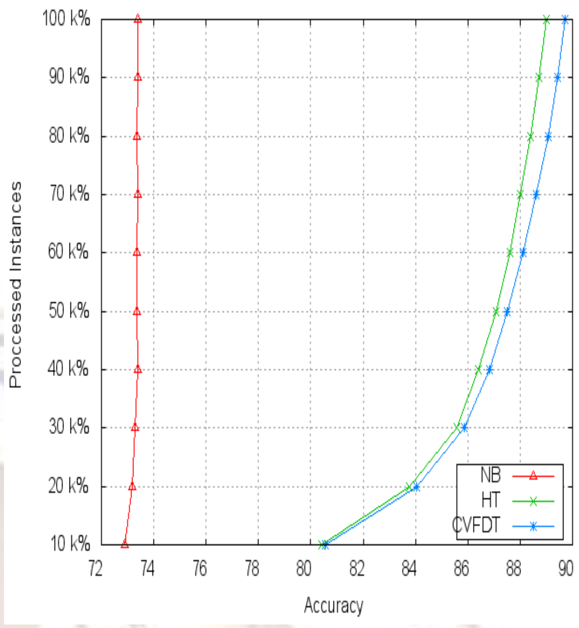
So the comparison shows that CVDFT is best algorithm for classification and also it handles concept drifts. This result is generated by using MOA software. Here, the criterion for evaluation is accuracy of all the algorithms compared:

### 4.1 Comparison of the CSV files generated.

| Learning evaluation instances | Naive Bayesian | Hoefdding Tree | CVFDT |
|---|---|---|---|
| 10000 | 72.91 | 80.4 | 80.56 |
| 20000 | 73.17 | 83.74 | 84.06 |
| 30000 | 73.27 | 85.57 | 85.86 |
| 40000 | 73.38 | 86.4 | 86.83 |
| 50000 | 73.35 | 87.07 | 87.49 |
| 60000 | 73.36 | 87.60 | 88.11 |
| 70000 | 73.38 | 88.02 | 88.61 |
| 80000 | 73.363 | 88.40 | 89.06 |
| 90000 | 73.39 | 88.70 | 89.43 |
| 100000 | 73.38 | 88.99 | 89.73 |

**(Fig 1.4: Comparison of CSV Files)**

### 4.2 Online Comparison With GNUPlot:



**(Fig 1.5: Comparision Chart)**

## VI.CONCLUSION

In this paper, we discussed about theoretical aspects and practical results of stream data mining classification algorithms. In these classification algorithms, Hoeffding trees spend small amount of time for learning. Hoeffding tree does not show any similarity with batch trees. In real world scenario we have a limited amount of hardware resources, despite this it analyzes and generates results with high accuracy. In data mining Systems VFDT is based on Hoeffding trees. Time-variant data is used to extend VFDT to develop the CVFDT system. Flooding is used to keep trees up-to-date with time variant data streams[7]. Results also show that performance analysis of CVFDT is better than VFDT and Hoeffding tree in terms of accuracy. Also CVFDT handles Concept Drift which is not handled by other compared algorithms. Concept drift and Memory utilization are the crucial challenges in Stream data mining field.

## REFERENCES

[1] Elena ikonomovska,Suzana Loskovska,Dejan Gjorgjevik, “A Survey Of Stream Data Mining” Eight National Conference with International Participation-ETAI2007
[2] S.Muthukrishnan, “Data streams: Algorithms and Applications”.Proceeding of the fourteenth annual ACM-SIAM symposium on discrete algorithms,2003
[3] Mohamed Medhat Gaber, Arkady Zaslavsky and Shonali Krishnaswamy. ]“Mining Data Streams: A Review”,Centre for Distributed Systems and Software Engineering, Monash University900

Dandenong Rd, Caulfield East, VIC3145, Australia

[4] P. Domingos and G. Hulten, "A General Method for Scaling Up Machine Learning Algorithms and its Application to Clustering", Proceedings of the Eighteenth International Conference on Machine Learning, 2001, Williamstown, MA, Morgan Kaufmann

[5] H. Kargupta, R. Bhargava, K. Liu, M. Powers, P.Blair, S. Bushra, J. Dull, K. Sarkar, M. Klein, M. Vasa, and D. Handy, VEDAS: "A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring", Proceedings of SIAM International Conference on Data Mining, 2004.

[6] B. Babcock, M. Datar, and R. Motwani. "Load Shedding Techniques for Data Stream Systems" (short paper) In Proc. of the 2003 Workshop on Management and Processing of Data Streams, June 2003

[7] Tusharkumar Trambadiya, Praveen Bhanodia, "A Comparative study of Stream Data mining Algorithms" International Journal of Engineering and Innovative Technology (IJEIT) Volume 2, Issue 3, September 2012.