Classification of Big Data Stream usingEnsemble Classifier

Usha.B.P [1], Sushmitha.J[2], Dr Prashanth C M[3]

Abstract- Big Data Stream mining has some inherent challenges which are not present in traditional data mining. Not only Big Data Stream receives large volume of data continuously, but also it may have different types of features. Moreover, the concepts and features tend to evolve throughout the stream Recently, mining data streams with concept drifts for actionable insights has become an important and challenging task for a wide range of applications including credit card fraud protection, target marketing, network intrusion detection, etc. Conventional knowl-edge discovery tools are facing two challenges, the overwhelming volume of the streaming data, and the concept drifts. Traditional data mining techniques are not sufficient to address these challenges In our current work, we have designed a multi-tiered ensemble based method to label instances in an evolving Big Data Stream In this paper, we propose a general framework for mining concept-drifting data streams using ensemble classifiers. We train an ensemble of classification models, such as naive Bayesian,decision tree etc., from sequential chunks of the data stream. The classifiers in the ensemble are judiciously weighted based on their expected classification accuracy on the test data under the time-evolving environment. Thus, the ensemble approach improves both the efficiency in learning the model and the accuracy in performing classification. Our empirical study shows that the proposed methods have substantial advantage over single-classifier approaches in prediction accuracy, and the ensemble framework is effective for a variety of classifition models.

Index Terms— Evolving Big Data Stream, Concept Drift,Datapreprocessing,Ensembleclassifiers, Scalability.

———————————— ◆ ————————————

1 Introduction

Nowadays, the development of computers technology and its recent applications provide access to new types of data, which have not been considered by the traditional data analysts. Two particularly interesting characteristics of such data sets include their huge size and streaming nature. In many fields the quantity of data collected every day has exponentially exploded . It means that it is not possible to store all data which is produced and such a large scale algorithms exceed the processing capacity of computing systems. Moreover, standard data mining methods cannot handle its complex structure and size and fulfill computational cost-effective requirements as even simple computational operations are too costly. On the other hand, this also opens many challenging tasks for data mining community, like developing new types of algorithms, protocols and other computer infrastructure to mine large, massive and high dimensional data dealing with new types of data complexity developing new distributed and parallel computation tools, considering new kinds of appropriate privacy and security issues.

The above-mentioned requirements and challenges are particularly visible in emerging applications where data are continuously generated at a high rate in a form of data streams. Sensor networks, monitoring, traffic management, telecommunication, or web log analysis are just representatives of such applications where machines working in dynamic environments generate streaming data.

1.1 Need for classifying big data streams

An important problem in the streaming scenario is that of data classification. In this problem, the data instances are associated with labels, and it is desirable to determine the labels on the test instances [4]. Typically, it is assumed that both the training data and the test data may arrive in the form of a stream. In the most general case, the two types of instances are mixed with one another. Since the test instances are classified independently of one another, it is usually not difficult to perform the real-time classification of the test stream. The problems arise because of the fact that the training model is always based on the aggregate properties of multiple records. These aggregate properties may change over time. This phenomenon is referred to as concept drift [5]. All streaming models need to account for concept drift in the model construction process. Therefore, the construction of a training model in a streaming and evolving scenario can often be very challenging.

In the last few decades several new algorithms for learning classifiers from evolving data streams have been introduced [6]. However, several research issues still remain open. Having a robust and scalable approach to mine these Big Data Streams is becoming increasingly more important.

2 RELATED WORK

Data stream processing has recently become a very important re-search domain. Much work has been done on modeling [1], querying [2], and mining data streams, for instance, several papers have been published on classification, regression analysis , and clustering].

