RESEARCH ARTICLE OPEN ACCESS

Efficient Analsysis on Map Reduce-Based Ensemble Learning Technique with Several Classifier Methods and Impact of Diversity for Condition-Based Maintenance with Concept Drifts

Alampally Sreedevi[1], Kavitha Gopu[2]
[1,2]Assistent Professor, Dept of CSE, Sri Indu College of Engineering and Technology, Hyderabad,Telangana, India

Abstract:

Condition-based maintenance (CBM) in Industry 4.0 gathers an enormous measure of generation information stream persistently from IoT gadgets appended to machines to conjecture the time when to keep up machines or supplant parts. Be that as it may, as conditions of machines change progressively with time inferable from machine maturing, glitch or substitution, the concept of catching the guaging design from the information stream could float erratically so it is elusive a strong determining technique with high accuracy. Subsequently, this work proposes a group learning strategy with different classifier composes and assorted variety for CBM in assembling ventures, to address the predisposition issue when utilizing just a single base classifier write. Beside controlling information assorted variety, this strategy incorporates numerous classifier writes, dynamic weight changing, and information based adaption to concept drifts for disconnected learning models, to advance accuracy of the determining model and exactly identify and adjust to concept drifts. With these highlights, the proposed technique requires capable registering assets to viably react to down to earth CBM applications. Consequently, moreover, the execution of this strategy based on the MapReduce framework is proposed to increment computational productivity. Recreation comes about demonstrate that this strategy can recognize and adjust to all concept drifts with a high accuracy rate.

***Index Terms*—Concept drift, ensemble learning, mapReduce, condition-based maintenance, Industry 4.0**

I. INTRODUCTION

Data classification has assumed an essential part in huge data examination and has an assortment of utilizations, e.g., spam sifting, medicinal analysis, and fault detection and classification (FDC) in semiconductor fabricating. A current pattern in data classification centers around how to identify and adjust to the issue of concept drifts. This work centers around condition-based maintenance (CBM) with concept drifts in Industry 4.0, in which the tremendous verifiable data stream of conditions of machines is gathered persistently from sensors or IoT gadgets appended to machines [1], and data classification calculations are adjusted to examine the data to identify the time when a few segments in machines begin to perform unusually and ought to be supplanted ahead of time, to evade machines from slamming or assembling an immense measure of flawed items [2]. Henceforth, chronicled data is the principle factor to influence the exactness performance of data classification calculations. In any case, as time passes by, machine parts end up maturing, failed, or should be kept up, with the goal that the status of the machine has continued evolving powerfully. Therefore, the concept of catching the classification examples may float erratically.

Fault Detection and Classification (FDC)

When equipment and process parameters are controlled, process outcomes are also well controlled. Fault Detection and Classification (FDC) is based on the idea that you can detect changing conditions within equipment and use that knowledge to improve process. First, detect a changing condition within the equipment that results in an abnormal status, then classify the detected abnormality as a root cause failure.

Using near real-time and historical data collected from the fab floor, FDC analyzes and tests that data against models to both detect problems as they occur and react to emergent conditions. By identifying processes with frequent alarm occurrences, placing definitive metrics on consumable life, and enabling

preventative maintenance, the FDC process lowers cost by drastically reducing scrap, rework, cycle time, Test/Qual wafers, and unscheduled downtime. FDC also curbs costs indirectly by empowering device manufacturers to measure and compare equipment health, transfer and tune recipes, compare and match tools and chambers, pre-classify faults, and capture engineering "best known methods".

Diversity for Dealing with Drift

Online learning algorithms often have to operate in the presence of concept drifts. A recent study revealed that different diversity levels in an ensemble of learning machines are required in order to maintain high generalization on both old and new concepts. Inspired by this study and based on a further study of diversity with different strategies to deal with drifts, we propose a new online ensemble learning approach called Diversity for Dealing with Drifts (DDD). DDD maintains ensembles with different diversity levels and is able to attain better accuracy than other approaches. Furthermore, it is very robust, outperforming other drift handling approaches in terms of accuracy when there are false positive drift detections. In all the experimental comparisons we have carried out, DDD always performed at least as well as other drift handling approaches under various conditions, with very few exceptions[2]

Bunches of works tended to the issue of concept drifts by troupe learning strategies, which are of directed learning in machine learning. Troupe adapting first sets up various classifiers and then creates a gathering comes about by applying some voting plan to total the outcomes produced by all classifiers. For example, Minku and Yao [3] proposed a promising gathering learning technique called Diversity for Dealing with Drift (DDD), which controls data diversity to create outfit models with various degrees of diversity, in which every group display comprises of various base classifiers of a similar type, to adjust to the concept float issue.

