IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 18, 2018, accepted August 20, 2018, date of publication August 31, 2018, date of current version September 21, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2868114

A Multi-Tier Stacked Ensemble Algorithm to Reduce the Regret of Incremental Learning for Streaming Data

R. PARI, M. SANDHYA, AND SHARMILA SANKAR
Department of Computer Science and Engineering, B. S. Abdur Rahman Crescent Institute of Science and Technology, Chennai 600048, India
Corresponding author: R. Pari (pari_ramalingam@yahoo.com)

ABSTRACT Incremental Learning (IL) is an exciting paradigm that deals with classification problems based on a streaming or sequential data. IL aims to achieve the same level of prediction accuracy on streaming data as that of a batch learning model that has the opportunity to see the entire data at once. The performance of the traditional algorithms that can learn the streaming data is nowhere comparable to that of batch learning algorithms. Reducing the regret of IL is a challenging task in real-world applications. Hence developing an innovative algorithm to improve the ILs performance is a necessity. In this paper, we propose a multi-tier stacked ensemble (MTSE) algorithm that uses incremental learners as the base classifiers. This novel algorithm uses the incremental learners to predict the results that get combined by the combination schemes in the next tier. The meta-learning in the next tier generalizes the output from the combination schemes to give the final prediction. We tested the MTSE with three data sets from the UCI machine learning repository. The results reveal that MTSE is superior in performance over the SE learning.

INDEX TERMS Incremental learning, stream data mining, ensemble learning, stacked generalization, stacked ensemble and classification accuracy.

I. INTRODUCTION

A streaming data emanates continuously from one or more data sources at high speed [1], [2]. The volume of data received from the data sources is vast and varies from one instant to another instant [3]. Though it is possible to store them on disk, processing and analysing them using multiple passes is challenging [4], [5]. As soon as they arrive, they are processed and discarded [6]. Due to the high frequency of arrival and high volume of data, some of the data are discarded without even getting processed [7]. In streaming data, it is necessary to train the model with different sets of data emanating from various sources at different points of time [8], [9]. There is a need to update the model incrementally with the new sets of data without impacting the performance of the model [10]. The process of learning continuously on streaming data and updating the model to adapt to the new sets of data is called incremental learning [1], [2], [8], [11]. There are two common scenarios in incremental learning. (i) The data received from the sources at each of the time instants contain the same inherent pattern in them (ii) New patterns evolve with the new data received from the sources. Accordingly, the data set is said to have a stationary target and a dynamic target [12]–[14]. When the target is dynamic in nature, the data set is said to have a concept drift [15]–[17]. The probability distribution of the data set which has the concept drift, varies with time [13]. In such data sets, the model trained at previous instants of time is not consistent with the data received at the current instant. Hence the model is updated for every new set of data received from the sources [1], [18]. In some extreme cases, the old models are discarded, and new models are created.

In incremental learning, the model evolves with every set of data made available for training [19]. The model dynamically adapts to the new patterns in the new set of data [20]. As the model evolves, the performance of the model keeps improving with every set of new data [11], [21]. The performances of such models are far below the performance of the models which have the opportunity to see the entire data. Incremental learning aims to reduce the regret [22]–[25]. The Regret of the incremental learning is the difference between the performances of incremental and full batch models [26]. The primary challenge in reducing the regret is that the endpoint of incremental learning keeps moving with every arrival of a new set of data [21]. Even if the regret is reduced at

instant 'T', with the arrival of data at 'T + 1', there is no guarantee that the regret continues to reduce or remains at the same level. To reduce the regret of incremental learning, the algorithm needs to increase the accuracy of the models and also adapt to the concept drift [27]. Rather than using a single classifier, ensemble learning gives better accuracy [1], [28]–[30]. When multiple weak learners carry out the incremental learning, their predictions can be combined using an ensemble algorithm [14], [31]. The combined predictions are more accurate than the predictions of the individual incremental learners [32]. Instead of using a static function like average or weighted sum, trainable combiners generalizes well [33]–[37]. This study goes one step further by using multiple ensemble algorithms and generalizing the predictions from the ensemble algorithms using another classifier. Thus it uses an MTSE algorithm for incremental learning. MTSE possesses the following characteristics which are mandatory for stream data mining.

- Ability to read and process blocks of data at a time, rather than processing all the data as a whole.
- Each block of data is processed only once
- Use a fixed amount of memory irrespective of the size of the data set
- Ability to stop the algorithm at any time and get the best prediction

In this study, MTSE simulates the steady flow of streaming data from a large static dataset. Hence the data stream is not handled in a pure stream fashion. The static dataset is divided into multiple partitions, and each partition is considered as a data stream at an instant 'T'. Thus the training happens in mini-batches with every new stream of data received from the source.

The contributions of this study are as follows. (i) Introduce a multi-tier stacked ensemble algorithm for incremental learning (ii) Demonstrate that the algorithm minimizes the regret and (iii) Motivate the intellectual community to make use of this algorithm for real-time classification problems with streaming data. The rest of the sections of this paper are organized as follows. Section II describes the related work in this area of data analytics. After explaining the proposed method or approach in Section III, the experimental evaluation is depicted in Section IV. Section V is the conclusion and the scope for future work.

II. RELATED WORK

Training a new model for every new concept in the data stream and keeping them in sleep mode helps to reactivate the old models if there is a recurrence of concept. Detecting the concept drifts and the recurrence of concepts helps to decide whether a new model needs to built or any of the old models is appropriate. Gama and Kosina [38] used a two-layered learning system for mining dynamic data streams. The first layer was trained on the data stream and the second layer was also trained using the same data stream. The class label of the second layer was decided based on the correctness of the first layer classifier. The second layer classifier acts as a meta-learner to learn about the performance of the first layer classifier. Depending on the delay in receiving the actual labels of the data stream the second layer was trained, only after the actual labels were received and the correctness of the first layer classifier was evaluated. The arrival of the actual labels also updated the first layer classifiers. The first stream of data was used only for training the base classifier. From the second stream onwards, both base classifier and the meta-learner were trained. The number of examples after which the meta-learner started to train was a hyper-parameter. The evolution of performance measures and the error rates were monitored to detect concept drift. Whenever there is a concept drift, the meta-learners predicted whether the old models were applicable or not. Accordingly either a new model was trained or one of the previously learned models was activated. If the number of concepts in the data stream is large, then the number of models to be maintained also becomes large. Hence in the case of concept drift, it took more time to decide whether a new model needs to be trained or an old model needs to be reactivated.

The stream mining community explored the use of ensemble learning for streaming data extensively. It is often used for streams where adapting to the concept drift is essential. Some of the studies used SE to tackle the dynamic nature of streaming data and got better accuracy. Bifet *et al.* [39] trained multiple Hoeffding trees with different subsets of features and generalized the class probabilities of their predictions. Simple perceptrons with sigmoid activation functions were used as generalizers. To minimize the squared loss, the stochastic gradient was used to train the perceptrons. Initially, equal weights were assigned to all the nodes and were updated based on gradient descent. The weights were updated in-line with the learning rate of the perceptrons. Whenever concept drift was detected, the learning rate was reset to a larger value. ADaptive sliding WINdow (ADWIN) was used to detect the concept drift. Though this approach improved the accuracy to some extent and also was coping with the concept drift, it is more dependent on the size of the feature subset, the number of feature subsets and the learning rate of perceptrons. It is challenging to optimize these parameters and to get the near optimal values.

When the streams keep arriving continuously from multiple sources, it is difficult to model the underlying function of the complete data set. Canzian *et al.* [40] used local learners for each of the data sources and combined the predictions with a weighted majority rule to arrive at the final prediction. The aggregation rules were updated to make it more relevant to the dynamic nature of the data. Here the dynamic nature was referred to the concept drift in the data. When the real label of the data was available, the prediction error was calculated. Based on the prediction errors, the weights of the aggregation were updated. The technique was named as Perceptron Weighted Majority (PWM). This technique assumed that each learner could have its statistical dependency on its observations and the actual labels. The aggregation rules were updated when the local prediction was not matching with the

actual label. The misclassification probability was used as the performance measure to assess the technique. The number of wrong predictions per instance was considered as the misclassification probability. The misclassification probability of this technique reached the minimum when the misclassification probabilities of each of the local classifiers were at a minimum. This technique achieved the performance gain of 34 to 71 % over the existing solutions. They achieved the misclassification probability of 31.5 % for one of the data sets which contained the concept drift. As this value is very high, this technique is suitable only for a data set which does not have concept drift.

Combining the predictions of a set of adaptive classifiers using meta-learners helps to deal with the concept drift and at the same time achieves high accuracy. Frías-Blanco *et al.* [41] used Fast Adaptive Stacking of Ensembles (FASE) for learning from non-stationary data streams. A two-sided variant of Hoeffding-based Drift Detection Method was used to estimate the error rate of the adaptive learners. 0-1 loss function was used to calculate the error. Meta-instances were generated using a predictive sequential approach. Based on the concept drift, the base classifiers were alternated or in some cases removed. The meta-learner was also an adaptive learner to handle the dynamic changes in the meta-instances. The number of base classifiers and the confidence level for the drift detection was used as hyperparameters for FASE. The performance of FASE was on par with similar ensemble methods like OzaBag, OzaBag-HDDM and OzaBagAD-WIN. Model alternations for every new concept with a smaller set of examples worsened the accuracy. There was an additional cost associated with the classifier replacements in case if the number of examples for the new concepts is small.