Traditional data mining algorithms are challenged by two characteristic features of data streams: the infinite data flow and the drifting concepts. As methods that require multiple scans of the datasets [3] cannot handle infinite data flows, several incremental algorithms that refine models by continuously incorporating new data from the stream have been proposed. In order to handle drifting concepts, these methods are revised again to achieve the goal that effects of old examples are eliminated at a certain rate. In terms of an incremental decision tree classifier, this means we have to discard, re-grow sub trees, or build alternative subtrees under a node [4]. The resulting algorithm is often complicated, which indicates substantial efforts are required to adapt state-of-the-art learning methods to the infinite, concept-drifting streaming environment. Aside from this undesirable aspect, incremental methods are also hindered by their prediction accuracy. Since old examples are discarded at a fixed rate (no matter if they represent the changed concept or not), the learned model is supported only by the current snapshot – a relatively small amount of data. This usually results in larger prediction variances.

Classifier ensembles are increasingly gaining acceptance in the data mining community. The popular approaches to creating ensembles include changing the instances used for training through techniques such as Bagging , Boosting], and pasting. The classifier ensembles have several advantages over single model classifiers. First, classifier ensembles offer a significant improvement in prediction accuracy [5]. Second, building a classifier ensemble is more efficient than building a single model, since most model construction algorithms have super-linear complexity. Third, the nature of classifier ensembles lend themselves to scalable parallelization and on-line classification of large database previously, we used averaging ensemble for scalable learning over very-large datasets. We show that a model's performance can be estimated before it is completely learned. In this work, we use weighted ensemble classifiers on concept-drifting data streams. It combines multiple classifiers weighted by their expected prediction accuracy on the current test data. Compared with incremental models trained by data in the most recent window, our approach combines talents of set of experts based on their credibility and adjusts much nicely to the underlying concept drifts. Also, we introduced the dynamic classification technique [6] to the concept-drifting streaming environment, and our results show that it enables us to dynamically select a subset of classifiers in the en-simple for prediction without loss in accuracy.

3 METHODOLOGY

Compared to static environments, data items arrive at high rate and algorithms that process them have to fulfill very strict constraints on memory usage, computing time, and on-line scan of incoming instances.

Another very important aspect of learning models from data streams refers to changes in the data distributions and target concepts over times which are usually called concept drifts. Detecting these changes and adapting classifiers to concept drift become one of the main challenges for streaming algorithms.

The current Hierarchical Stream Miner requires the separation of features into nominal and numeric. Different classifiers are used to train the model with different features. So the classifier becomes application specific, i.e. the algorithm has to know in prior the type of attributes. This limitation has to be eliminated.

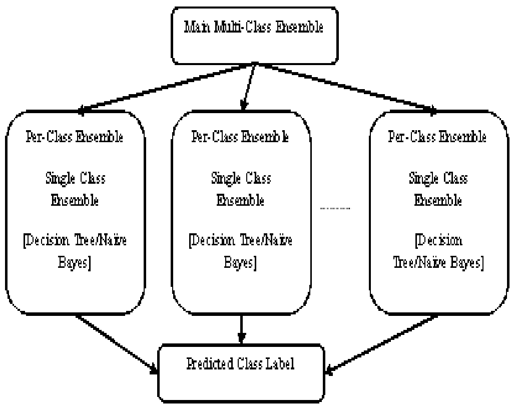


Fig 3.1 Multi –tier Ensemble

Ensemble classifier refers to a group of individual classifiers that are cooperatively trained on data set in a supervised classification problem. A supervised classification problem falls under the category of learning from instances where each instance/pattern/example is associated with a label/class. Conventionally an individual classifier like Neural Network, Decision Tree, or a Support Vector Machine is trained on a labeled data set. Depending on the distribution of the patterns, it is possible that not all the patterns are learned well by an individual classifier. Classifier performs poorly on the test set under such scenarios.

A solution to this problem is to train a group of classifiers (Fig.3.1) on the same problem. Individual classifiers are called base/weak classifiers. During learning, the base classifiers are trained separately on the data set. During prediction, the base classifiers provide a decision on a test pattern. A fusion method then combines the decisions produced by the base classifiers. There exists a good number of fusion methods in the literature including majority voting, Borda count, algebraic combiners etc.