A dispersed framework arranges organized processing gadgets to accomplish complex errands, e.g., the hole point forecast from enormous data [4][5]. MapReduce is a conveyed distributed computing framework that has gotten a great deal of consideration for huge data handling [6]. For example, Palit and Reddy [7] proposed a helped troupe classifier that encourages the concurrent cooperation of multiple figuring assets and further connected the MapReduce to expand the computational effectiveness of a gathering learning strategy. The possibility of MapReduce is based on a partition and-vanquish methodology, which separates the data into various littler data pieces, at that point embraces the Map capacity to process every datum piece in parallel to get a quick outcome, and then receives the Reduce capacity to total these prompt outcomes into a last outcome. Note that the Map and Reduce capacities can be altered by clients, and henceforth can be connected to different calculations.

This work proposes a multiple-classifier-type DDD (MDDD for short) group learning strategy for CBM in assembling ventures, which broadens the DDD with multiple classifier types, and incorporates a novel data-based adaption conspire that permits disconnected base classifiers, to expand the accuracy of the forecasting model. As a rule, in any case, gathering learning strategies incorporate complex plans and tedious classifiers, with the goal that they frequently take excessively computational time at times attributable to an excessive number of data measurements, a gigantic measure of preparing data, or an excessive number of base students in the outfit demonstrate. Therefore, the usage of the proposed MDDD based on the MapReduce framework is proposed, to expand the computational effectiveness to meet commonsense figuring prerequisites of CBM applications. Through reenactment on a benchmarking dataset with concept drifts, the performance of the proposed MDDD is confirmed.

II. RELATED WORKS

"An algorithm that implements classification, especially in a concrete implementation, is known as a classifier. The term "classifier" sometimes also refers to the mathematical function, implemented by a classification algorithm, that maps input data to a category.[1]
Given a dataset $D = \{e1, e2, \ldots, ei, \ldots\}$, in which ei represents a data point; $ei = (Xi, yi)$ in which Xi is a feature vector and yi is a class label; each element in feature vector Xi is called an attribute. Given a data point $ei = (Xi, yi)$, the process of mapping feature vector Xi to class label yi through some function f is called classification (i.e., $y \leftarrow f(Xi)$), and the function f is called a classifier.

Most data classification applications by and by are intended for the data stream, e.g., IoT gadgets connected to assembling machines ceaselessly gathers generation data on machine conditions. A concept speaks to the entire dissemination of the classification issue at some particular time point, which is described by a joint circulation P(X, y) in which X incorporates the information highlight vectors, and y speaks to their class marks. A concept float implies a difference in the dispersion [3].

Concept drifts incorporate four examples [8]: sudden float, continuous float, incremental float, and reoccurring float. For instance, consider a two-dimensional component space, including a circle A focused at (0.5, 0.5) with range 0.2 and a circle B focused at (0.5, 0.5) with sweep 0.5. Characterize the component vectors X inside hover A to be named by y = – 1, and those outside hover A to be named by y = 1. Therefore, the joint circulation P of the component vectors X with class name y = – 1 constitutes a concept. On the off chance that the joint circulation P changes from hover A to hover B as time passes by, the procedure is a sudden concept float.

By and by, concepts are insecure and change powerfully and capriciously with time, e.g., in assembling ventures, concepts for the data of machine conditions may change when machines end up maturing, broke down, or supplanted. Troupe learning strategies have been appeared to perform superior to anything single learning models [9]. In troupe taking in, various "specialists" constitute an "advisory group" that applies some voting plan to total a last predictable outcome among the prompt outcomes forecasted by all specialists. As of late, a considerable measure of gathering learning techniques for tending to concept drifts have been proposed. For example, Minku et al.

[10] employed a web based sacking procedure to set up an online group learning model. Freund and Schapire [11] tended to concept drifts by an incremental learning group show that incorporates a powerfully weighted dominant part voting plan. Minku and Yao [3] controlled data diversity to set up an online troupe learning model called DDD to address concept drifts. Gama et al. [8] reviewed a great deal of strategies for concept float adjustment.