Most of the incremental learning algorithms work reasonably well for the balanced data sets and are not suitable for imbalanced data set. Wang *et al.* [42] proposed Over-sampling-based Online Bagging (OOB) and Under-sampling-based Online Bagging (UOB). The study used random oversampling and random under-sampling to increase or decrease the number of minority class instances. A Poisson distribution parameter ''λ'' was used for performing the re-sampling. The integration of random sampling with the bagging algorithm had a positive impact on improving the minority-class recall and G-mean. Based on the ANOVA analysis, it was found that data distribution is the more influential factor than the imbalance rates and the types of base classifiers. The weighted ensemble of OOB and UOB assigned the weights based on the normalized G-mean. These weights were either proportional weights or binary weights and accordingly the strategies were named WEOB1 and WEOB2. Both WEOB1 and WEOB2 achieved better performance than that of OOB and UOB individually. The performance of these strategies for data streams with concept drift was not studied.

Neural activities produce the data continuously and hence can be considered as a streaming data. Mining this stream helps to treat the brain related diseases. The challenge here is that this stream needs to be processed in real time and the predictions are made in real time. The accuracy of such predictions is critical for treating the patients. Lee *et al.* [43] used Incremental Linear Discriminant Analysis to decode the neural signal. Data flow for the algorithm was implemented using DSPCAD Lightweight Data flow environment and was tested with the traces of the animal's neurons. Animal's behavior data was recorded for five days and three times a day. Accuracy was used as the performance measure. The algorithm showed a tremendous improvement in the execution time. The accuracy of the algorithm was closer to the batch Linear Discriminant Analysis algorithm. While experimenting, for each of the test cases only one increment was considered. Hence the experiment was not reflecting the true nature of incremental learning using streaming data. The change in the concept was not simulated in the experiment, and hence the applicability of this approach for dynamic data streams is not known.

Bifet [3] used Leveraging Bagging (LB) for classifying the examples in the evolving data streams. They leveraged the performance of bagging with two improvements. They used resampling with replacement and output detection codes for achieving the randomization improvement. Poisson distribution was used for increasing the weights of resampling. They assigned a binary string of length 'n' to each of the class labels. They trained each of the classifiers on one-bit position. For each of the new instances, their proximity with the binary string is checked, and as per that, they are assigned to the corresponding classes. They used random output codes to increase the diversity of the ensemble. The random output code is a form of voting system with the flexibility of correcting the incorrect votes. ADWIN was used for detecting the change in the error code of the classifiers. Whenever there was a change in the error, classifiers with higher error were replaced with a new classifier. They implemented the algorithms using Massive Online Analysis. They used four artificial data sets namely Streaming Ensemble Algorithm (SEA) Concept Generator, Rotating Hyperplane, Random Radial Basis Function (RBF) Generator and Light-Emitting Diode (LED) Generator for testing their algorithms. They also tested the algorithms with three real-world data sets namely Forest Cover Type, Poker-Hand and Electricity. Hoeffding Naive Bayes Tree was used as a base learner. Three different bagging strategies namely subagging, half subagging and bagging were used. Though these strategies were performing better than ADWIN in terms of the time, their accuracies were less. When compared with random forest, LB showed improvement.

Enabling Random Forest to adapt to the concept drifts in the evolving data streams is a state-of-the-art strategy. Gomes *et al.* [44] used an updated version of the Adaptive Random Forest (ARF) for evolving data streams. They combined the characteristics of the batch algorithm with dynamic adaptation. They used the Hoeffding Tree algorithm as a base classifier. Drift adaptation was handled by training a background tree when a warning was detected. The base

model was replaced by the background tree only when the drift occurred. Rather than considering all the features to make the leaf split decision, they limited it to a feature subset. For online bootstrap aggregation, they used Poisson distribution with a λ value of 6. They ensured that the number of background trees is less than the maximum number of trees. They achieved the parallelism by managing the operations of each tree in separate threads. In addition to the accuracy, they also captured the Kappa M and Kappa Temporal to support the data sets with class imbalance or temporal dependencies. They used ten synthetic and six real data sets for testing their algorithm. They compared the six different variations of their algorithm. From the experimental results, they concluded that ARF yielded good classification accuracy, Kappa M and Kappa Temporal. The memory usage and the processing time of ARF was less than the state-of-the-art ensemble classifiers. The limitation of ARF was that it was not improving the performance for data sets where all the features are critical for building a model.

The proposed approach is different from all the above approaches. It creates an ensemble of ensemble system. The exception is that it does not use any ensemble classifiers like Random Forest, AdaBoost, *etc.* Instead, it only uses a set of combination schemes to combine the predictions of the base classifiers and a generalizer to generalize the predictions of the combination schemes. The generalizer in the proposed approach is a simple classifier. In this study the results of the proposed approach is compared with the results of Adaptive Random Forest, Leveraging Bagging and the traditional SE algorithm.

III. METHODOLOGY

The diversity of the base learners and a proper choice of generalize ensures that the accuracy of stream mining using SE is not impacted by the concept drift [39]. Though concept drift is handled well by SE, the accuracy is not improving beyond a certain level. Hence there is a scope for fine-tuning SE and improving its accuracy. The literature discusses some of the best ways of fine-tuning SE [34], [45]. This study intends to fine-tune SE by introducing a new tier in between the base classifiers and the meta-learner. This study takes the stance that the accuracy of classification is improved when a new tier is introduced between the incremental learners and the meta-learner in SE. Figure 1, Figure 2 and Figure 3 depict the different tiers in MTSE.

The base tier which is also called the incremental tier is the first tier. Different combination schemes form the ensemble tier. The combined predictions from these schemes along with the top three critical features form the input space for generalization tier. Hence the bias and variance are controlled in different tiers. Using multiple incremental learners with cross-validation ensures that the bias is reduced in the base tier. The combinations schemes and the meta-learners reduce the variance. Hence a perfect balance between the bias and the variance is achieved. As a result of this, the classification accuracy improves. The main difference between MTSE and

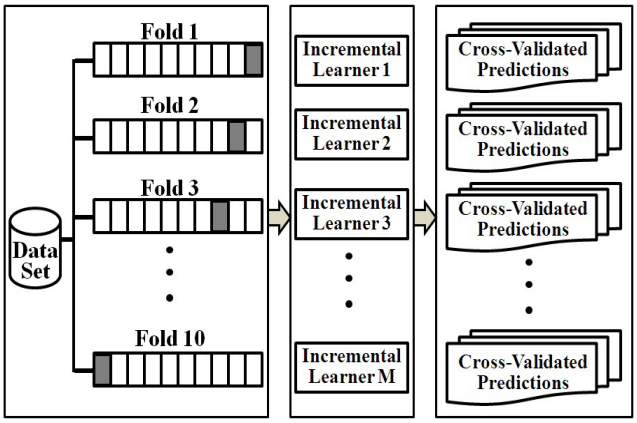

FIGURE 1. The Base Tier of the MTSE Algorithm. Classification by Incremental Learners to produce the cross validated predictions.

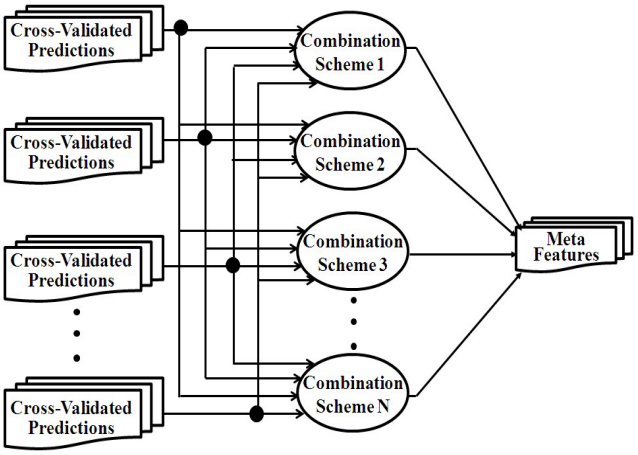

FIGURE 2. The Ensemble Tier of the MTSE Algorithm. Classification by Ensemble Learners to produce the Meta-Features.

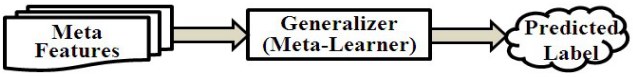

FIGURE 3. The Generalization Tier of the MTSE Algorithm. Classification by Meta-Learner to produce the final prediction.

SE discussed in the literature is the introduction of a new tier. Most of the studies found in the literature used Meta-learning as part of the ensemble process. Whereas, this algorithm separates the entire combination schemes together as a separate tier. The output of this tier forms the input space for meta-learning. At the same time, this algorithm still uses the most popular principle of the ensemble. For a given data set, different classifiers are effective and efficient within different subsets of data.

