# Learning on High Frequency Stock Market Data Using Misclassified Instances in Ensemble

Meenakshi A.Thalor
Research Scholar,Computer Engineering
Vishwakarma Institute of Technology
Savitribai Phule Pune University
Pune, India

Dr. S.T.Patil
Professor, Computer Engineering
Vishwakarma Institute of Technology
Savitribai Phule Pune University
Pune, India

***Abstract*—Learning on non-stationary distribution has been shown to be a very challenging problem in machine learning and data mining, because the joint probability distribution between the data and classes changes over time. Many real time problems suffer concept drift as they changes with time. For example, in stock market, the customer's behavior may change depending on the season of the year and on the inflation. Concept drift can occurs in the stock market for a number of reasons for example, trader's preference for stocks change over time, increases in a stock's value may be followed by decreases. The objective of this paper is to develop an ensemble based classification algorithm for non-stationary data stream which would consider misclassified instances during learning process. In addition, we are presenting here an exhaustive comparison of proposed algorithms with state-of-the-art classification approaches using different evaluation measures like recall, f-measure and g-mean.**

***Keywords—Classifiers; Concept drift; Data stream; Ensemble; Non-stationary Environment***

## I. INTRODUCTION

Nowadays most of the applications are online applications, where huge amount of data increasingly arrives at every time stamp which is generated from different sources. So it is very important to train classifiers incrementally over the time so that they can learn different concepts of non-stationary data streams.

Conventional data mining algorithms assumes that each dataset is produced from a single, static and hidden function i.e. the function (model/classifier) generating data at training time is the same as that of testing time. Whereas in non-stationary data stream, data is continuously coming and the function which is generating instances at time *t* need not be the same function at time *t+1*. This difference in the underlying function is called as concept drift [1].

The concept drift problem is studied in literature with different terminology as "concept shift" , "concept drift", dataset shift ,"change of classification", "changing environments", "non-stationary environment" etc. Concept drift in data stream happens when the relationship between the input and class variables changes over time and this can happen because of change in the following:

*1) The class priors, $P(c_i)$, $i = 1, 2, 3, \ldots k$, where k is the number of classes;*

*2) The distribution of the classes, $P(X| c_i)$, where $i = 1, 2, 3, \ldots k$ and X is a vector of labeled instances; and*

*3) The posterior distribution of the class membership $P(c_i |X)$, $i = 1, 2, 3, \ldots k$*

For providing training to classifiers incrementally over the time so that they can learn different concepts of non-stationary data streams we are using ensemble based approach [2] as shown in fig 1.

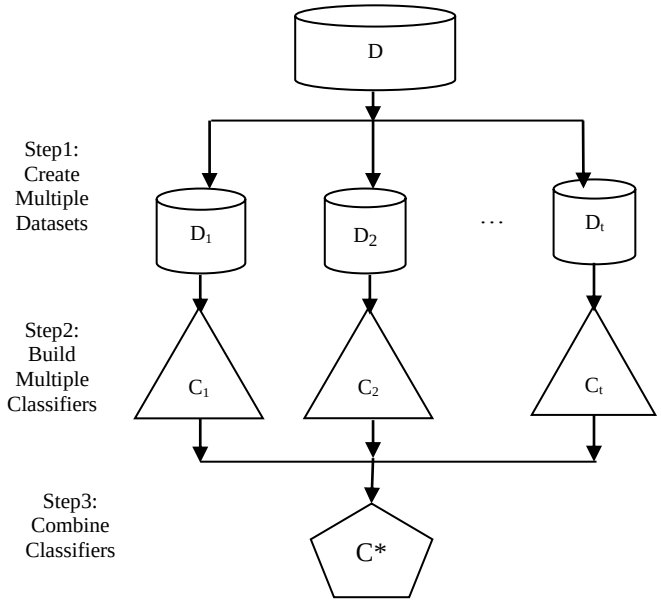


Fig. 1. Ensemble Based Learning

In data mining, the ensemble is a pool of classifiers whose individual classifications/predictions are combined in some way to classify unseen examples. The strategy in ensemble systems [3] is to create subsets of incoming data stream and for each subset a classifier is trained and tested and then these classifiers collectively would do decision making and predict the label for unseen data

Performance of learning algorithms dependent upon the size of the data chunks (block/batch). Bigger blocks [4] can results in accurate classifiers as classifiers are getting more data for training, but can contain too many different concept drifts. Whereas smaller blocks are better for drifted data stream, but usually lead to poorer classifiers as training data is less.