The DDD is an online outfit learning strategy [3] comprises of three phases: gathering, detection, and adjustment. At Stage 1, two group models with high and low assorted varieties, separately, are set up through controlling the preparation dataset. At Stage 2, the early float detection strategy (EDDM) is adjusted to identify whether a concept float happens. In the event that a concept float is recognized, the DDD enters Stage 3. At Stage 3, the current preparing data indicates are adjusted prepare two new troupe models with high and low assorted varieties. At that point, new and old troupe models with high and low assorted varieties are composed to adjust to the concept float. A standout amongst the most huge commitments of the DDD is that the DDD found distinctive practices of outfits prepared by various degrees of diversity under the nearness of concept drifts, and utilized them as a stopgap measure to manage concept drifts. Moreover, it built up a splendid method to control the diversity of gatherings by only transforming one parameter with the goal that it can adjust to concept drifts effectively. Recreation comes about likewise demonstrated that DDD was extremely effective in

dealing with concept drifts. Notwithstanding, the downside of the DDD is that it must be executed by a specific type of base classifier and just a single type at any given moment. This leaves an issue: it is obscure which type of base classifier ought to be picked. And, it is the inspiration of this work to make up this insufficiency.

III. PROPOSED METHOD

All the base classifiers in the original DDD ensemble method are of the same type and can only be trained by online learning. The DDD did not consider diversity of multiple classifier types, and hence may have the bias problem of base classifiers. Bias problem is the weakness of the ensemble methods based on a single base classifier type, which may perform not well in some concepts due to the nature of this base classifier type. Although ensemble methods had been designed to address this kind of problem, the bias problem still existed when DDD uses only online bagging to establish ensembles based on one single classifier type. This work discovers this problem while testing the same dataset by several different base classifier types (also observable in Subsection IV- C). The same base classifier type always performs better for a certain certain concept, but may not for other concepts. Therefore, the precision of DDD after a concept drift may become worse, regardless of applying any base classifier type. As a result, the MDDD extends the DDD with multiple classifier types, and dynamically adjusts the

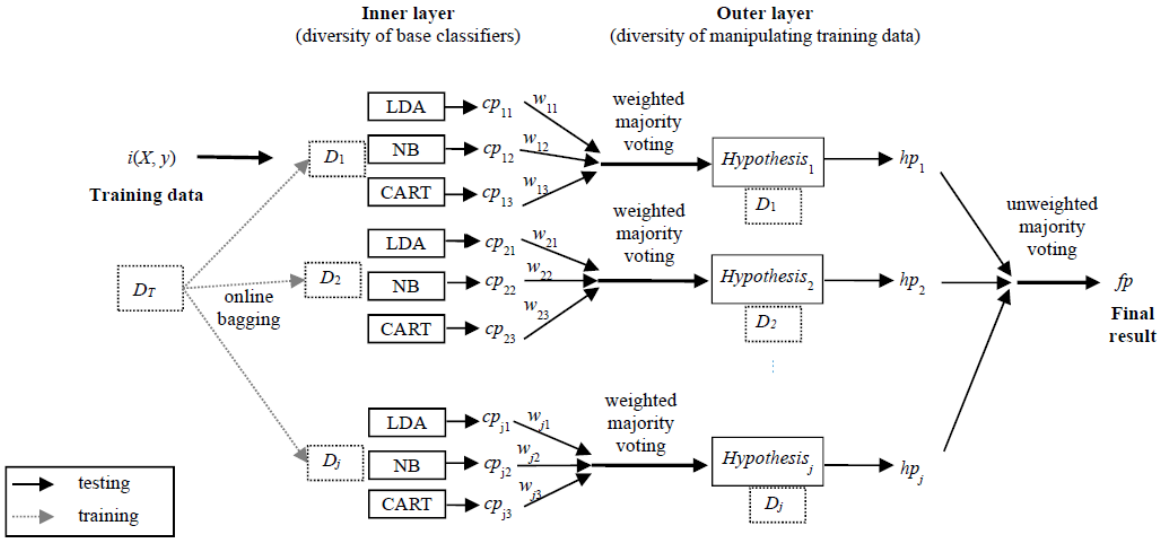

Fig 1. Illustration of Stage 1 in the MDDD.

the heaviness of every classifier to adjust to the present concept. Furthermore, a ton of classifiers with great performance can't be prepared by internet adapting however can be by disconnected learning. Therefore, unique in relation to the DDD, the MDDD permits disconnected classifier types. For disconnected learning base classifiers, the MDDD briefly stores some important data before concept drifts and then prepares these disconnected learning classifiers when required to be retrained.

Thusly, the MDDD enhances Stages 1 and 3 of the DDD and proposed a usage of the MapReduce framework. Next, this area presents the three phases of the proposed MDDD, and the MapReduce framework.