Using a wide range of suitable learners along with 10-fold cross validation reduces the bias. Hence MTSE uses a set of incremental classifiers with the partial fit as weak learners in the base tier. These learners produce the cross-validated predictions. The process of producing the cross validated

predictions using the following classifiers is depicted in Algorithm 1.

- Stochastic Gradient Descent (SGD)
- Passive Aggressive Classifier (PAC)
- Gaussian Naïve Bayes (GNB)
- Multinomial Naïve Bayes (MNB)
- Bernoulli Naïve Bayes (BNB)
- Perceptron (Linear Model)
- Multi Layer Perceptron (MLP)

The ensemble tier uses the following combination schemes. Plural Voting (PV), Majority Voting (MV), Weighted Majority Voting (WMV) and Confidence-Based Voting (CBV). PV and MV use the cardinality to make the decision. WMV assigns the weights based on the class probabilities of the incremental learners. CBV decides based on class probability. The generalization is made at the final tier using Logistic Regression (LR). Algorithm 2 depicts how the cross validated predictions from the base tier are combined to produce the meta-features. Algorithm 4 depicts how the optimal weights of the base classifiers are obtained using brute-force approach. When the number of base learners is more than seven, the time complexity of the brute-force approach becomes unmanageable. Hence the number of base learners is chosen as either seven or less than seven.

The predictions from the ensemble tier and the top three features from the original space are used as input for LR. The top three features are selected by analysing the correlation between each of the features and the class labels. Algorithm 3 depicts how the meta-features along with the critical features from the original space are used for training the generalizer. Algorithm 3 also depicts how the regret of MTSE is calculated.

As depicted in Figure 4, this algorithm trains a model on the first stream of data received from the source. After that for every new stream of data received from the source, the model predicts the labels for those examples. Based on the predictions, the MR of the model is calculated. In line with the calculated MR, the model is updated. The entire process is repeated for every arrival of a new stream of data from the source. At any instant, only one model exists, and its parameters are updated based on the streams of data received till that instant. Hence the model adapts to the change in the target concept and thus taking care of concept drift in the data stream. By maintaining an updated model at any instant, MTSE supports any-time learning. MTSE can be stopped at any point in time to get the best prediction then. Whenever a new stream of data is received from the source, the model predicts the labels for those examples. The model's performance is evaluated. Thus it tracks how the accuracies of the models are improving with every new stream of data.

In each of the epoch, MTSE calculates the "Regret" to find out how close this model is to the model trained by the offline or batch algorithm. For this purpose, the batch algorithm is initially trained on 80 % of the data and tested with the remaining 20 % of the data. Testing the batch algorithm helps in calculating the loss of the batch algorithm.

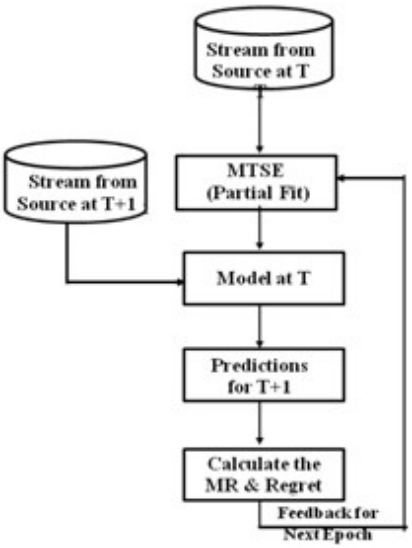

FIGURE 4. Use of Multi-tier Stacked Ensemble Algorithm for Streaming Data. Describes how the data received at any instant is used for training the model and how the model is tested with the data received for the next instant.

The difference between the cumulative loss of the models generated by MTSE till that instant and the loss of batch algorithm gives the regret at that instant. The performance of SE is also captured as a benchmark so that the performance of MTSE is compared with SE.

A. MATHEMATICAL MODEL

Let $D_1, D_2, \ldots, D_T$ be the stream of data received from the source at difference instants of time.

Where, $D_1 \bigcup D_2 \bigcup \ldots \bigcup D_T = D$.

$$In\ general,\ D_T = \begin{bmatrix} x_{T11} & x_{T12} & \cdots & x_{T1d} & y_{T1} \\ x_{T21} & x_{T22} & & x_{T2d} & y_{T2} \\ & \vdots & & & \vdots \\ x_{Tz1} & x_{Tz2} & \ldots & x_{Tzd} & y_{Tz} \end{bmatrix}$$

Where'd' is the number of features (dimensions) in the data stream and 'z' is the size of each streams simulated.

$$Input\ data\ is\ X = \begin{bmatrix} x_{T11} & x_{T12} & \cdots & x_{T1d} \\ x_{T21} & x_{T22} & & x_{T2d} \\ & \vdots & & \\ x_{Tz1} & x_{Tz2} & \cdots & x_{Tzd} \end{bmatrix} \quad (1)$$

$$The\ Label\ is\ y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_z \end{bmatrix} \quad y_1, y_2, \cdots y_z \epsilon \{-1, +1\} \quad (2)$$

In the case of multi-class classification problem, $y_1, y_2, \cdots y_N \epsilon \{c_1, c_2, \ldots, c_l\}$ where $c_1, c_2, \ldots, c_l$ are the class labels.

1) BASE TIER

Let $IC = \{IC_1, IC_2, \cdots, IC_m\}$ be the set of incremental classifiers.

At any instant, when the data stream D_T is received from the source, it is partitioned into ten equal subsets at 10 %. Let $D_{T1}, D_{T2}, \cdots, D_{T10}$ be the equal size sets, partitioned from the set D_T.

Fold 1: The training set $T_T = D_{T1} \cup D_{T2} \cup \ldots D_{T9}$

The testing set $V_T = D_{T10}$

Train the incremental classifiers using T_T. The incremental classifiers IC_m map the elements of T_T into one of the classes in Y.

$$IC_m : T_T \rightarrow Y \tag{3}$$

$$IC_m : X_{Tg} \rightarrow \{c, c_2, \ldots, c_l\} \tag{4}$$

Where $X_{Tg} \in T_T$

In the next fold, D_{T9} is taken as the testing set and the remaining nine sets combined together are taken as the training set. This is repeated until all the partitioned sets in S_{T1} are taken as testing sets. This results in the predictions for all the instances in S_{T1}. Let $IL_T = \{IL_{T1}, IL_{T2}, \ldots, IL_{Tm}\}$ be the labels predicted by the incremental classifiers at any time instant 'T'.

$$i.e.\ IL_{Tm} = IC_m(X_{Tg}) \tag{5}$$

2) ENSEMBLE TIER

Let $EC = \{EC_1, EC_2, \ldots, EC_m\}$ be the set of ensemble classifiers, and they combine the labels predicted by the incremental classifiers and maps them into one of the classes in Y.

$$EC_m :\ IL_{Tm} \rightarrow \{c_1, c_2, \ldots, c_l\} \tag{6}$$

$$EC_m = Aggregation\ of\ (IL_{Tm}) \tag{7}$$

Here the Aggregation of IL_{Tm} is dependent on the choice of the combination scheme used in the ensemble tier. Let $EL_T = \{EL_{T1}, EL_{T2}, \cdots, EL_{Tm}\}$ be the labels predicted by the ensemble classifiers at any time instant 'T'.

$$EL_{Tm} = EC_{Tm}(IL_{Tm}) \tag{8}$$

3) GENERALIZATION TIER

Let GC be the generalization classifier, and it combines the labels predicted by the ensemble classifiers and maps them into one of the classes in Y. This tier takes the set of ensemble labels ($\{EL_{t1}, EL_{t2}, \cdots, EL_{tm}\}$) and the actual labels (Y) and produces the generalized label. This can be mathematically represented as

$$GL_m : EL_{Tm} \rightarrow \{c_1, c_2, \ldots, c_l\} \tag{9}$$

Where GL_{Tm} is the final result produced by the m^{th} meta-learner at T^{th} time instant.

4) MISCLASSIFICATION RATE OF THE MODEL

In this study, MR of the classifiers at T^{th} instant is calculated by using the set of data received at $(T+1)^{th}$ instant as the test set. Hence the testing is always done with the unseen examples. The ratio between the counts of wrongly classified instances to the total number of instances in the test data gives MR of the classification.

$$\textbf{\textit{Misclassification Rate}}\ (\boldsymbol{MR_t}) = \frac{(FP + FN)}{(TP + FP + TN + FN)} \tag{10}$$

5) REGRET OF THE MODEL

Regret is defined as the difference between the cumulative loss of the incremental models and the loss of the batch or offline model. The cumulative loss is calculated by summing up the loss of the offline models for all the instants. The regret of the incremental model over different instants is calculated as follows.

$$\textbf{\textit{Regret}} = L - \sum_{t=1}^{T} L_t \tag{11}$$

Where 'L' is the loss of the model trained on the entire data set by the batch algorithm. 'L_t' is the loss of the model trained by MTSE at any particular instant 'T'. Regret is calculated for MTSE as well as SE. Calculating regret for both MTSE and SE helps in comparing their performance for different data sets.

