IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Revue d'Intelligence Artificielle
Vol. 37, No. 5, October, 2023, pp. 1301-1310
Journal homepage: http://iieta.org/journals/ria

Build a Lightweight Dataset and Concept Drift Detection Method for Evolving Time Series Data Streams

Nitin B. Ghatage*, Pramod D. Patil

Computer Engineering, Dr. D Y Patil Institute of Technology, Pimpri, Pune 411018, India

Corresponding Author Email: nitinbaghatage@gmail.com

https://doi.org/10.18280/ria.370523

Received: 2 August 2023
Revised: 2 October 2023
Accepted: 9 October 2023
Available online: 31 October 2023

Keywords:
timeseries, lightweight, recurrent neural networks, concept drift

ABSTRACT

Time series forecasting, a potent tool for predicting real-world entities such as financial markets and weather patterns, often grapples with the issue of concept drift, characterized by changes in the behaviour of the time series over time. This study aims to develop a lightweight time series model, efficient in training time, to match the data stream's arrival rate. Furthermore, a method to detect the presence of concept drift in the data stream, regardless of the time point, is discussed. Presented herein is a benchmark dataset, publicly accessible and specifically designed to simulate changing time series scenarios across diverse industries including Energy, Air Quality, and Pollution. This dataset amalgamates synthetic and actual time series along with ground truth concept drift locations, facilitating a comprehensive evaluation of concept drift detection techniques. A novel, lightweight concept drift detection method, which integrates supervised methodologies with statistical metrics to surmount the resource constraints often encountered in streaming data scenarios, is proposed. This method minimizes computational overhead while ensuring reliable drift detection in response to shifting data distributions. Experimental results indicate that the proposed approach surpasses prior methods in computational performance whilst accurately identifying idea drifts in evolving time series data streams. The study contributes a valuable dataset and a lightweight feature selection method, advancing the knowledge in the field of concept drift detection within the context of time series data streams. These advancements provide an efficient technique for tracking changing data patterns across various application domains, thus offering significant implications for future research.

1. INTRODUCTION

Time series is a sequence of data measured with a constant frequency. Forecasting of time series is a very important task in today's world with several applications starting from financial markets to weather prediction etc. Univariate time series is a time series having information about only one real world entity, however often we see any real world entity is dependent on multiple other features, which leads us to multivariate time series Multivariate time series forecasting can be challenging due to high volume of the data. However, one of the major issues with time series forecasting is the possible presence of concept drift Concept drift is the change in the data pattern in such a way that a model trained on old data cannot produce good prediction anymore. Thus, it's necessary to tackle these issues while building a time series prediction model. Online models are suitable to handle huge stream of data as they do not require to store all the past data. Recurrent neural network based models are popular for time series forecasting as they can capture complex pattern in the data. It has been proven over several review studies like [1] that deep learning based RNN moels tend to perform better than conventional models for forecasting problems. Although Several RNN based models [2-4] and LSTM-CNN hybrid models [5] have been developed for multivariate time series forecasting addressing issues but there are few such models that can handle both concept drift and well as high volume of data in efficient manner. Distributed machine learning techniques [6] are also used for time series forecasting but they have high hardware requirements. Predictive tool like MARS (Multivariate Adaptive Regression Splines) have been used for forecasting problems such as transportation energy demand [7] but their application is limited. In this document we discuss about feature selection as a way to effectively reduce the computation cost of the model and make it lightweight in nature and we also provide a way to detect and adapt to possible concept drift in the data stream for an online RNN model using a sliding window approach.

To increase the reliability and accuracy of predictions, combine several lightweight models. To capitalise on the advantages of many models, strategies such as model stacking, bagging, or boosting might be used. Time Series Decomposition use techniques like Seasonal Decomposition of Time Series (STL) or Empirical Mode Decomposition (EMD) to break down time series data into its individual components, such as trend, seasonality, and residuals. This division can make modelling simpler and predicting more precise.

Use specialised feature selection techniques, such as Recursive Feature Elimination (RFE), L1 Regularisation (Lasso), or Mutual Information-based feature selection, to determine which characteristics are the most instructive and pertinent for prediction.

Use specialised stream mining techniques to handle changing data streams effectively, such as Hoeffding Trees, VFDT (Very Fast Decision Tree), or SAMOA (Scalable

Advanced Massive Online Analysis).