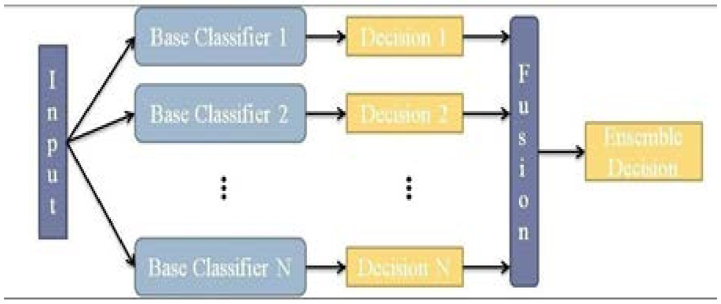


Fig 3.2 Ensemble Classifiers

The philosophy of the ensemble classifier is that another base classifier compensates the errors made by one base classifier. However, training the base classifier in a straightforward manner is not going to solve this problem. As pointed out in an ensemble classifier performs better than its base counterpart if the base classifiers are accurate and diverse.

The term diversity refers to the fact that the base classifier errors be uncorrelated. There are a good number of ways to compute diversity including Pairwise diversity measures (the Q statistics, the correlation coefficient, the disagreement measure, the double fault measure) and non-pairwise diversity measure (the entropy measure, Kohavi-Wolpert variance, measurement of interrater agreement).Different ensemble classifier generation methods aim to achieve diversity among the base classifiers. Some ensemble classifiers are also developed targeting specific problems/applications.

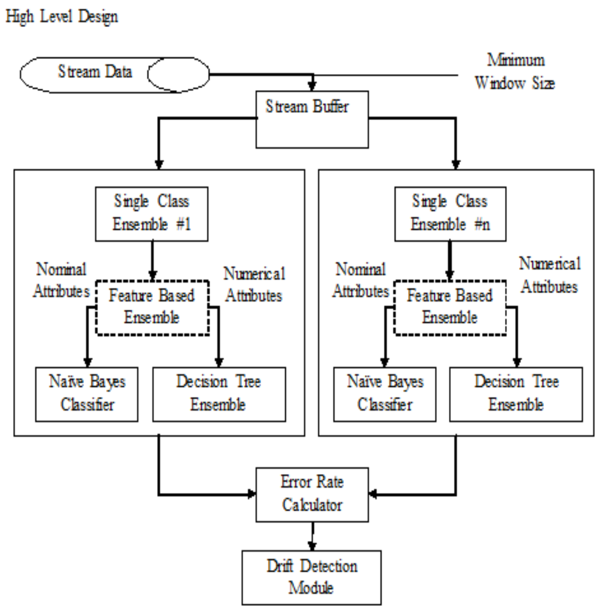


Fig 3.3 Sytem Architecture

- **Stream Data:** This is the stream of training samples used for training the classifier.
- **Stream Buffer:** This modules stores the stream data in the buffer the specified number of stream instances reach the specified minimum window size for training.
- **Single Class Ensemble:** This module contains many feature based ensemble classifiers. . The result of these feature based ensemble are collected and the final class label based on voting is generated.
- **Naïve Bayes Classifier:** This module implements the naïve bayes classification algorithm for nominal values.
- **Decision Stump Ensemble:** This module implements the decision stump ensemble classifier algorithm. For each numeric feature, a 1-level decision tree is created. The final prediction of the given instance is performed by voting the class labels predicted by all Single Class Classifiers.
- **Error Rate Calculator:** This module calculates the error rate of the classifier ensemble. .
- **Drift Detector:** This module measures the error rate and take appropriate action based on following conditions.
- If the error rate is increased beyond **(Average + (2 x Standard Deviation))** then a warning is raised and instances are stored in anticipation of a possible change of context.
- If the error rate is increased beyond **(Average + (3 x Standard Deviation))** then the drift is confirmed. The model is reset and a new model is build using the stored instances