B. ALGORITHM

1) BASE TIER

See Algorithm 1.

2) ENSEMBLE TIER

See Algorithm 2.

3) GENERALIZATION TIER

See Algorithm 3.

4) BRUTE FORCE ALGORITHM

See Algorithm 4.

IV. EXPERIMENTAL EVALUATION

A. EXPERIMENTAL SETUP

Scikit-learn library in Python was used to implement the experiment [46]. Three real-world data sets from the UCI Machine Learning repository [47] which are very popular in the data mining community were used for the experiment. These data sets are listed in Table 1. The following are the reasons for picking these three data sets for conducting our experiments. (i) They have a large volume of data (ii) They have a large number of features with mostly numerical or categorical data types and (iii) The data is evenly distributed across different classes. The large volume helps in splitting each data set into multiple data streams. All the missing values or NaN values were replaced with the most frequently occurring values depending upon the nature of the features. For every data set, ten runs of experiments were conducted to ensure the consistency of the results. At the end of all the trials, the minimum, the maximum, the mean and the standard deviation of the performance metrics were calculated.

Algorithm 1 Classification by Incremental Learners

Input: Data streams at instants 'T' and 'T + 1' (D_T and D_{T+1})
Output: The labels (IL_{mT}) and (IL_{mT+1}) predicted by the incremental learners
Randomly Split D_T into ten partitions (folds) of equal size. $D_T = \{F_1, F_2, F_3, \ldots, F_{10}\}$
Do for m = 1, 2, . . . , M:
  Do for i = 1, 2, . . . , 10:
    $L_i \leftarrow \bigcup_{j=1}^{j=10} F_j \; if \; i! = j$
    $V_i \leftarrow F_i$
    Partially fit an incremental learner IC_m with L_i
    Using the model obtained in step 8, **predict** the labels for V_i and name them as IL_i
  End
  Combine the predictions obtained from step 9 into a single vector IL_{mT} (Predictions from m^{th} incremental learner for the data stream at instant 'T'.
End
Repeat steps 3 through 12 with D_{t+1} and get the labels IL_{mT+1}
####Calculate the MR of the incremental learners
$s \leftarrow Size(D_{t+1})$
Do for m = 1, 2, . . . , M:
  FP_FN ← 0
  Do for j = 1, 2, . . . , s:
    If $IL^j_{mT+1}! = y_j$:
      FP_FN++
    End if
  End
  $MR_{mT} \leftarrow \frac{FP\_FN}{s}$
End
Return the predictions (IL_{mT}) and (IL_{mT+1}) obtained in step 11 and step 13
End

In the above table, CR refers to the class ratio of majority class to the rest of the classes.

The data set is split into 'n' number of partitions to simulate the streaming data from the static data set. Each partition represents the data stream received from the source at a particular instant. Hence 'n' number of partitions represents the data stream received from the source at 'n' number of different instants. Incremental models are built and updated for each of these 'n' instants using MTSE algorithm. The experiment is repeatedly conducted for different values of hyper-parameters as shown in Table 2.

MR of the batch learner is required to calculate the regret of incremental learning. Hence the data set as a whole was trained using a batch algorithm resulting in an offline model. This model was tested with the test data, and thus MR of this offline model was calculated. For each of the 'n' different instants, MR of the incremental model was calculated.

Algorithm 2 Classification by Ensemble Classifiers

Input: The labels predicted by the incremental learners IL_{mT} *and* IL_{mT+1}
Output: The labels predicted by the ensemble classifier EL^i_{tj}
Plural Voting:
Do for n = 1, 2, . . . , N:
  $EL^n_{1T} \leftarrow Mode\ of\ \left(IL^n_{1T},\ IL^n_{2T} \ldots IL^n_{mT}\right)$
End
Simple Majority Voting:
Do for n = 1 to N:
  Corresponding to each class label, initialise a counter IL_i to zero
  //Iterate for each of the base learner
  Do for m = 1, 2, . . . , M:
    For each occurrence of a class label in the classifier's prediction, increment the corresponding counter
  End
  $Max\_Count \leftarrow Max(AllCounters)$
  If max_Count > M/2:
    EL^n_{2T} ← The class label corresponding to Max_Count
  End if
End
Confidence-Based Voting:
For all the instances, store the class probabilities of all the classifiers for all the classes into an array
Do for n = 1, 2, . . . , N:
  $Max\_Prob \leftarrow$ Max(class probabilities of all the classifiers for all the classes)
  $EL^n_{3T} \leftarrow EL^n_{3T} \leftarrow$ the class label corresponding to Max_Prob
End
Weighted Majority Voting:
Use the Brute Force Algorithm to find the optimal weights of the base learners. $W = \{W_1, W_2, W_3, W_4, W_5, W_6, W_7, W_8\}$
$WTot \leftarrow W_1 + W_2 + W_3 + W_4 + W_5 + W_6 + W_7 + W_8$
Do for n = 1, 2, . . . , N:
  Do for k = 1, 2, . . . , L: // Repeat for each class label
    $WSum_{nk} \leftarrow 0$
    Do for m = 1, 2, . . . , M //Repeat for each weak classifier
      //Find the sum of the products of the weights (W_m) and the class probabilities (P_{mk})
      $WSum_{nk} \leftarrow WSum_{nk} + W_m * P_{mk}$
    End
  End
  $WAvg \leftarrow \frac{WSum_{nk}}{WTot}$
  $EL^n_{4T} \leftarrow$ *Class label with the highest weighted average*
End
Repeat steps 3 through 34 with IL_{mT+1} and get the predictions EL_{mT+1}
Calculate the Accuracy of the ensemble classifier
End

Algorithm 3 Classification by Meta-Learner

Input: The labels predicted by the ensemble classifiers EL_{mT} *and* EL_{mT+1}

Output: The final labels (GL_{mT}) predicted by the generalizer

Calculate the correlation between each of the features in the input space and the class label using numpy.corrcoef method in numpy library of Python. //Picking the critical features

$$r = \frac{N\sum xy - (\sum x) * (\sum y)}{\sqrt{\left[N\sum x^2 - (\sum x)^2\right]\left[N\sum y^2 - (\sum y)^2\right]}}$$

$CF_t \leftarrow$ ***Pick*** *three features with high corelation with* the *class labels*

Combine the critical features and the ensemble predictions $TS_T \leftarrow CF_T \cup EL_{mT}$

Train the generalizer with the set TS_t

Run the model obtained in step 6 with the set EL_{mT+1}

Calculate the loss of the generalizer for the current stream

Calculate cumulative loss of MTSE

Calculate the Regret of MTSE

End

TABLE 1. Summary of data sets.

Data Set	Network Intrusion
Description: It consists of a series of TCP connection records, labelled as either a normal connection or an intrusion attack.	
# Instances : 459943 **# Features** : 42 **# Classes** : 6 **CR:** 65.9 %	
Objective: Predict the good connections (Normal) and bad connections (Intrusions or Attacks)	

Data Set	Electricity Pricing
Description: It consists of the electricity pricing information for the Australian New South Wales electricity market. It has a binary label (UP or DOWN).	
# Instances : 45313 **# Features** : 42 **# Classes** : 2 **CR:**57.5 %	
Objective: Predict whether the electricity price change moves UP or DOWN	

Data Set	Forest Cover Type
Description: It consists of four wilderness area of Roosevelt National Forest in Northern Colorado. The instances of the data set represent the $30X30$ meter cell of the areas	
# Instances : 581012 **# Features** : 55 **# Classes** : 7 **CR:** 64.5 %	
Objective: Predict whether the instance belongs to the first forest type or other forest types.	

The number of misclassified instances in each of the instants was added to get the cumulative count of misclassification instances. Dividing this data by the total number of instances

Algorithm 4 Find the Optimal Weights of the Base Classifiers

Input: The labels predicted by the base classifiers IL_{mT} *and* IL_{mT+1}

Output: The optimal values of the weight $W = \{W_1, W_2, W_3, W_4, W_5, W_6, W_7, W_8\}$

max_$acc \leftarrow 0$

Do for $W_1 = 1, 2, 3, \ldots, 8$:

Do for $W_2 = 1, 2, 3, \ldots, 8$:

Do for $W_3 = 1, 2, 3, \ldots, 8$:

Do for $W_4 = 1, 2, 3, \ldots, 8$:

Do for $W_5 = 1, 2, 3, \ldots, 8$:

Do for $W_6 = 1, 2, 3, \ldots, 8$:

Do for $W_7 = 1, 2, 3, \ldots, 8$:

Do for $W_8 = 1, 2, 3, \ldots, 8$:

Combine the labels of the base classifiers using these weights and class probabilities

Calculate the accuracy using the actual labels

If acc > max_acc:

max_$acc \leftarrow acc$

weight_vector $\leftarrow \{W_1 W_2 W_3 W_4 W_5 W_6 W_7 W_8\}$

End if

End

End

End

End

End

End

End

End

Return the weight_vector

End

TABLE 2. Hyper-Parameters used in this algorithm.