In this paper, state of the art Learn$^{++}$.NSE algorithm is evaluated for handling non-stationary data with our proposed approach. This paper is organized as follows: Section 2 offers an overview of related work. Section 3 presents proposed algorithm i.e. ENSDS_P and Section 4 provides detail of proposed algorithm with its pseudo code. Section 5 provides a rigorous evaluation of the proposed algorithm with one of the existing algorithms. Section 6 concludes the paper.

## II. Related Work

The first experiment of ensembles in data streams was the one proposed by Street and Kim with their Streaming Ensemble Algorithm [5] (SEA) where a chunk of d instances is read from the data stream and used to build a classifier. As fixed size of ensemble was used, so they compare new generated classifier against a pool of previously trained classifiers (from previous chunk), and if its current classifier improves the quality of ensemble it is included at the cost of the worst classifier. SEA uses a simple majority vote and may not be able to perform in recurring environments.

Wang et al. proposed Accuracy Weighted Ensemble [6] (AWE) of classifiers on each incoming data chunk and use that chunk to evaluate the performance of all existing classifiers in the ensemble. The weight of each classifier is the difference of error rate of a random classifier and the mean square error of the classifier for the current chunk. The mean square errors of old classifiers are high, and thus the weights of old classifiers are small.

Brzezinski and Stefanowski proposed the Accuracy Updated Ensemble [7] (AUE) which is derived from AWE.It uses the same principles of chunk-based ensembles, but with incremental base components/classifiers. It not only builds new classifiers, but also conditionally updates existing classifiers on new chunks rather than just adjusting their weights. The updation of existing base classifiers makes AUE better than AWE in case of gradual drift but conditionally updating of base classifiers is less accurate for sudden drift.

Robi Polikar et al. proposed Learn$^{++}$.NSE [8], [9], [10], [11], [12] (Nonstationary Environment) which generates classifiers sequentially using batches of examples/instances (Not true online learner as it converts the online data stream into a series of chunks of a fixed size). At each time step, one new classifier is trained on recent distribution, using an instance weighting distribution. In Learn$^{++}$.NSE each classifier's weight is computed using a weighted average of its prediction error on old and current batch and finally uses weighted majority voting to obtain ensemble's output.

Most recently, Brzezinski and Stefanowski proposed AUE2 [13] introduces a new weighting function, does not require cross-validation on the existing classifiers, does not keep a classifier buffer, prunes its base learners, and always unconditionally updates its components. Classifiers are updated after every chunk, so they can react to gradual drifts. It can react to sudden drifts and gradual drifts but not for reoccurring concepts. Compared to Learn$^{++}$.NSE, AUE2 incrementally trains existing component classifiers, retains only k of all the created components, and uses a different weighting mechanism which ensures that components will have non-zero weights.

## III. Proposed Algorithm

Fig. 2 depicts the flow diagram of ensemble for non-stationary data stream with propagation (ENSDS_P). ENSDS_P is our proposed algorithm, which is an ensemble of classifiers, where the classifiers are generated from data arrived at time t and evaluated on recent data. All generated classifiers are combined by using weighted majority voting to provide the predictions of unseen data.