Ensemble learning

The fundamental thought of a general troupe learning strategy is as per the following. To begin with, j classifiers are built up. At that point, for each info data point, the j classifiers produce j comes about, and a specific voting plan is embraced to vote the j results to create a last outcome. Be that as it may, it is difficult to build up j distinctive classifiers. Thus, most past works just thought to be one base classifier (e.g., guileless Bayes (NB), bolster vector machine, and choice tree), and controlled the preparation data so j same-type classifiers with similar substance are prepared to wind up j same-type classifiers with various substance. That is, these works were based on diversity of controlling preparing data to maintain a strategic distance from issues of change and overfitting. In any case, such a procedure can't take care of the predisposition issue of utilizing just a single base classifier type.

To take care of the above issue, the proposed MDDD considers gathering learning with multiple diverse type classifiers. The MDDD comprises of external and inward layers (Fig. 1), in which the inward layer is based on diversity of base classifiers to maintain a strategic distance from the inclination issue of classifiers; and the external layer is based on diversity of controlling preparing data to stay away from the issue of change and overfitting.

The diversity of controlling preparing data in the external layer of MDDD is accomplished by internet stowing [14]. Given a preparation dataset DT, the external layer applies web based packing to develop j distinctive preparing datasets D1, D2, … , Dj. That is, for every datum point in DT, every one of the j preparing datasets randomly incorporates K duplicates of this data point in which K is a random variable after the Poisson dissemination of a given parameter λ.

In the internal layer, every one of the j preparing datasets D1, D2, … , Dj is additionally used to prepare r base classifiers of various types which might be prepared by on the web and disconnected getting the hang of (contingent upon the client's settings), e.g., direct discriminant examination (LDA), NB, and classification and relapse tree (CART) in Fig. 1. And, for every one of the j preparing datasets, the r base classifiers constitutes a speculation. At last, the j theories close a last outcome fp based on unweighted larger part voting (Fig. 1). The last outcome fp applies unweighted lion's share voting since it is a conglomeration of the external layer, which is a diversity of preparing data, which were created randomly and are seen as equivalent. Unique in relation to the collection of the internal layer, base classifiers have contrasts in nature.

To show the base classifiers with better accuracy performance in every theory, the voting plan in the internal layer takes after the accompanying weighted greater part voting procedure. Consider modifying the weights of r base classifiers in a speculation. Each weight is introduced with an esteem 1/r. Each time when a base classifier creates an outcome, the outcome is contrasted and the real outcome. On the off chance that the two outcomes are the same, the weight is increased with γ; else, it is separated by γ. At long last, all weights in a similar speculation are standardized for later utilize.

While introducing the MDDD (i.e., at the preparation arrange), web based stowing is embraced on the preparation dataset DT to create two noteworthy datasets called old-low-diversity and old-high-diversity, individually, in the K factors in the