Include change detection techniques like CUSUM (Cumulative Sum) or Page-Hinkley to identify concept drifts and activate model adaptation when significant alterations in the distribution of the data take place.

To divide the computational load and enable effective real-time processing of time series data streams, use parallel processing frameworks like Apache Flink or Apache Kafka Streams.

Reduce the dimensionality and memory needs of time series data while maintaining its key properties by using data quantization and compression techniques.

Hyper parameter Tuning, to improve performance while preserving efficiency, fine-tune the hyper parameters of lightweight models using methods such as Bayesian optimisation, grid search, or random search.

Online Anomaly Detection: Time series forecasting and online anomaly detection algorithms can be combined to quickly spot outliers or strange trends, which is essential for many applications including fraud detection and fault monitoring.

Making predictions about future data points based on prior time-ordered data is an important component of data analysis and predictive modelling. This method is used in a variety of fields, including finance, economics, climate research, retail, healthcare, and others. Time series data is useful for identifying trends and patterns as well as for making decisions because it is comprised of observations that are collected or recorded over time. Key Challenges in Time Series is given as:

- Data Quality: Measurement mistakes, missing values, outliers, and poor data quality can all have a detrimental effect on forecasting accuracy. In order to address these difficulties, data pre-treatment and purification are crucial tasks.
- Overfitting: When a model is overly complicated and attempts to match the underlying patterns of the data, it instead tries to fit the data noise. It's crucial to strike a balance between model complexity and overfitting.
- Concept Drift: Concept drift occurs when data patterns in emerging systems alter over time. It is a huge task to adjust to these changes and update predicting models accordingly.
- Scalability: Effective techniques and processing resources are needed to handle large-scale time series data streams in real-time or near-real-time applications.

Motivation:

A simple model for time series forecasting in changing data streams is inspired by a number of real-world issues and new trends.

Many contemporary applications, including financial market research, industrial equipment monitoring, and energy consumption forecasting, need for real-time or almost real-time forecasts using streaming time series data. Lightweight models are appropriate for many applications because they can make predictions more quickly. There could not be enough processing power or memory in situations with limited resources, such as edge computing devices or IoT sensors. A lightweight model is necessary for the effective use of the available resources. Low latency is essential in some applications. For rapid decision-making in applications like autonomous vehicles and predictive maintenance, a lightweight model can lower prediction latency.

Energy-efficient models can increase battery life and lower energy consumption in battery-operated or energy-sensitive devices like smartphones and Internet of Things sensors. In changing time series data streams, lightweight models are frequently more nimble and capable of idea drift adaptation. They effectively adapt to shifting patterns, lowering the danger of model obsolescence. In many fields where real-time or resource-constrained forecasting is advantageous, such as healthcare, finance, environmental monitoring, and others, lightweight models can find use.

The motivation for a lightweight model for time series forecasting in evolving data streams is driven by the need for efficient, scalable, and adaptable solutions that can meet the demands of modern data-intensive applications while operating within resource constraints. These models aim to strike a balance between prediction accuracy and computational efficiency, making them highly relevant in the evolving landscape of data analytics.

The rest of the paper is organised as follows, first we give theoretical explanation on the lightweight feature selection technique used, followed by explaining how our concept drift detection approach works. Then we provide the real world datasets used and their descriptions followed by results and analysis of the results collected on those datasets. Finally, we briefly summarize the effectiveness of our proposed model and how it can be further improved in the future.

2. LIGHTWEIGHT METHOD

A lightweight model takes less time to run compared to other traditional models. Making a model lightweight in beneficial while dealing with data streams since the amount of data is potentially massive and it becomes very expensive to store the whole data in data storage. In literature there are some techniques that can make a model lightweight in nature. Building a model on a representative sample of data rather than entire data can be done, however it's not very intuitive for time series data since the order of the data points is important in that case and so sampling data points randomly from the data will break the order of the data points that. Feature selection is suggested in the study [2] as another way where we only consider the important features from a dataset and discard the rest of the features while building the model. This way the processing time of the model can be improved. There are many feature selection methodologies in machine learning for example, random forest based, IG (Information Gain) value based as shown to be applied with ANN [8] etc. Segmentation [9] of total data into multiple segments and then building one model or an ensemble is another approach that can be used for faster training of the model. Also, Stratification [10] of the time series into several smaller homogeneous time series can be used as suggested to reduce gradient estimation time for model training.

For our model we used Random Forest based feature selection technique to make the model lightweight as shown in Figure 1.