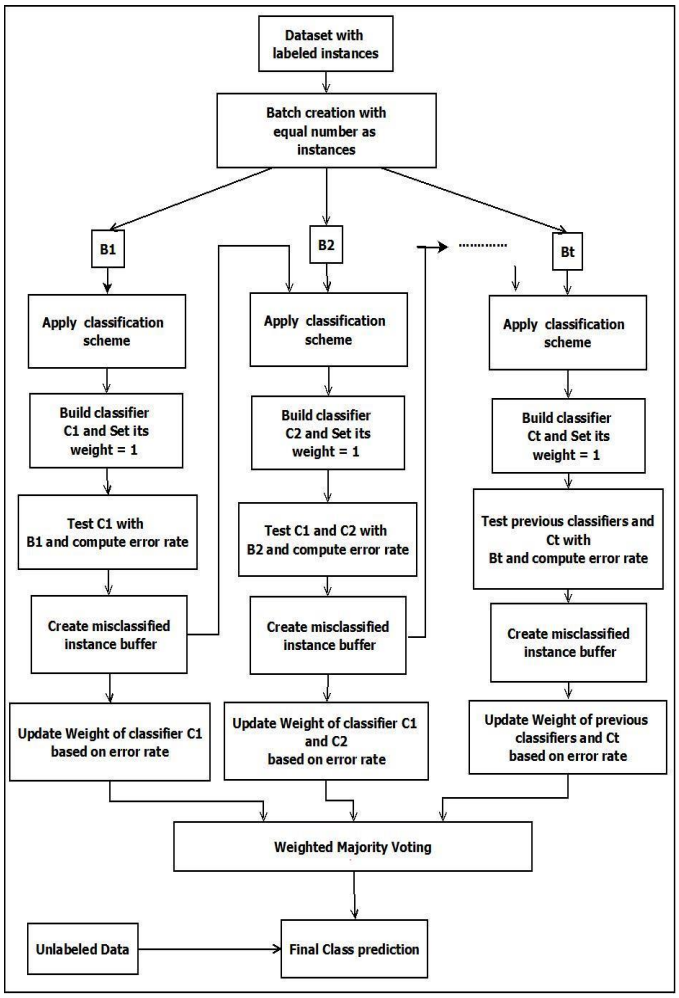


Fig. 2. Flow Diagram of ENSDS_P

One of the major differences in ENSDS_P as compared to existing approaches is, we are not updating a set of weights for each instance rather we believe all instances are equally important while they are using in training so uniform weight is considered and secondly we are propagating the misclassified instances of a classifier to subsequent classifier for improving the performance.

In this system, data is continuously arriving in non-stationary manner. For learning purpose, we take dataset containing labeled instances. Divide this incoming data into number of batches where each batch contains equal number of instances. First apply any suitable classification scheme to

create a classifier. The performance of classifier is then evaluated with same batch of instances. If the error rate of classifier is more than 50% i.e. half of the predictions are wrong then delete that recently generated classifier and again repeat the classification process till we get a classifier having an error rate less than 50%.

After creation of first classifier, a misclassified instance buffer is used to store hard to classify instances. All hard to classify instances are propagated to next classifier so that next subsequent classifier can learn them with their training chunk and overall system performance can be improved.

When the next batch of data get available, the incorrect classified instances of previous classifier would be combined with labeled instances of current batch and then apply classification scheme. From this step we get next classifier. This process is continued till we get classifier for all batches and all these classifiers are combined using weighted majority voting scheme. When unlabeled data is arrived, it is predicted by created ensemble using weighted majority voting.

Two variations of ENSDS_P are developed and analyzed, first approach named as ENSDS_P_F where we are propagating misclassified instances, but preserving fix batch size for all classifiers while other approach named as ENSDS_P_D where we are propagating misclassified instances and dynamic chunk size is used for training of classifiers.

## IV. Algorithmic Description

Fig. 3 presents the pseudo code of proposed algorithm. For each t, a new classifier generates on current training chunk $D^t$, and the performance of all previously generated classifiers would be evaluated on current data chunk by $\varepsilon_k^t$ parameter and misclassified instances would be saved in a buffer $M_k^t$ .The misclassified instances will be propagated to next subsequent classifier with their training chunk.

In step 1, a uniform weight is assigned to all instances of current data chunk .Step 2 is only different in ENSDS_P_D and ENSDS_P_F rest of algorithm will remain same for both. ENSDS_P_D achieves $D^t$ by eq. 1 where total no. of instances in current data chunk would be union of $D^t$ and misclassified instances of previous data chunk.

$$D^t = D^t \cup M_k^{t-1} \quad (1)$$

ENSDS_P_F achieves $D^t$ by eq. 2 and 3 where $ND^t$ represent new dataset whose size equal to size of current data chunk minus size of misclassified instance buffer.

$$Size(ND^t) = Size(D^t) - Size(M_k^{t-1}) \quad (2)$$

$$D^t = ND^t \cup M_k^{t-1} \quad (3)$$

After formation of $k^{th}$ classifier as in step 3, the performance of existing classifiers will be evaluated over the current training dataset $D^t$ and we will get $\varepsilon_k^t$ which is error of $k^{th}$ classifier on current $D^t$. If error generated by current classifier is more than .5 that is half of the predictions are wrong then generate a new classifier for the current distribution. If error generated by one of theprevious classifier is more than .5 then set its $\varepsilon_k^t = 0.5$ as in step 4.We are not normalizing the $\varepsilon_k^t$ as its value remains between 0 to 0.5 and voting power of a classifier having $\varepsilon_k^t = .5$ will remain low.