Hyper-parameters of Incremental MTSE	
# Streams	The number of partitions made from the data set to simulate the data stream. The experiment is repeated by simulating 10, 25, 40 and 50 instants.
Training Data Percentage	The percentage of data considered for training. The data at any instant is split as per this ratio to yield the training & testing data. The experiment is repeated for (i) 90 % (ii) 80 % (iii) 75 % and (ii) 60 % Training data

in the test data gives the overall MR of MTSE. The difference between the MR of the batch algorithm and the overall MR of MTSE gives the regret of MTSE.

Benchmarking: Massive Online Analysis (MOA) framework was used to conduct the benchmarking for LB and ARF. Both LB and ARF were tested with the same three data streams (Forest Cover Type, Network Intrusion and Electricity Price Change). Table 3 depicts the configurations

TABLE 3. Configurations of LB and ARF in MOA framework.

Algorithm	Leveraging Bagging
Base Learner	HoeffdingAdaptiveTreeClassifLeaves
Evaluator	BasicClassificationPerformanceEvaluator
Evaluation Strategy	EvaluateInterLeavedTestThenTrain
Drift Detection Method	ADWIN ChangeDetector
Warning Detection Method	ADWIN ChangeDetector

Algorithm	Adaptive Random Forests
Base Learner	ARFHoeffdingTree
Numerical Estimator	GaussianNumericAttributeClassObserver
Nominal Estimator	NominalAttributeClassObserver
Split Criterion	InfoGainSplitCriterion
Evaluator	BasicClassificationPerformanceEvaluator
Evaluation Strategy	EvaluateInterLeavedTestThenTrain

of LB and ARF in the MOA framework. The benchmarking experiments were also repeated for ten runs to ensure the consistency of results.

B. RESULTS AND DISCUSSION

For each of the three data sets, this study analyses the performances of LB, ARF, SE and MTSE. It compares the accuracies of these approaches. This analysis gives the insight about the performance of MTSE over the other three approaches. As the experiments were repeated for ten runs, the basic statistics about the accuracy (the minimum value, the maximum value, the mean value and the standard deviation) is reported. This is shown in Table 3. For Forest Cover Type and Network Intrusion, the mean accuracy of MTSE is better than the mean accuracies of LB, ARF and SE. For Electricity Pricing, LB performs better than all other three approaches. The size of the data streams simulated using Electricity data set was much smaller than the size of the data streams simulated from the other two data sets. When the size of the data streams simulated using Network Intrusion and Forest Cover Type were large, MTSE performed better than other three approaches. Hence for larger data streams, MTSE achieves higher accuracy than LB, ARF and SE. When compared with SE, MTSE performs better for all the data streams. In the case of Forest Cover Type, MTSE gives superior performance.

As the proposed approach is an extension of the traditional SE algorithm, further analysis is done to compare the performance of the proposed approach with the performance of SE. A comparison of the regret of SE with the regret of MTSE for different values of the hyper-parameters was carried out. This analysis helps to determine if the hyper-parameters have any impact on the regret of MTSE. This study analyses the MRs of the classifiers in different tiers for each of the iterations within each of the data sets. This study uses a matrix of graphs to perform this analysis. This matrix of graphs considers the data sets as rows and either the number of time instants or the training data percentage as columns. Figure 5 depicts the bar chart matrix of Regret for different number of data streams simulated using the three data sets.

For all the data sets, irrespective of the number of streams simulated, the regret of MTSE is better than that of SE. The negative value of regret for all the experiments indicates that the performance of MTSE is not only superior to SE but also to the batch algorithm. Amongst the 3 data sets used for experimenting; MTSE gives the best performance for forest cover type data set. For this data set, when the number of streams is 10, the regret of MTSE reaches the least value of -6.35 %. There are two reasons for this behavior. (i) Size of the data set is huge and (ii) the classes are evenly distributed within the data set. When MTSE splits the data set into ten partitions to simulate the streaming data, the sizes of each of the partitions are still large enough to produce a stable model with optimal bias and variance. As the model is already stable, it requires very less fine tuning with every new set of data corresponding to the time instants. The distribution of classes within each of the partitions is also contributing to the stability of the model. Even though MTSE randomly partitions the data sets, there is a perfect distribution of classes within the partitions. Here, the incremental models get better and better with every time instant. Hence it helps in reducing the MR in each of the time instants, and thus the overall regret is also at a minimum.

After the forest cover type data set, MTSE performed better for electricity price change data set. Here, when the number of instants is 40, MTSE achieves the best performance with the least regret value of -5.26. For network intrusion data set, the performance of MTSE is just as good as the batch algorithm. Though it is not giving a considerable improvement in performance over the batch algorithm, the regret is still on the negative side. Because all the individual classifiers in the base tier achieve the same performance level and they are not complementing each other. Hence when MTSE combines them in the ensemble tier, there is no considerable improvement in the performance. All the combination schemes in the ensemble tier are also providing almost the same output. The combinations schemes are also not complementing each other. Hence the generalization tier is not showing any considerable improvement in the performance. When the number of time instants is less than 10, incremental models start to behave like the offline models.

Figure 6 depicts the bar chart matrix of Regret across the tiers for different percentages of training data across the data sets.

The analysis reveals that the percentage of training data is not having any major impact on the performance of MTSE. For all the data sets, irrespective of the percentage of training data, the regret of MTSE is better than that of SE. The negative value of regret of MTSE indicates that MTSE performs better than the batch algorithm also. The regret is negative for all the percentages of training data. By digging deep into these graphs, they reveal that there is an indirect correlation

TABLE 4. Comparison of performance of MTSE, ARF, LB and SE. This table depicts the basic statistics for accuracy, for all the four approaches.

Data Set	MTSE				ARF				Leveraging Bagging				SE			
	Accuracy				Accuracy				Accuracy				Accuracy			
	Min	Max	Mean	SD	Min	Max	Mean	SD	Min	Max	Mean	SD	Min	Max	Mean	SD
Forest Cover Type	72.74	73.23	72.94	0.13	61.80	61.80	61.80	0	61.25	61.25	61.25	0	65.56	66.22	65.91	0.18
Network Intrusion	99.71	99.91	99.78	0.06	99.41	99.41	99.41	0	99.52	99.52	99.52	0	99.41	99.63	99.51	0.07
Electricity Price Change	86.19	87.29	87.09	0.17	89.61	89.61	89.61	0	83.20	83.20	83.20	0	84.05	85.90	84.73	0.67