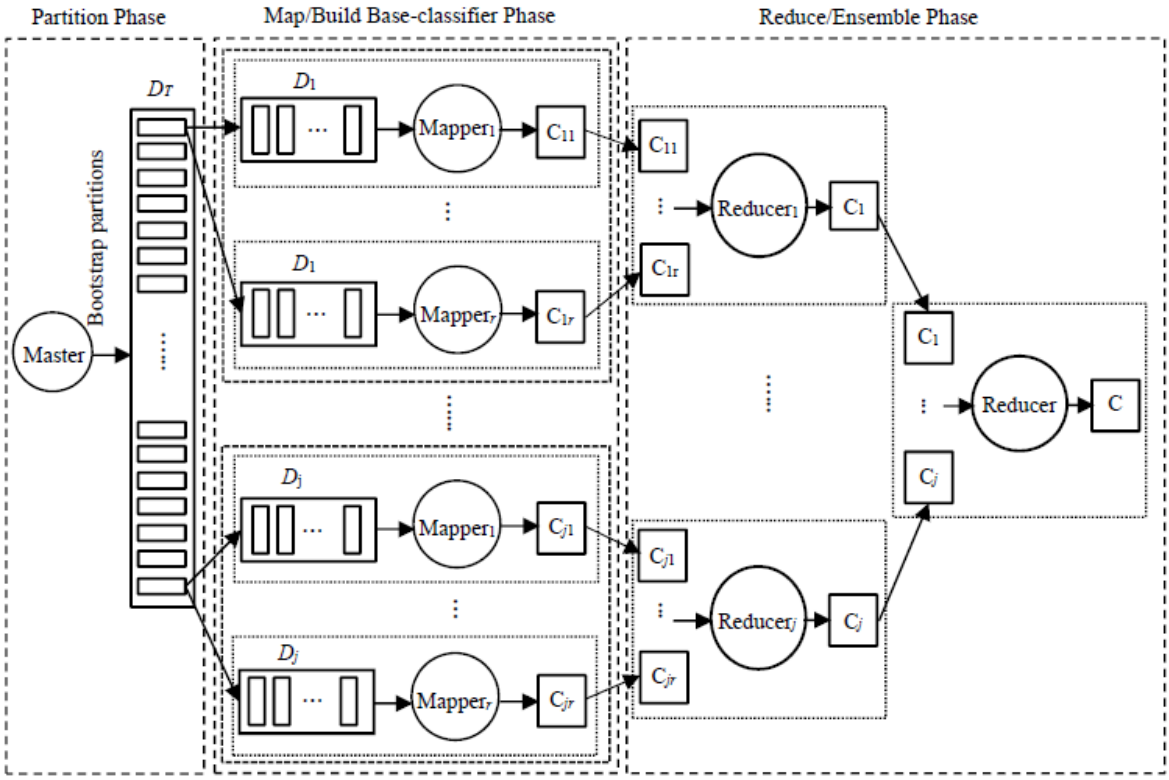

Fig 2. The MDDD based on the MapReduce framework.

old-low-diversity (resp., old-high-diversity) major dataset takes after a Poisson dissemination with λ = 1 (resp., λ = 0.001). Note that each major dataset comprises of j preparing datasets, and the diversity speaks to the distinction level of the j preparing datasets produced by internet sacking. While instating the MDDD, the j preparing datasets in the old-low-diversity major dataset are received to prepare the underlying r classifiers in jth theory, yet the old-high-diversity major dataset isn't utilized until adaption at Stage 3.

Subsequent to preparing the MDDD, consider the testing stage, i.e., every datum point in the testing data is tried by the j speculations. On the off chance that the class name of this testing data point forecasted by the MDDD is the same with the genuine class mark, at that point the MDDD is perceived to get a right forest; generally, a forecast fault. At that point, the exactness performance of the MDDD can be processed.

Unique in relation to the DDD that uses each new testing data point to prepare various base classifiers of a similar type (e.g., NB classifier) at Stage 1, the MDDD does not. Be that as it may, the MDDD stores the new data point just when a notice level is resolved at Stage 2 (which will be presented later), and sits tight for a float level to be resolved at Sage 2 to lead web based packing based on these data focuses to retrain the j speculations.

Concept float detection

After a testing result is acquired at Stage 1, Stage 2 is to distinguish whether a concept float exists after the information data point is considered. The same with the DDD, this work applies the EDDM [12] to recognize concept drifts. The possibility of EDDM is to figure the "separation" between two

sequential forecast faults, in which the purported "remove" is characterized as the quantity of the data focuses that are ordered effectively between two back to back forecast faults. In perfect, if more exact outcomes are acquired amid the learning procedure, the separation between two back to back forecast faults is longer. Then again, if the separation between faults ends up shorter, it infers that a concept float happens.

The EDDM judges event of a concept drifts through whether the separation between two back to back forecast faults has a measurably momentous lessening. The points of interest of EDDM are given as takes after. To begin with, let di be the separation between the (I – 1)th and the ith faults since the last concept float. At that point, let di′ , dmax′ , and si′ be the normal, most extreme, and standard deviation of every one of these separations until the ith faults, separately, since the last concept float. And, let s′s'$_{max}$= {s ′ , s ′ ,..., s′} . Given a α esteem, if the info data point is the ith forecast fault since the last concept float, at that point a notice level is identified if the accompanying disparity holds:

$$(d_i' + 2 \cdot s_i') / (d_{max}' + 2 \cdot s_{max}') < \alpha$$

Given a β esteem, a float level is distinguished if the accompanying disparity holds:

$$(d_i' + 2 \cdot s_i') / (d_{max}' + 2 \cdot s_{max}') < \beta$$

In the event that the MDDD enters a notice level, the information testing data focuses after the notice level are put away. On the off chance that the MDDD enters a float level, the event of a concept float is resolved. Note that notice and float levels speak to the levels of centrality of increment of the forecast fault recurrence of the model at a specific time point. Notwithstanding, when di′ surpasses the notice level, and di′□1 is not as much as the notice level, these data focuses put away for later utilize are erased, and it is viewed as a false caution for the notice level. That is, the notice level can not exclusively be utilized for putting away the data focuses for later utilize, yet in addition be utilized for judge false cautions. In any case, the float level is utilized just for judge event of a concept float.