*Algorithm:* ENSDS_P
*Input:* For each dataset $D^t$ where t=1, 2, ….
*Training Data*: $\{X_i^t \in X;\ y_i^t \in Y = \{1, \ldots, c\}\}$, i=1,…, m instances
*Description:* Supervised learning algorithm to handle non-stationary data stream
Do for t = 1, 2, ...

1. Initialize $D^t(i) = 1/m, \forall i,$
2. If t = 1 then
   Goto step 3
   Else
   Refer Eq. 1 for ENSDS_P_D
   or
   Refer Eq. 2,3 for ENSDS_P_F
3. Call base classifier with $D^t$, obtain $h_k: X \rightarrow Y$ where k = 1, 2, ...t
4. Evaluate all existing classifiers $(h_k)$ on $D^t$

$$\varepsilon_k^t = \sum_{i=1}^{m} D^t(i).\ [|h_k(X_i) \neq y_i|]$$

   If $\varepsilon_{k=t}^t > \frac{1}{2}$ generate a new $h_k$
   If $\varepsilon_{k<t}^t > \frac{1}{2}$ Set $\varepsilon_k^t = \frac{1}{2}$
5. $M_k^t$ = ∀i where $[|h_k(x_i) \neq y_i|]$ is true
6. Compute the weight for kth classifier$h_k$

$$Sig_k^t = \frac{1}{1 + e^{-a(t-k-b)}}$$

$$w_k^t = \begin{cases} 1 & t = k \\ \dfrac{Sig_k^t}{Sig_k^t + \sum_{j=1}^{t-1} w_k^{t-j}}, & \text{otherwise} \end{cases}$$

7. Calculate classifier voting weights
   Voting $w_k^t = \ln\left(\frac{1}{\sum_{j=1}^{t} w_k^j \varepsilon_k^j}\right)$ for k = 1, …, t
8. Obtain the final hypothesis

$$H^t(X_i) = arg \max_c \sum_k w_{k*}^t\ [|h_k(X_i) = c|]$$

Fig. 3. The Pseudo code of the algorithm ENSDS_P

In step 5, we are creating $M_k^t$ parameter which represents a buffer to hold misclassified instances. These misclassified instances would be propagated to next classifier before its formation with its training chunk. A nonlinear sigmoid function is used to set weight of a classifier. Because of this, if a classifier will be evaluated more than once then its sigmoid weight will get increased.

The weight to a classifier is assigned based on its performance on previous distributions as well as on recent distribution so weighted average of classifier is computed in step 6. When a classifier is generated it's $w_k^t = 1$, after its evaluation on recent environment its $w_k^t$ gets keep updated. If a classifier does not performs well on recent environment, then its weightederror $(w_k^t . \varepsilon_k^t)$ will gets increased. In step 7 the weight error average is computed to determine the voting

weight of classifiers. The voting power of each classifier is computed using logarithm of the inverse of its weighted error average. If weighted error average is high a classifier will get less power of voting.

The time complexity of ENSDS_P is O(t*k*O(x*m)+k*t*m) where O(x*m) is the time complexity of Naïve Bayes classifier, x is number of features and m is number of instances in training set, k indicates number of classifiers, t indicates number of data chunks to be predicted.

## V. Comparative Evaluation and Analysis

In the following subsections; we describe the tested datasets, experimental setup, and comparative analysis of experimental results.

### A. Datasets

For doing the comparison of ENSDS_P and existing algorithm (Learn$^{++}$.NSE) we are using different datasets with different batch sizes. The proposed algorithm is tested over real time datasets.

*1) IBM_EOD_Direction:* The IBM_EOD_Direction dataset contains stock data of IBM Company, where we are considering open, high, low, close, volume and rate of change in closing price to find out the stock index movement (Up, Down) for classification task. For training purpose data from period 2-Jan-2000 to 13-April-2016 (3999 examples) is fetched and for testing purpose data from period 2-Jan-2001 to 13-April-2016 (3841 examples) is fetched using Google finance.