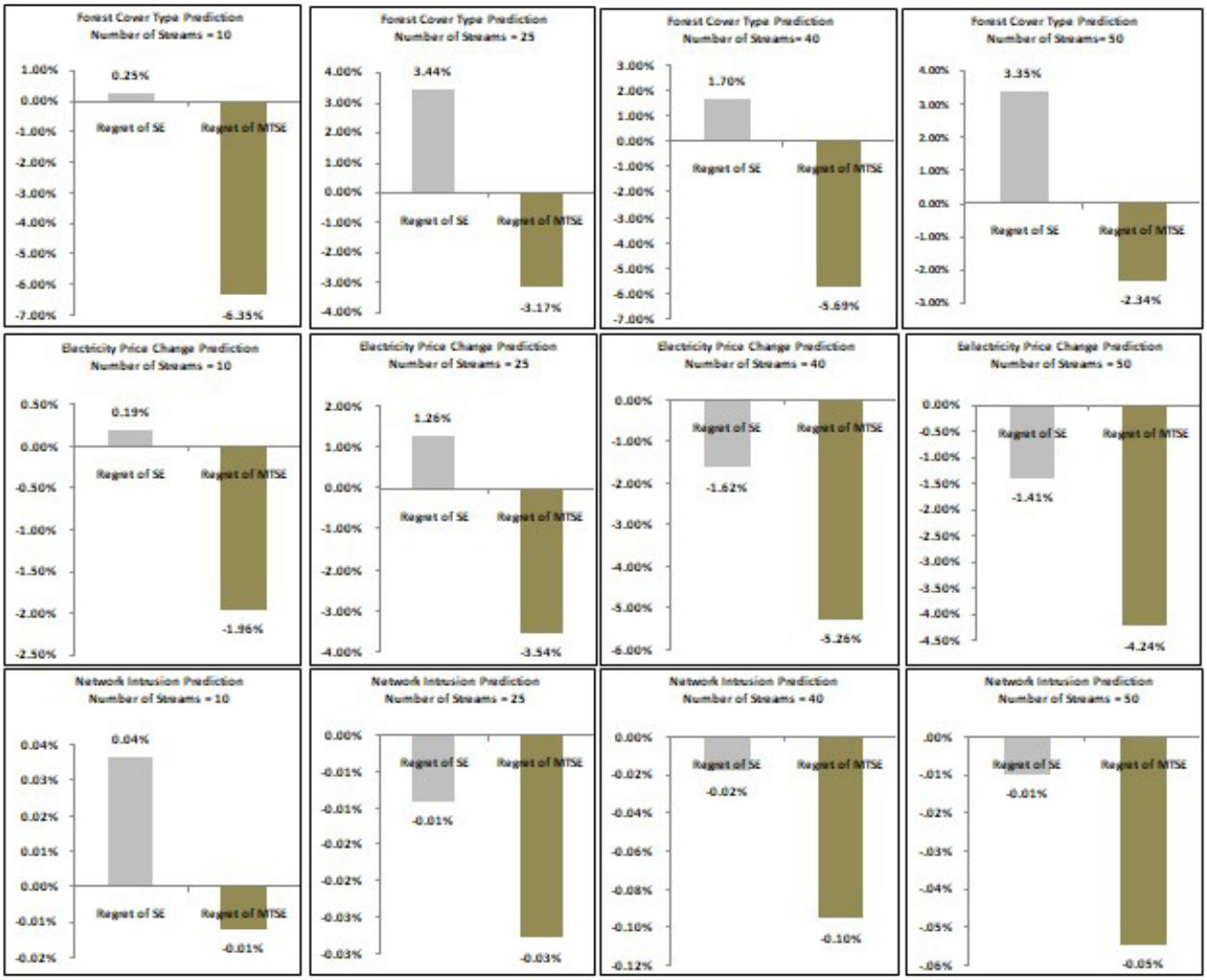

FIGURE 5. Regret of SE and MTSE. Regret is plotted for SE and MTSE for different number of Streams across the Data sets. Numbers of Streams considered are 10, 25, 40 and 50.

between the performance of SE and MTSE for different percentages of training data.

Figure 7 depicts the trend of regret of MTSE against the number of time instants for each of the data sets. The trend lines of all the three data sets lie below the horizontal axis. This confirms our earlier inference that MTSE's performance is better than that of the batch learning. This graph also reveals that the regret of MTSE is not having a direct correlation with the number of time instants. Hence the regret of MTSE does not either increase or decrease with the increase in the number of time instants. For network intrusion data set the trend line almost coincides with the horizontal axis. i.e. the regret is just below zero. It means that the cumulative MR of MTSE is almost the same as the MR of the batch algorithm. This is mainly because all the base classifiers almost produce the same predictions and all the combination schemes produce the same output. Hence generalising them with meta-learning is not reducing the MR. Figure 8 depicts the trend of regret of MTSE against the training data percentage for each of the data sets.

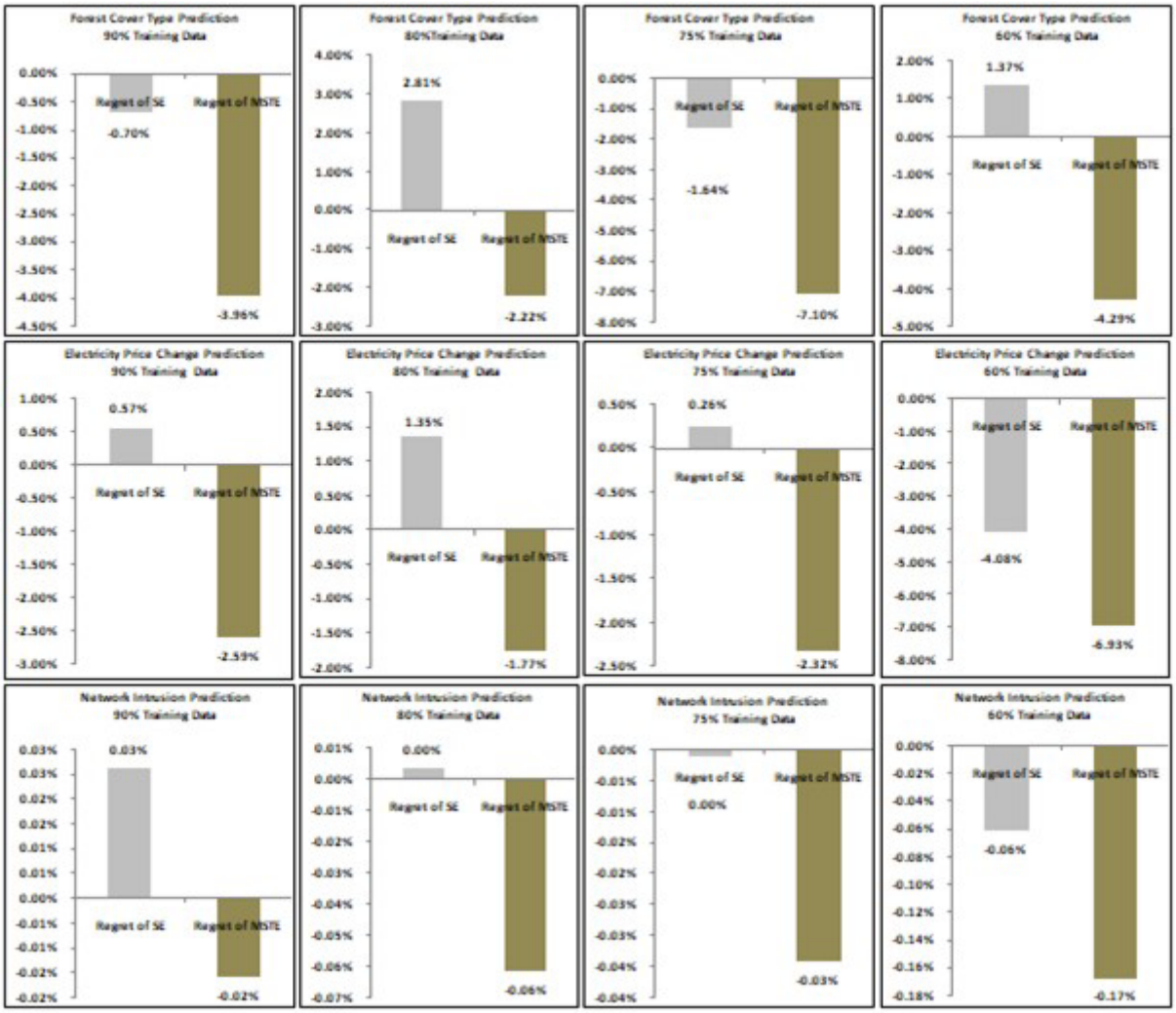

FIGURE 6. Regret of SE and MTSE for different percentages of training data across the Data sets. Percentages of training data considered are 90, 80, 75 and 60.

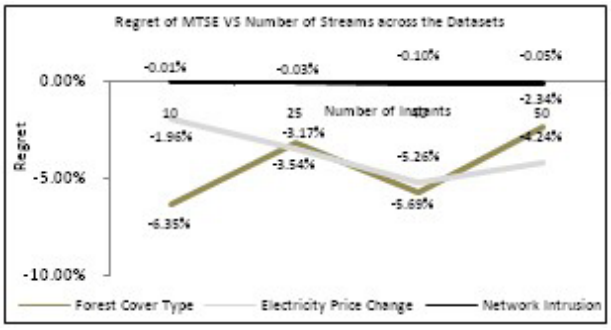

FIGURE 7. Regret of MTSE against the number of streams.

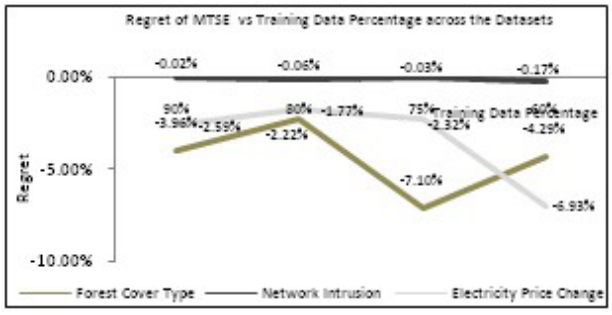

FIGURE 8. Regret of MTSE against the training data percentage.

For each of the data sets, the MR of SE and MTSE are plotted for comparative analysis. Figure 9 depicts how MR of SE and MR of MTSE are varying for network intrusion data set. For both SE and MTSE, MR hits the lowest value during the initial time instants and increases after 2 or 3 instants. It becomes stagnant for later time instants. The low value of MR irrespective of the number of time instants shows that the accuracy is close to 100 %. Hence the scope for reducing the MR is less in Network Intrusion data set. This, leads to less reduction in the regret. Every feature within this data

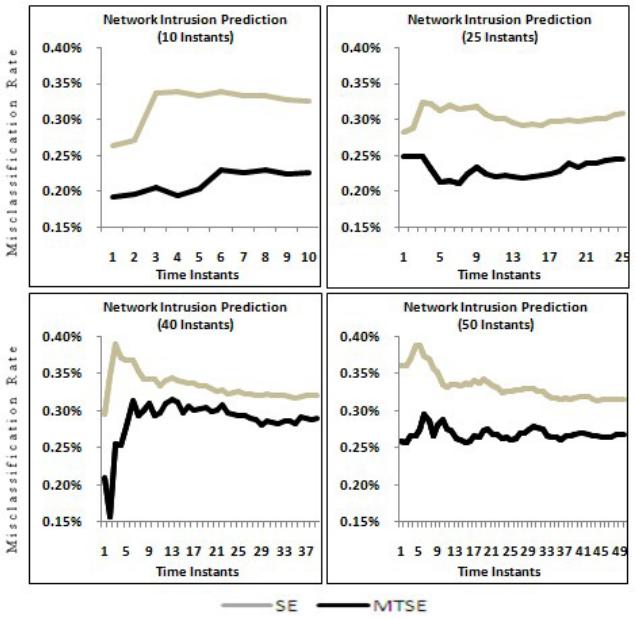

FIGURE 9. Misclassification rate – network intrusion data set.