Figure 1. Light weight model architecture

Random Forest based Feature selection: As shown in Figure 2, Random Forest is a tree based ensemble learning algorithm. In ensemble learning the prediction is the average of all the prediction from all the individual models.

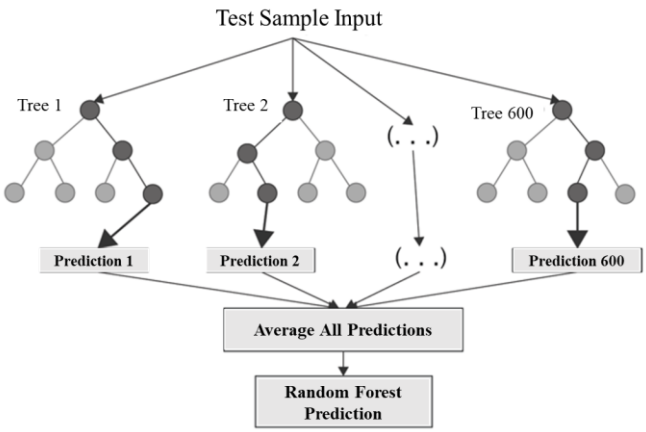


Figure 2. Random forest algorithm structure

Lightweight Feature selection Algorithm:

Let:

D be the dataset.

F be the set of features.

T be the number of decision trees in the forest.

GI(*X*) be the Gini Importance of feature '*X*'.

For each decision tree t in the forest:

Randomly select a subset of data samples D_t from *D*.

Randomly select a subset of features F_t from *F*.

Build a decision tree using D_t and F_t.

Calculate the Gini Importance for feature '*X*' in a tree *t*:

GI(X)_t =∑[reduction in Gini impurity caused by feature '*X*' at each node in tree t]

Calculate the Gini Importance for feature '*X*'

GI(X)=(1/T) * ∑[GI(X)_t for each tree *t*]

A random forest regressor is used for regression tasks. The random forest algorithm can be used to compute feature importance.

• Random Forest contains several decision trees, each built from randomly selected instances from the data and also randomly selected subset of features.

• Each tree contains several nodes where the data is split into two categories based on the feature values.

• Each of the category contains observations that are more similar among themselves and different from the ones in another category.

• The importance of the features is decided based on how impactful it is while splitting the data. For regression it is based on reduction of impurity or variance in each category.

• The impurity or variance reduction from each feature while splitting based on it can be averaged across trees to determine the final importance of the variable.

• By default, it selects the features with higher feature importance than the average feature importance across all the features.

The impurity is computed using Mean Squared Error (MSE), MSE is computed as follows:

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(y_i - \mu)^2 \quad (1)$$

where, y_i is the value of target, N denotes the number of instances and μ denotes the mean value of the target over all the instances.

This variance reduction is computed over all the nodes of the trees that are part of the Random Forest. The importance of the nodes is computed as follows:

$$Imp_{j=W_jC_j-W_{lj}C_{lj}-W_{rj}C_{rj}}$$

where, Imp_j stands for importance of node j. W_j denotes the weighted number of samples that belongs to node j, C_j denotes the impurity at node j. l_j and r_j subscripts stand for left and right child after the split at node j respectively.

The importance of each feature is then calculated as follows:

$$FI_i = \frac{\sum_{j:node\ j\ splits\ on\ feature\ i} Imp_j}{\sum_{k\in all\ nodes} Imp_k} \quad (2)$$

where, *El* stands for feature importance of ith feature and *Imp*, denotes the importance of node *j*. The feature importance is normalized in the range of 0 to 1 by dividing the feature importance values by the sum of all feature's importance.

$$\text{Norm } FI_i = \frac{FI_i}{\sum_{j\in all\ features} FI_j} \quad (3)$$

The final feature importance computed over the entire random forest is the mean of feature importance over all the trees in the forest:

$$RF\_FI_i = \frac{\sum_{j\in all\ trees} NormFI_j}{T} \quad (4)$$

where, *RF FI* is the feature importance of ith feature for the entire random forest, *NormEl* is the normalized feature importance at the ith tree and *T* denotes the total count of decision trees in the forest. Table 1 shows number of features selected using Random Forest for Engery, Air Quality and Air Pollution dataset.