*2) IBM_EOD_Trading:* The IBM_EOD_ Trading dataset contains stock data of IBM Company, where we are considering open, high, low, close, volume and rate of change in closing price to find out Buy or Sell class for Stock data. For training purpose data from period 2-Jan-2000 to 13-April-2016 (3999 examples) is fetched and for testing purpose data from period 2-Jan-2001 to 13-April-2016 (3841 examples) is fetched using Google finance.

The purpose of considering stock data is as we know that stock market data is high frequency data which is complex, non-stationary, chaotic and non-linear and suites our research topic .Concept drift can occurs in the stock market for a number of reasons for example, traders preference for stocks change over time, increases in a stock's value may be followed by decreases. Stock market data can possess sudden, gradual and recurring drift at any moment of time.

The analysis over IBM_EOD_Direction dataset would help trader to know the position of stock market index at next moment of time and analysis over IBM_EOD_Trading would help trader to take decision whether its right time to sell or purchase the stock.

### B. Experimental Setup

For experiment analysis, the proposed algorithm is implemented in Java using MOA and WEKA framework. The source code of Learn$^{++}$.NSE is obtained from MOA extensions for comparison purpose. The experiments were conducted on a machine equipped with Processor Intel(R) Core(TM) i3-2120 CPU @ 3.30GHz, 2 Core(s), 4 Logical Processor(s) and 4 GB of RAM. Here we have used different batch size for comparison purpose. However, the optimal batch size is different for each stream. For rigorous evaluation, we are considering different evaluation measures [14] like P=precision, R=recall, A=accuracy, F-M=f-measure, and G-M=g-mean.

### C. Results

Table 1 depicts the performance of Learn$^{++}$.NSE and both the versions of ENSDS_P respectively to classify the stock index movement(Up, Down) over IBM_EOD dataset where we are considering Naïve Bayes as base classifiers, different batch size and no pruning strategy is used.

TABLE I. Comparison Over IBM_EOD_Direction Dataset

| Batch Size | Algorithms | P | R | A | F-M | G-M |
|---|---|---|---|---|---|---|
| 500 | Learn++.NSE | 58.77 | 94.13 | 91.07 | 72.36 | 92.37 |
| | ENSDS_P_D | 87.83 | 84.94 | 94.48 | 86.36 | 90.75 |
| | ENSDS_P_F | 88.61 | 76.84 | 92.42 | 82.31 | 86.36 |
| 400 | Learn++.NSE | 57.33 | 98.21 | 91.30 | 72.40 | 94.22 |
| | ENSDS_P_D | 79.32 | 94.98 | 95.05 | 86.45 | 95.03 |
| | ENSDS_P_F | 82.20 | 86.86 | 93.99 | 84.47 | 91.14 |
| 300 | Learn++.NSE | 29.84 | 87.02 | 85.16 | 44.44 | 86.02 |
| | ENSDS_P_D | 88.61 | 56.32 | 84.07 | 68.87 | 73.80 |
| | ENSDS_P_F | 91.23 | 63.71 | 87.92 | 75.03 | 78.84 |

It is clear from fig. 4 that for each batch size we are retrieving high true positives and low false positives hence precision is higher. The results shows precision of both the versions of ENSDS_P is significantly high and recall is approximately equal.

Generally, there always remains a tradeoff between precision and recall. F-measure is appropriate evaluation measure which gives the balance between precision and recall. As compare to Learn$^{++}$.NSE we are able to maintain a good balance between precision and recall so proposed algorithm can also be used with imbalanced data. The values of evaluation measures proved the validity of proposed work hence evaluation results shows that proposed algorithms effectively provides incremental learning over high frequency stock market data.