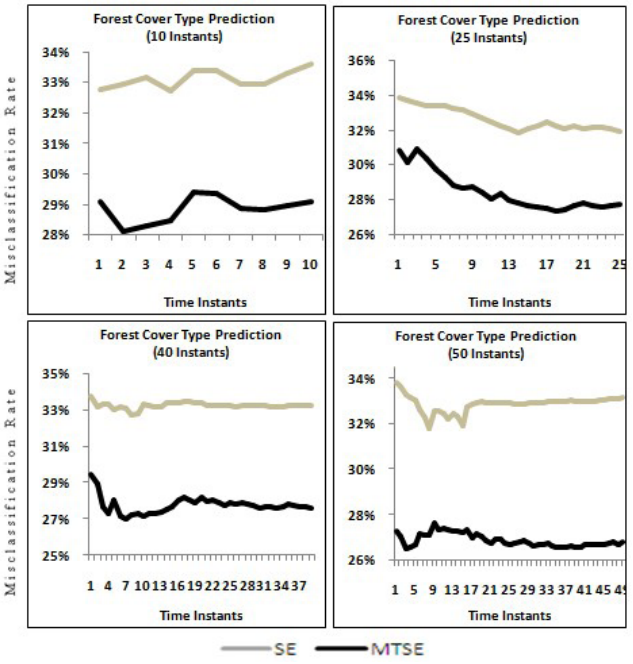

FIGURE 10. Misclassification rate – forest cover type data set

set is having a direct correlation with the class labels. Hence picking the top three features to form the input space for generalization tier is not helping in any way to minimize the MR and thus to reduce the regret.

Figure 10 depicts how the MR is varying for forest cover type data set, with every new set of data received at every time instant. MR hits the peak value in the very first instant and reduces after 2 or 3 instants. It becomes stagnant for later time instants. For this data set, MR varies between 26 and 34. When the number of time instants is 50, it is hovering around 26.5. There is a significant difference between the MR of SE and MTSE. This difference is in the range of 4 to 8. The highest value of 8 is reached when the number of time 50 instants is 50. Except for a few time instants, there is a synchronous behavior between the trend of SE's MR and MTSE's MR.

Figure 11 depicts how the MR is varying for electricity price change data set, with every new set of data received at every time instant. MR starts with a high value, reduces after 2 or 3 instants and climbs again. It reaches the peak towards later time instants and drops again slightly. In all the cases, the trend of MTSE's MR is below the trend of SE's MR. When the number of time instants is 50; this difference reaches the maximum value of 4. When the number of time instants is 25, this difference reaches the minimum value of 2. The fluctuations in the MR of all the three tiers are happening at the same instant of time. Hence there is synchronous behavior between the trend of MTSE's MR and other tier's MR.

One common observation across the data sets is that the MR of MTSE is much lower than that of SE. This positively impacts the regret of MTSE. It reduces the regret by over 100 % when compared to SE. Hence MTSE is

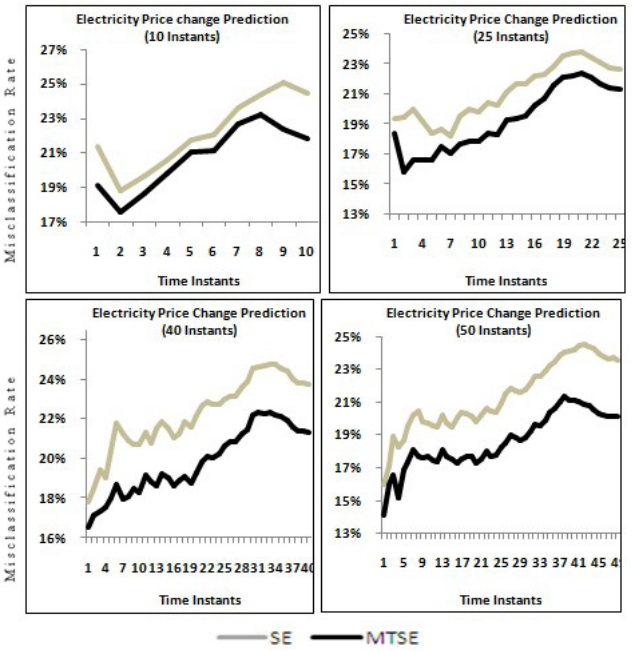

FIGURE 11. Misclassification rate – electricity price data set.

the "go-to" algorithm for incremental learning in the real-time classification problems involving streaming data.

V. CONCLUSION

This study introduced a new algorithm called MTSE for incremental learning using streaming data. MTSE is an extension of SE algorithm. MTSE ensures that the accuracy keeps increasing in every tier. Thus it makes the incremental model's accuracy to be better than the accuracy of the

models trained by batch algorithms. In challenging cases, the accuracy of MTSE is as close as possible to that of batch algorithms. The regret is either negative or tends to be closer to zero. The experimental results show that MTSE reduces the regret by over 100 % when compared to SE. This implies that MTSE is superior over SE and also better than the batch learners. When compared with LB, MTSE performed better for all the three data sets. When compared with ARF, MTSE performed better for two data sets. Hence we conclude that MTSE is a better choice than ARF and LB for real time incremental learning problems.

The limitation of this approach is that applying the cross validation for data streams where the labels do not become available in the next stream; rather they become available at the predefined interval. For example in real time, the electricity price change label becomes available at the end of each day.

We plan to extend this study by carrying out the analysis of run time of MTSE. We want to consider the parallel learning of the base learners to minimize the run time of MTSE. We also want to explore the impact of feature engineering on the regret of incremental learning. Selecting the optimal features reduce the regret further and make MTSE even more reliable solution for data stream mining.