3.1 Pseudo Code:

Pseudo Code #1: Single Class Ensemble Classifier

```
Input: Training Instance Stream
Output: Trained Classifier
Step 1:- Read instance from stream buffer

Instances = ReadInstancesFromBuffer( )

Step 2:- Find the decision rule using features
Step 3:-Initialize it to maximum misclassification error

BestSplitMerit = 1
BestSplitFeature = NONE
```

```
Step 4:- Split the records at value V
    PreSplitDistributionfactor = Instances
    PostSplitDistributionfactor = Split(Instances)

Step 5:- Calculate the Split Merit Score using Information Gain
    SplitMeritscore = FindBestSplit(PreSplitDistributionfactor, PostSplitDistributionfactor)

Step 6:-Update the Best Split Score
    IF SplitMerit < BestSplitMerit
    THEN
        BestSplitMerit = SplitMerit
        BestSplitFeature = feature
    END IF
Step 6: Generate Decision Tree after finding the best fit
Step 7:-If the node tree height has reached the limit than add a Naïve Bayes Classifier as leaf
Step 8:- node to at the bottom of the tree to classify the instances
```

Pseudo Code #2: Error rate calculator

```
Input: Classifier Prediction value, MinimumInstancesThresholdrate
Ouput: Error Rate Value

Step 1:-Calculate Miss Classification Error for each Class Label

MissClassificationError[] = (Predictionrate – Actualvalue)/TotalsetofInstances)

Step 2:-Calcualte Mean of Miss Classification Error
Mean = Mean / TotalClassLabelCountvalue

Step 3:-Calculate accuracy for each classification
Accuracy value=Number of correctly classified test instances/Total number of training instances
Error rate value=1-Accuracy value
```

Pseudo Code #3: Drift Detection

```
Input: Classifier Prediction value, MinimumInstancesThresholdrate, ErrorThresholdevalu
Ouput: Warning statement and Change Detection ranges

Step 1:-calculate Miss Classification Error for each Class Label

MissClassificationErrorvalue[] = (Predictionrate – Actualevalue)/TotalsetofInstances)

Step 2:- Calcualte Mean of Miss Classification Error
FOR all Class Labels
START
  Mean value = MissClassificationErrorvalue[ClassLabel]
END

Step 3:- Calculate Root Mean Squared Error
RMSE = Sqrt(Sum(Squre(Mean – MissClassificationError value[ ])))

Step 4:-Warning Detection
IF RMSE > errorThreshold
THEN
  Warning = True
  Change = False

Step 5:-Change Detection

Step 6:- Reset Warning and Change flags if the drift is identified
```