Adjustment

At the point when a concept float is identified at Stage 2, Stage 3 conducts web based packing with λ = 1 (resp., λ = 0.001) on the data focuses gathered from the notice level to the float level to produce a noteworthy dataset called amazing failure diversity (resp., new-high-diversity). Next, old-high-diversity and amazing failure diversity major datasets are joined to prepare j speculations to fill in as the outfit at Stage 1 of the MDDD, and the new-high-diversity major dataset replaces the old-high-diversity major dataset. The motivation behind why to do as such is clarified as takes after. After a concept float, the low-diversity group prepared by old-low-diversity major dataset will lose its precision quickly and should be retrained by other major datasets. The extraordinary failure diversity major dataset can adjust to an event of a concept float quickly. It is instinctive to retrain group with amazing failure diversity major dataset yet it will dispose of all the information estimation of the old data, which may be helpful to manage the new concept. From encounters of past works, the old-high-diversity major dataset with high diversity would perform superior to anything the old-low-diversity major dataset after concept drifts. What's more, the old-high-diversity major dataset still saves the estimation of old data. Henceforth, the two noteworthy datasets are embraced to retrain low-diversity outfit demonstrate after a concept float.

At last, when a low-diversity gathering model is prepared, the new-high-diversity major dataset replaces the old-high - diversity dataset to save the information of this concept, and the MDDD backpedals to Stage 1 for later testing.

MapReduce-based MDDD

Since the MDDD includes an extensive number of base classifiers of various types, the MDDD runs not productively on a PC with just a single processor, which can't meet the necessity of genuine CBM applications. From the writing, the MapReduce framework has been produced to expand the effectiveness of troupe learning. For example, Hegazy et al.

[13] proposed four group learning models based on the MapReduce framework. Therefore, this work proposes the usage of the MDDD based on the MapReduce framework (Fig. 2) to increment computational productivity. The framework comprises of three stages: data segment, the foundation of base classifiers by Map capacities, and troupe by Reduce capacities.

Stage 1 applies web based stowing on DT to create j datasets D1, D2, … , Dj. For every one of the j datasets, Phase 2 receives r Mappers to fill in as r preparing classifiers to create r prompt outcomes. At Phase 3, since Stage 1 of the MDDD incorporates external and inward layers, we receive two layers of Reduce capacities to total the outcomes created by all Mappers. The primary layer of Reducers receives weighted dominant part voting to total all the r prompt consequences of every theory. At that point, the second layer of Reducer embraces the unweighted greater part

voting to total every one of the outcomes into a solitary last outcome fp. Since preparing and testing the MDDD takes the most

computational time, the proposed MapReduce framework just thinks about the outline of preparing and testing MDDD, and alternate parts of the MDDD (e.g., setting up high-diversity and low-diversity major datasets, and identifying concept drifts) are accomplished by the Master hub.

While introducing the MDDD, the Master hub receives web based stowing on DT to create old-high-diversity and old-low-diversity major datasets. At that point, the MDDD embraces the old-low-diversity dataset and the Map capacity to set up j × r base classifiers. At the point when a testing data point enters the Master hub, we enter Stage 1 of the MDDD. The Master hub passes the testing data point to every Reducer. At that point, every Reducer receives Map capacities (base classifiers) to forecast the preparation data point, and the outcomes are called prompt outcomes. At that point, the weighted lion's share voting is received on these quick outcomes to get last outcome fp. At that point, the MDDD enters Stage

2. The Master hub at that point receives the EDDM to check whether the last outcome fp causes a concept float. On the off chance that the event of a concept float is affirmed, we enter Stage 3 of the MDDD. At that point, internet sacking is led on the data focuses put away from the notice level to the float level to set up new-high-diversity and extraordinary failure diversity major datasets and consolidates amazing failure diversity and old-high-diversity major datasets. At that point, the MDDD embraces the consolidated major dataset and the Map capacity to prepare j classifiers to serve the gathering for later utilize. At that point, the MDDD backpedals to Stage 1 of the MDDD.

IV. EXPERIMENTAL IMPLEMENTATION AND RESULTS

This section first introduces the experimental data, then shows the experimental analysis of the proposed MDDD, and then compares the experimental results using multiple classifier types and only one classifier type.