REFERENCES

[1] C. Aggarwal, *Data Streams: Models and Algorithms*, vol. 31. 2007.
[2] C. C. Aggarwal and J. Wang, *Data Streams: Models and Algorithms*, vol. 31. 2007, pp. 9–38.
[3] A. Bifet, *Adaptive Stream Mining: Pattern Learning and Mining From Evolving Data Streams* (Frontiers in Artificial Intelligence and Applications), vol. 207. 2010, pp. 1–212.
[4] G. Hulten, L. Spencer, and P. Domingos, "Mining time-changing data streams," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2001, pp. 97–106.
[5] J. Gama, P. P. Rodrigues, E. Spinosa, and A. Carvalho, "Knowledge discovery from data streams," Tech. Rep., 2010, pp. 125–138.
[6] B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom, "Models and issues in data stream systems," in *Proc. 21st ACM SIGMOD-SIGACT-SIGART Symp. Princ. Database Syst. (PODS)*, 2002, pp. 1–16.
[7] P. Domingos and G. Hulten, "Mining high-speed data streams," in *Proc. KDD*, 2000, pp. 71–80.
[8] C. C. Aggarwal, "A survey on stream classification algorithm," in *Data Classification: Algorithms and Applications*. 2014, pp. 245–273.
[9] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proc. 29th Int. Conf. Very Large Data Bases*, 2003, pp. 81–92.
[10] R. Polikar, L. Upda, S. S. Upda, and V. Honavar, "Learn++: An incremental learning algorithm for supervised neural networks," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 31, no. 4, pp. 497–508, Nov. 2001.
[11] R. Polikar, L. Udpa, S. S. Udpa, and V. Honavar, "LEARN++: An incremental learning algorithm for multilayer perceptron networks," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process. (ICASSP)*, vol. 6, Jun. 2000, pp. 3414–3417.
[12] D. Brzezinski and J. Stefanowski, "Reacting to different types of concept drift: The accuracy updated ensemble algorithm," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 25, no. 1, pp. 81–94, Jan. 2014.
[13] M. Scholz and R. Klinkenberg, "An ensemble classifier for drifting concepts," in *Proc. 2nd Int. Workshop Knowl. Discovery Data Streams*, 2005, pp. 53–64.
[14] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Wozniak, "Ensemble learning for data stream analysis: A survey," *Inf. Fusion*, vol. 37, pp. 132–156, Sep. 2017.
[15] J. Gama, I. Žliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, 2014, Art. no. 44.
[16] G. Widmer and M. Kubat, "Learning in the presence of concept drift and hidden contexts," *Mach. Learn.*, vol. 23, no. 1, pp. 69–101, 1996.
[17] P. Laird, "Weighing hypotheses: Incremental from noisy data," *Learning*, pp. 91–98, 1993.
[18] G. Ditzler and R. Polikar, "Incremental learning of concept drift from streaming imbalanced data," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 10, pp. 2283–2301, Oct. 2013.
[19] S. Shalev-shwartz, "Online learning: Theory, algorithms, and applications," Ph.D. dissertation, Jul. 2007.
[20] A. Gepperth and B. Hammer, "Incremental learning algorithms and applications," Tech. Rep., 2016, pp. 27–29.
[21] T. Seipone and J. A. Bullinaria, "Evolving improved incremental learning schemes for neural network systems," in *Proc. IEEE Congr. Evol. Comput.*, vol. 3, Sep. 2005, pp. 2002–2009.
[22] A. Rakhlin, K. Sridharan, and A. Tewari, "Online learning: Beyond regret," in *Proc. COLT*, 2011, pp. 559–594.
[23] E. Hazan and S. Kale, "Beyond the regret minimization barrier: Optimal algorithms for stochastic strongly-convex optimization," *J. Mach. Learn. Res.*, vol. 15, pp. 2489–2512, Jul. 2014.
[24] E. Hazan, "The convex optimization approach to regret minimization," *Optim. Mach. Learn.*, pp. 1–20, 2011.
[25] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2012.
[26] A. Blum and Y. Mansour, "From external to internal regret," in *Proc. Int. Conf. Comput. Learn. Theory*, 2005, pp. 621–636.
[27] F. Chu, Y. Wang, and C. Zaniolo, "An adaptive learning approach for noisy data streams," in *Proc. ICDM*, Nov. 2004, pp. 351–354.
[28] T. G. Dietterich, "Ensemble methods in machine learning," in *Proc. 1st Int. Workshop Multiple Classifier Syst.*, vol. 1857. 2000, pp. 1–15.
[29] R. Polikar, "Ensemble based systems in decision making," *IEEE Circuits Syst. Mag.*, vol. 6, no. 3, pp. 21–45, Sep. 2006.
[30] M. Muhlbaier and R. Polikar, "An ensemble approach for incremental learning in nonstationary environments," in *Multiple Classifier Systems* (Lecture Notes in Computer Science). 2007, pp. 490–500.
[31] L. L. Minku and X. Yao, "DDD: A new ensemble approach for dealing with concept drift," *IEEE Trans. Knowl. Data Eng.*, vol. 24, no. 4, pp. 619–633, Apr. 2012.
[32] W. N. Street and Y. Kim, "A streaming ensemble algorithm (SEA) for large-scale classification," in *Proc. 7th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2001, pp. 377–382.
[33] D. H. Wolpert, "Stacked generalization," *Neural Netw.*, vol. 5, no. 2, pp. 241–259, 1992.
[34] S. Džeroski and B. Ženko, "Is combining classifiers with stacking better than selecting the best one?" *Mach. Learn.*, vol. 54, no. 3, pp. 255–273, 2004.
[35] K. Tumer and J. Ghosh, "Analysis of decision boundaries in linearly combined neural classifiers," *Pattern Recognit.*, vol. 29, no. 2, pp. 341–348, 1996.
[36] L. Breiman, "Stacked regressions," *Mach. Learn.*, vol. 24, no. 1, pp. 49–64, 1996.
[37] A. A. Ghorbani and K. Owrangh, "Stacked generalization in neural networks: Generalization on statistically neutral problems," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, vol. 3, Jul. 2001, pp. 1715–1720.
[38] J. Gama and P. Kosina, "Learning about the learning process," in *Proc. 10th Int. Conf. Adv. Intell. Data Anal.*, 2011, pp. 162–172.
[39] A. Bifet, E. Frank, G. Holmes, and B. Pfahringer, "Accurate ensembles for data streams: Combining restricted Hoeffding trees using stacking," in *Proc. 2nd Asian Conf. Mach. Learn.*, 2010, pp. 1–16.
[40] L. Canzian, Y. Zhang, and M. van der Schaar, "Ensemble of distributed learners for online classification of dynamic data streams," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 1, no. 3, pp. 180–194, Sep. 2015.
[41] I. Frías-Blanco, A. Verdecia-Cabrera, A. Ortiz-Díaz, and A. Carvalho, "Fast adaptive stacking of ensembles," in *Proc. 31st Annu. ACM Symp. Appl. Comput. (SAC)*, 2016, pp. 929–934.
[42] S. Wang, L. Minku, and X. Yao, "Resampling-based ensemble methods for online class imbalance learning," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 5, pp. 1356–1368, May 2015.
[43] Y. Lee, S. C. Madayambath, Y. Liu, D.-T. Lin, R. Chen, and S. S. Bhattacharyya, "Online learning in neural decoding using incremental linear discriminant analysis," in *Proc. IEEE Int. Conf. Cyborg Bionic Syst. (CBS)*, Oct. 2017, pp. 173–177.

[44] H. M. Gomes *et al.*, "Adaptive random forests for evolving data stream classification," *Mach. Learn.*, vol. 106, nos. 9–10, pp. 1469–1495, 2017.
[45] R. Vilalta and Y. Drissi, "A perspective view and survey of meta-learning," *Artif. Intell. Rev.*, vol. 18, no. 2, pp. 77–95, 2002.
[46] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in Python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Oct. 2011.
[47] D. Dheeru and E. K. Taniskidou, "UCI machine learning repository," Tech. Rep., 2017.
[48] K. I. Tsianos and M. G. Rabbat, "Efficient distributed online prediction and stochastic optimization with approximate distributed averaging," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 4, pp. 489–506, Dec. 2016.
[49] Y. Sun, K. Tang, L. L. Minku, S. Wang, and X. Yao, "Online ensemble learning of data streams with gradually evolved classes," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 6, pp. 1532–1545, Jun. 2016.
[50] T. Al-Khateeb *et al.*, "Recurring and novel class detection using class-based ensemble for evolving data stream," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 10, pp. 2752–2764, Oct. 2016.
[51] T. Chen, Q. Ling, and G. B. Giannakis, "An online convex optimization approach to proactive network resource allocation," *IEEE Trans. Signal Process.*, vol. 65, no. 24, pp. 6350–6364, Dec. 2017.
[52] Y.-S. Jeong, Y.-J. Byon, M. Mendonca Castro-Neto, and S. M. Easa, "Supervised weighting-online learning algorithm for short-term traffic flow prediction," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 4, pp. 1700–1707, Dec. 2013.
[53] A. Yassine, S. Singh, and A. Alamri, "Mining human activity patterns from smart home big data for health care applications," *IEEE Access*, vol. 5, pp. 13131–13141, 2017.
[54] S. Shaghaghian and M. Coates, "Online Bayesian inference of diffusion networks," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 3, pp. 500–512, Sep. 2017.
[55] J. Zhu, C. Xu, J. Guan, and D. O. Wu, "Differentially private distributed online algorithms over time-varying directed networks," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 4, no. 1, pp. 4–17, Mar. 2018.
[56] P. Bouboulis, S. Chouvardas, and S. Theodoridis, "Online distributed learning over networks in RKH spaces using random Fourier features," *IEEE Trans. Signal Process.*, vol. 66, no. 7, pp. 1920–1932, Apr. 2018.
[57] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

R. PARI was born in Chennai, India, in 1971. He received the B.E. degree in computer science and engineering from Madras University, India, in 1992, and the M.Tech. degree in computer science and engineering from PRIST University, India, in 2015.

He is currently a Research Scholar with the Department of Computer Science and Engineering, B. S. Abdur Rahman Crescent Institute of Science and Technology, India. He has the industry experience of over 20 years in prominent software services organizations in India. He is a Certified Scrum Master from Scrum Alliance, USA. His research interests include machine learning, artificial intelligence, and software engineering.

M. SANDHYA was born in Chennai, India. She received the B.E. degree in computer science and engineering from Madras University, India, in 1998, the M.B.A, degree from Annamalai University, India, in 2000, the M.E. degree from Madras University in 2002, and the Ph.D. degree in computer science and engineering from Anna University, India, in 2012.

She is currently a Professor and the Head of the Department of Computer Science and Engineering, B. S. Abdur Rahman Crescent Institute of Science and Technology, India. She contributed to the research and education in computer science and engineering, wireless networks, cloud computing, and big data analytics. She is a Life Member of the Indian Society for Technical Education. She is a reviewer of many international journals.

SHARMILA SANKAR was born in Chennai, India. She received the B.E. degree in computer science and engineering from Bharathiyar University, India, in 1993, the M.S. degree from BITS, Pilani, India, the M.E. degree from Madras University, India, and the Ph.D. degree in computer science and engineering from Anna University, India, in 2012.

She is currently a Professor of computer science and engineering with the B. S. Abdur Rahman Crescent Institute of Science and Technology, India. She contributed to the research and education in computer science and engineering, Internet of Things, wireless networks, and big data analytics. She is a member of the Association for Computing Machinery and the Computer Society of India. She is a reviewer of many international journals.