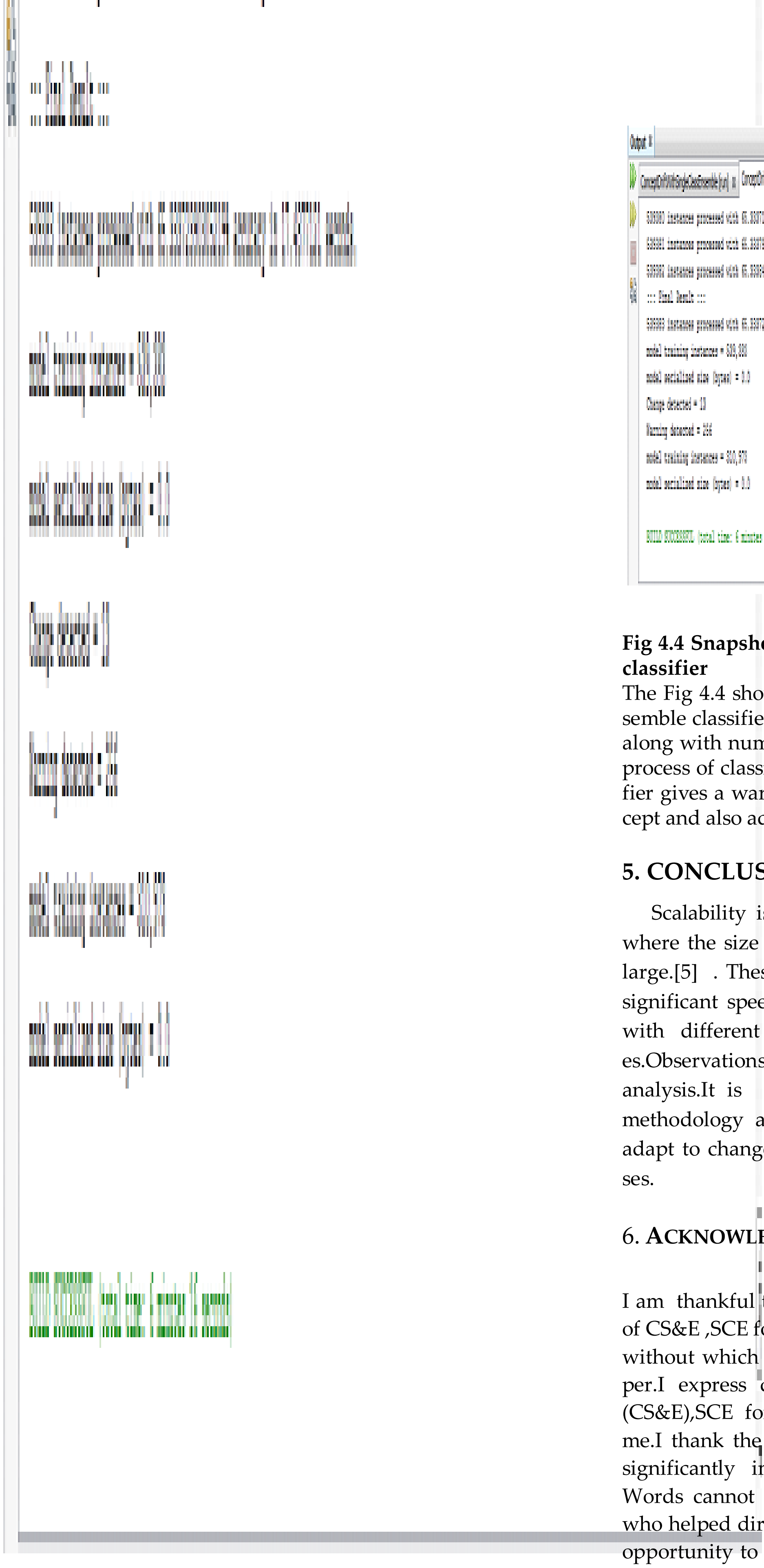
4. RESULTS

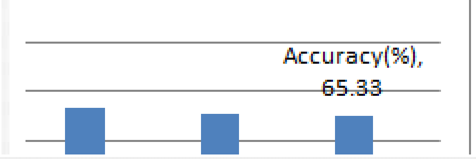
The table 4.1 shows different instances sampled in a particular set of time intervals and the rate of accuracy achieved respectively. The Airline dataset is used for sampling of instances, the classifier labels the data and classifies the data record.

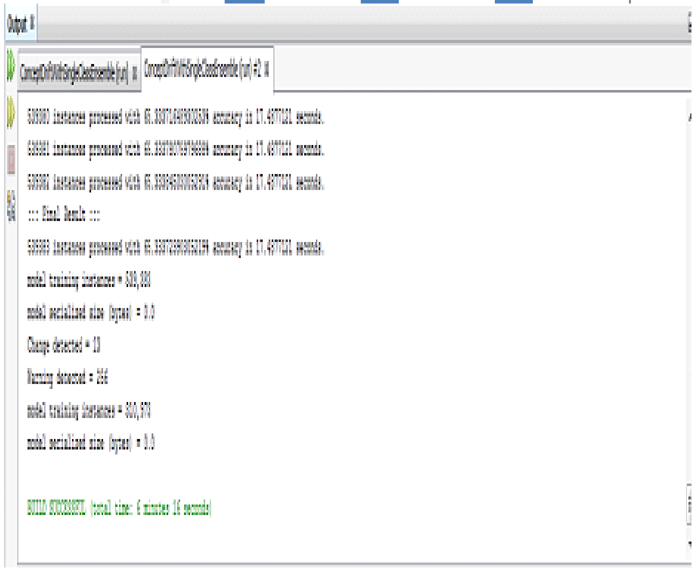
No of Instances	**Accuracy(%)**	**Processing Time(seconds)**
1,00,000	**68.74**	**3.47**
2,00,000	**68.98**	**6.52**
3,00,000	**67.56**	**9.78**
4,00,000	**66.66**	**13.10**
5,00,000	**65.49**	**16.20**
5,39,383	**65.33**	**17.48**