A. Experimental data

The experimental dataset is a benchmarking dataset used by the streaming ensemble algorithm called SEA [15], including 60,000 data points, each of which has three attributes and a real class label; the value of each attribute is a real number between 0 and 10 following a uniform distribution; two of the three attributes are relevant with the real class label, but the other attribute is irrelevant. The class label is binary. If the sum of relevant attributes is greater than a given threshold, the real label is 1; otherwise, 0. The simulation processes each data point of the dataset sequentially, to simulate the feature of data stream. For each 15,000 data points, we suppose a concept drift in which the threshold changes from one value to another value, and hence the dataset has four thresholds and three concept drifts. In the experimental setting, the thresholds are 8.0, 9.0, 7.5, and 9.5 in the ordering of time, and each 15,000 data points have the same concept. The patterns of concept drifts considered in the experiment are abrupt drifts. In addition, we assume that 10% of the data in the same concept has noise. That is, the class labels of the noise data points are opposite to the real class labels in the SEA dataset, to increase the difficulty in forecasting the classification of this data set. As shown in Fig. 3, the SEA dataset is divided into two parts: the first 1000 data points serve as the training data, and the remaining 59,000 data points serve as the testing data. In the experiment, the MDDD reads each data point in the training dataset sequentially, forecasts the class label of the data point, and the real class labels of the data points that have been read have been known.

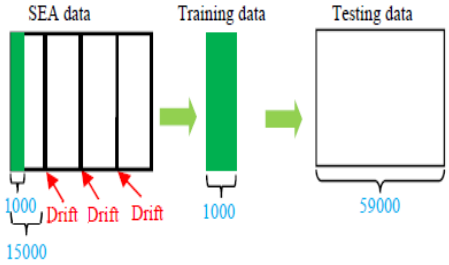

Fig 3. Illustration of partitioning experimental data.

B. Experimental analysis of the MDDD

This section analyzes whether the MDDD is successful in detecting and adapting to concept drifts, and the performance of the experimental results. The plots of accuracy versus the number of data points that has been tested are shown in Fig. 4, in which the positions of three concept drifts are marked by green dash lines (i.e., 14000, 29000, 44000); the blue solid-line curve represents the overall forecast accuracy for the data points that have been processed so far; and the red dotted-line curve represents the accuracy for the data points processed after a concept drift is detected. Note that the first concept starts from -1000 because the first 1000 data points (denoted from -1000 to 0) are reserved as the training data and thus have no accuracies. From Fig. 4, the MDDD is successful to detect the right occurrence times of three concept drifts. In addition, the MDDD can adapt to the concept drifts and raise the overall accuracy to 87.98%. In this experiment, the inner layer of MDDD consists of three types of base classifiers (i.e., LDA, NB, and CART), and the outer layer consists of 13 hypothesis, making 39 base classifiers in total.

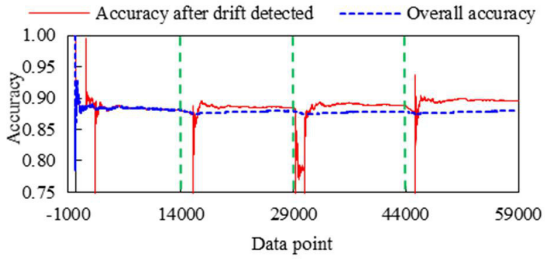

Fig. 4. Experimental results of analyzing the accuracy of the MDDD.

C. Comparing the results between using multiple classifier types and only one classifier type

Fig. 5 shows the experimental comparison of accuracies of the proposed MDDD (i.e., with three classifier types: LDA, NB, and CART) and three MDDDs with only one single classifier type (i.e., LDA, NB, and CART, respectively). The three MDDDs with only LDAs, NBs, and CARTs are denoted by DDD-LDA, DDD-NB, and DDD-CART, respectively.
From Fig. 5, the accuracy of the proposed MDDD performs better than that of the other three models.

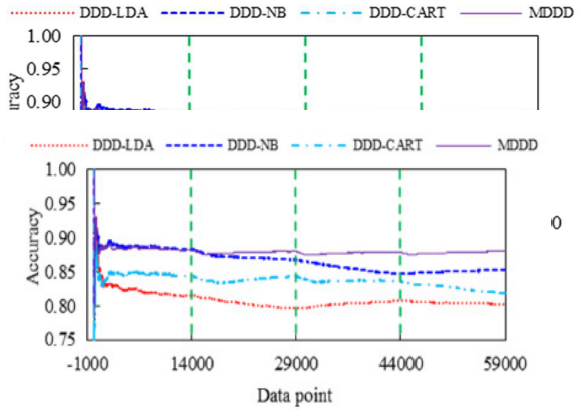

Fig. 5. Comparison of the MDDD with other classifiers.