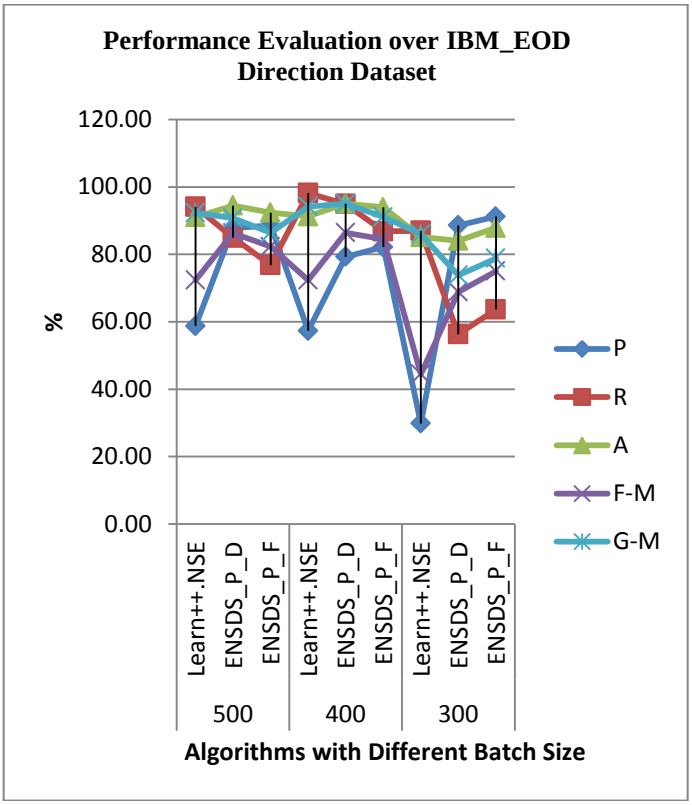


Fig. 4. Performance Analysis of Learn++.NSE and ENSDS_P over IBM_EOD_Direction dataset

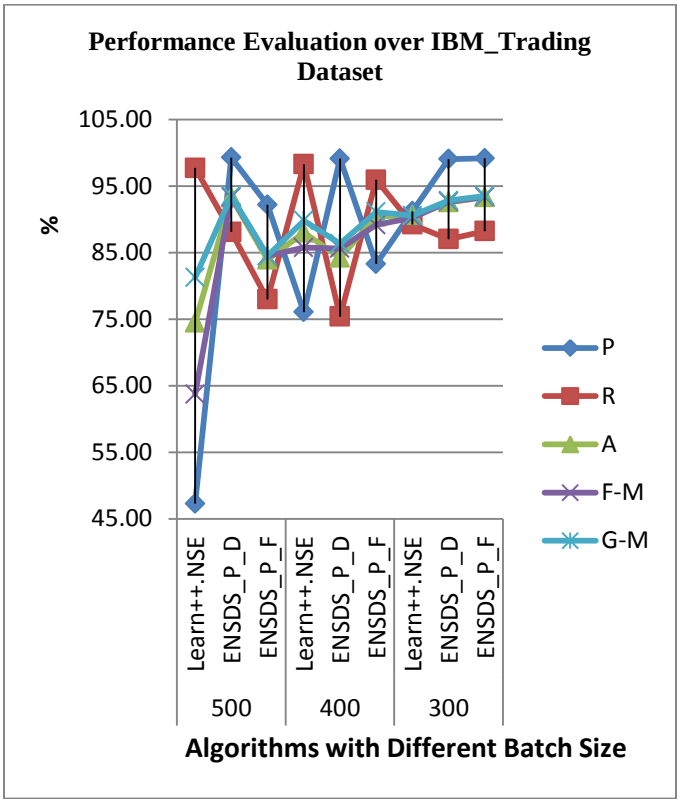


Fig. 5. Performance Analysis of Learn$^{++}$.NSE and ENSDS_P over IBM_EOD_Trading Stock dataset

Table 2 depicts the performance of Learn$^{++}$.NSE and versions of ENSDS_P respectively to classify the stock trading (Buy, Sell) over IBM_EOD_Trading dataset where we are considering Naïve Bayes as base classifiers, different batch size and no pruning strategy is used.

TABLE II. COMPARISON OVER IBM_EOD_TRADING DATASET

| Batch Size | Algorithms | P | R | A | F-M | G-M |
|---|---|---|---|---|---|---|
| 500 | Learn++.NSE | 47.28 | 97.73 | 74.49 | 63.73 | 81.26 |
| | ENSDS_P_D | 99.29 | 88.11 | 93.31 | 93.36 | 93.52 |
| | ENSDS_P_F | 92.20 | 77.98 | 83.96 | 84.50 | 84.51 |
| 400 | Learn++.NSE | 76.06 | 98.30 | 88.02 | 85.76 | 89.82 |
| | ENSDS_P_D | 99.12 | 75.40 | 84.25 | 85.65 | 86.35 |
| | ENSDS_P_F | 83.31 | 95.95 | 90.42 | 89.18 | 91.13 |
| 300 | Learn++.NSE | 91.21 | 89.25 | 90.63 | 90.22 | 90.58 |
| | ENSDS_P_D | 99.07 | 87.02 | 92.55 | 92.66 | 92.84 |
| | ENSDS_P_F | 99.18 | 88.23 | 93.34 | 93.38 | 93.54 |

Fig. 5 represents that as compared to Learn$^{++}$.NSE we have achieved high precision, accuracy, f-measure and g-mean for all batches on IBM_EOD_Trading dataset.