Table 4.1 Different test instances showing corresponding accuracy and processing time intervals.
Note:-The units of the values are accuracy in percentage and processing time in seconds

The following Fig 4.2 shows the graph derived by sampling the airline data instances. As the number of instances increases the processing time increases correspondingly. It demonstrates the way of prediction done about delay in







```
530380 instances processed with 65.3387[illegible] accuracy in 17.497121 seconds.
530381 instances processed with 65.3387[illegible] accuracy in 17.497121 seconds.
530382 instances processed with 65.3384[illegible] accuracy in 17.497121 seconds.
::: Final Results :::
530383 instances processed with 65.3387[illegible] accuracy in 17.497121 seconds.
model training instances = 520,380
model serialized size (bytes) = 0.0
Change detected = 13
Warning detected = 256
model training instances = 510,573
model serialized size (bytes) = 0.0

BUILD SUCCESSFUL (total time: 6 minutes 16 seconds)
```

Fig 4.4 Snapshot of detecting concept drift with ensemble classifier

The Fig 4.4 shows the sampled training instances to an ensemble classifier with the accuracy rate and processing time along with number of changes and warning detected in the process of classifying data streams. It shows how the classifier gives a warning message if there is any change in concept and also adapts to the samples of updated classifier.

5. CONCLUSION

Scalability issue can arise in case of Big Data Stream where the size of data chunk or the number of features is large.[5] . These algorithms help base method to achieve significant speedup and scaleup as shown. It is analyzed with different aspects of design for those approaches.Observations with the empirical data also supports that analysis.It is intend to continue to improve upon our methodology and add additional components to further adapt to changes in the data stream, including novel classes.

6. ACKNOWLEDGMENT

I am thankful to Mrs.Sushmitha.J Assistant Professor,Dept of CS&E ,SCE for her valuable advice and support extended without which I could not have been able to complete paper.I express deep thanks to 'Dr.Prashanth CM' HOD (CS&E),SCE for warm hospitality and affection towards me.I thank the anonymous referees for their reviews that significantly improved the presentation of this paper Words cannot express our gratitude for all those people who helped directly or indirectly in my endeavor.I take the opportunity to express my sincere thanks to all staff members of dept CS&E of SCE for the valuable suggestion.

REFERENCES

[1] A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, and R. Gavald`a, "New ensemble methods for evolving data streams," in KDD '09: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2009, pp. 139–148.

[2] P. H. dos Santos Teixeira and R. L. Milidi´u, "Data stream anomaly detection through principal subspace tracking," in Proceedings of the 2010 ACM Symposium on Applied Computing, ser. SAC '10. New York, NY, USA: ACM, 2010, pp. 1609–1616.

[3] I. Katakis, G. Tsoumakas, and I. Vlahavas, "Dynamic Feature Space and Incremental Feature Selection for the Classification of Textual Data Streams," in ECML/PKDD-2006 International Workshop on Knowledge Discovery from Data Streams. 2006, 2006.

[4] T. M. Al-Khateeb, M. M. Masud, L. Khan, and B. Thuraisingham, "Cloud guided stream classification using class-based ensemble," in Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing, ser. CLOUD '12. Washington, DC, USA: IEEE Computer Society, 2012, pp. 694–701.

[5] B. Parker, A. M. Mustafa, and L. Khan, "Novel class detection and feature via a tiered ensemble approach for stream mining," in ICTAI, 2012, pp. 1171–1178.

[6] A. Haque, B. Parker, and L. Khan, "Labeling instances in evolving data streams with mapreduce," in 2013 IEEE International Congress on Big Data (BigData Congress), 2013, pp. 387–394.