The accuracies of the four models for the testing data points divided by three concept drifts are shown in Table 1. From Table 1, the proposed MDDD performs better than the other three MDDDs with only one classifier type in all parts except it losses DDD-NB a bit for the first part of the testing data points. The other three MDDDs with only one classifier type have no consistent conclusion on performance for different parts of the testing data points. In addition, from these experimental results, the advantage of the weighted majority voting in the MDDD can be observed. The weight (reflecting the precision) of a classifier in the ensemble is always updated to be applied, so that the overall precision increases.

TABLE. 1. ACCURACY OF FOUR MODELS FOR FOUR PARTS OF DATA

Data points / Model	1~14000	14001~29000	29001~44000	44001~59000
DDD-LDA	0.816	0.778	0.830	0.786
DDD-NB	0.883	0.853	0.808	0.867
DDD-CART	0.843	0.843	0.821	0.766
MDDD	0.881	0.878	0.876	0.883

V. CONCLUSION

This work has proposed a MDDD group learning technique for CBM with concept drifts. The MDDD enhances the DDD with multiple classifier types and diversity of classifiers and preparing data, to dodge the issue of fluctuation and overfitting and to proficiently identify and adjust to concept drifts. To build the computational proficiency, the execution of the MDDD based on the MapReduce framework has been proposed. Reproduction comes about demonstrate that the MDDD can effectively distinguish the correct event time of concept drifts and adjust to them.

REFERENCES

1. *Alpaydin, Ethem (2010). Introduction to Machine Learning. MIT Press. p. 9. ISBN 978-0-262-01243-0.*
2. L. L. Minku and X. Yao, "DDD: A New Ensemble Approach for Dealing with Concept Drift," in *IEEE Transactions on Knowledge & Data Engineering*, vol. 24, no. , pp. 619-633, 2011. doi:10.1109/TKDE.2011.58
3. Wang, Y. Wang, Y. Sun, S. Guo, J. Wu, "Green industrial Internet of things architecture: An energy-efficient perspective," *IEEE Communications Magazine*, vol. 54, no. 12, pp. 48-54, 2016.
4.C.-C. Lin, D.-J. Deng, J.-R. Kang, S.-C. Chang, and C.-H. Chueh, "Forecasting rare faults of critical components in LED epitaxy plants using a hybrid grey forecasting and harmony search approach," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 6, pp. 2228-2235, 2016.
5.L. L. Minku and X. Yao, "DDD: A new ensemble approach for dealing with concept drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 4, pp. 619-633, 2012.

[4] K. Wang, H. Lu, L. Shu, J.J.P.C. Rodrigues, "A context-aware system architecture for leak point detection in the large-scale petrochemical industry," *IEEE Communications Magazine*, vol. 52, no. 6, pp. 62-69, 2014.
[5] K. Wang, L. Zhuo, Y. Shao, D. Yue, K. F. Tsang, "Toward distributed data processing on intelligent leak-points prediction in petrochemical industries," *IEEE Transactions on Industrial Informatics*, vol. 12, no. 6, pp. 2091-2102, 2016.

[6] H. Ke, P. Li, S. Guo, and M. Guo, "On traffic-aware partition and aggregation in MapReduce for big data applications," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 3, pp. 818-828, 2016.
[7] I. Palit and C. K. Reddy, "Scalable and parallel boosting with MapReduce," *IEEE Transactions on Knowledge and Data Engineering* vol. 24, no. 10, pp. 1904-1916, 2012.
[8] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, Article ID: 44, 2014.

[9] Y. Freund and R. E. Schapire, "Experiments with a new boosting algorithm," in *Proc. of the 13th International Conference on Machine Learning (ICML 1996)*, pp. 148-156, 1996.
[10] L. L. Minku, A. P. White, and X. Yao, "The impact of diversity on online ensemble learning in the presence of concept drift," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 5, pp. 730-742, 2010.

[11] R. Elwell and R. Polikar, "Incremental learning of concept drift in nonstationary environments," *IEEE Transactions on Neural Networks*, vol. 22, no. 10, pp. 1517-1531, 2011.

[12] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, "Early drift detection method," in *Proc. of 4th International Workshop on Knowledge Discovery from Data Streams*, vol. 6, pp. 77-86, 2006.

[13] O. Hegazy, S. Safwat, and M. El Bakry, "A MapReduce fuzzy techniques of big data classification," in *Proc. of SAI Computing Conference (SAI)*, IEEE Press, pp. 118-128, 2016.
[14] N. C. Oza, "Online bagging and boosting," in *Proc. of 2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, pp. 2340-2345, 2005.

[15] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proc. of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ACM Press, pp. 377-
[16] 382, 2001.