After testing proposed algorithm on different datasets, evaluation measures confirm the validity and excellence of proposed algorithm. The non-stationary data can have class imbalanced problem so result can be biased toward the majority class; thus the classifier tends to misclassify the minority class instances. In imbalanced application area, proposed algorithm can be used and can provide a balance between majority and minority instances.

## VI. CONCLUSION

From the implementation and analysis of ENSDS_P we can conclude that the performance of ENSDS_P is better as compared to Learn$^{++}$.NSE on different datasets. Evaluation measures also confirm the validity of proposed algorithm's scores. The selection of optimal batch size varies from dataset to datasets. The non-stationary data can have class imbalanced problem so result can be biased toward the majority class; thus the classifier tends to misclassify the minority class instances. If dataset is highly imbalanced then there is need to add some balancing mechanism in proposed algorithm to achieve high performance.

## REFERENCES

[1] Moreno-Torres, J., Raeder, T., Alaiz-Rodríguez, R., Chawla, N.V.,Herrera, F., "A unifying view on dataset shift in classification", Pattern Recogniion, 45, 521–530 , 2011.

[2] R. Polikar , "Ensemble based systems in decision making", IEEE Circuits and Systems Magazine, Vol. 6, No. 3 ,pp. 21-45, 2006

[3] Meenakshi A.Thalor ,Dr.S.T.Patil, "Review of ensemble based classification algorithms for nonstationary and imbalanced data" ,IOSR Journal of Computer Engineering,e-ISSN: 2278-0661, Vol. 16, pp. 103-107, Feb 2014.

[4] Read, J., Bifet, A., Pfahringer, B. & Holmes, G. "Batch-incremental versus instance-incremental learning in dynamic and evolving data ", IDA 2012, pp. 313-323, Helsinki, Finland, October 25-27 2012

[5] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," Intellegent Conference on Knowledge Discovery & Data Mining, pp. 377-382, 2001.

[6] H. Wang, W. Fan, P. Yu, and J. Han, "Mining concept-drifting data streams using ensemble classifiers," in Proc. ACM SIGKDD Int. Conf. Knowl. Disc. Data Min., pp. 226–235, 2003.

[7] Dariusz Brzezinski and Jerzy Stefanowski , "Accuracy updated ensemble for data streams with concept drift,"Proceedings of the 6th international conference on Hybrid artificial intelligent systems - Volume Part II,2011,pp. 155-163.

[8] Elwell R. and Polikar R., "Incremental learning of concept drift in non-stationary environments, "IEEE Trans. on Neural Networks, vol. 22, 2011,pp. 1517-1531.

[9] R. Elwell and R. Polikar, "Incremental learning of variable rate concept drift, "International Workshop on Multiple Classifier Systems (MCS 2009) in Lecture Notes in Computer Science, vol. 5519, pp. 142-151, 2009.

[10] M. Karnick, M. Ahiskali, M. Muhlbaier, and R. Polikar, "Learning concept drift in nonstationary environments using an ensemble of classifiers based approach, "International Joint Conerence.on Neural Network,2008, pp. 3455-3462.

[11] M. Muhlbaier and R. Polikar, "An ensemble approach for incremental learning in nonstationary environments, "Multiple Classifier Systems, pp. 490-500, 2007.

[12] Michael D. Muhlbaier and RobiPolikar, "Multiple classifiers based incremental learning algorithm for learning in nonstationary environments", Proceedings of the Sixth International Conference on Machine Learning and Cybernetics, vol. 6, 2007,pp. 3618–3623.

[13] Brzezinski, D.; Stefanowski, J., "Reacting to different types of concept drift: the accuracy updated ensemble algorithm, " , IEEE Transactions on Neural Networks and Learning Systems Vol. 25(1),2014, 81-94.

[14] Jesse Davis, Mark Goadrich , "The Relationship Between Precision-Recall and ROC Curves," In Proceedings of the 23rd international conference on Machine learning , pp. 233-240,2006.