Table 1. Summary of random forest based feature selection algorithm lightweight model

Dataset	No of Feature	No of Record	No of Feature Selected using Random Forest
Energy	29	19735	13
Air quality	15	9358	3
Air Pollution	9	43801	8

3. DETECTION OF CONCEPT DRIFT

Concept drift is known as the change in the relationship between the target variable and independent variables in the data. It can severely affect a model performance. It is true that there exist forecasting methods that do not perform explicit drift detection, for example one of such method simulates drift by adding noise [11] into the training data to make it more responsive to actual concept drift. Some methods do continuous adaption [12] without explicit detection of drift. However; there are limitations to such methods, adapting to changing conditions in time series data without effective drift

detection can lead to inaccurate predictions and suboptimal model performance. Hence its necessary to detect concept drift accurately while dealing with streaming data. The paper [13] discusses existing concept drift detection approaches in the literature. Some of the existing methods to detect concept drift includes sliding window techniques like OASW [14], ADWIN [15]. In OASW they measure the performance of the model over a recent sliding window of size t (that holds t data points) and compare it with previous model performance over the data points in the sliding window at t time in the past. If the difference is more than a threshold, concept drift is detected. In ADWIN two window is used, one fixed window holds past data and another sliding window holds recent data. The model performance is measure over both the windows and the difference are checked against a threshold. If threshold is crossed concept drift is detected. ADWIN is utilized in many forecasting methods such as load forecasting using deep learning with smart meter [16] data. Adaptive SVR [17] also follows the same principle of error threshold-based triggering of adaption. Many conventional idea drift detection techniques, such as statistical or window-based tests, call for the processing and storage of sizable historical data windows. In situations where resources are scarce or when dealing with enormous data streams, this substantial computing overhead becomes prohibitive. Some techniques may not work well in situations where different types of notion drift coexist, such as those involving slow or abrupt drift. High-velocity, high-volume data streams are difficult to handle with traditional batch learning techniques. The requirement to retrain models on a regular basis might cause delays and ineffective responses to idea drift. Some other techniques to handle concept drift include matching new data with old, segmented data [9] and continuous adaptation [12] without explicit detection of drift.

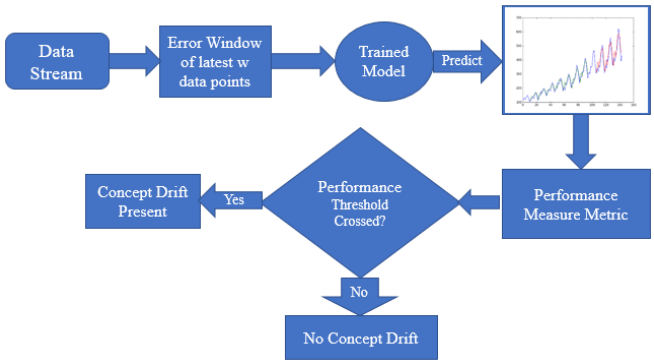


Figure 3. Concept drift detection method

As shown in Figure 3, Our Model detects concept drift by monitoring the model performance over a window time frame. The size of the time frame window is fixed say w, so the model's performance metric (accuracy for classification, RMSE or MAE or MAPE for regression) is tracked over the last w predictions. A threshold Th is defined beforehand which indicates the critical values for the performance metric used. The value Th is chosen based on the performance of the model on the training data itself, it this way we can use one sliding window only compared to other methods with relative comparison of performance across two windows. When the average of the performance metric over the last w prediction across Th, we say that the underlying concept of the data has changed significantly and the model can no longer proves satisfactory predictions thus Concept Drift has occurred.

So, assuming using RMSE as the error metric the concept drift detection algorithm is as follows:

Concept Drift Detection (ground_truth, pred, W, Th):

Ground_truth: actual target values
Pred: model predicted values
W: Latest data window of size w
Th: Error threshold based on Training RMSE

START
Current_error: RMSE over the latest window W, $\sqrt{\frac{1}{w}\sum_{i=1}^{W}(Pred_i - ground\_truth_i)^2}$
If (Current_error > Th);

Concept drift detected;
Else:
  Get next prediction;
  Get next ground_truth;
  Update window W;
  Call Concept Drift Detection function again;
END

The proposed lightweight concept drift detection method for evolving time series data streams' time and space complexity may be analysed as follows:

Time Complexity:

• Measures of statistical significance: The computation of statistical measures, such as density estimates or distance metrics, is often linear in the number of data points and the number of features, resulting in a time complexity of O(N * M), where M is the number of features.

• Adaptation and Detection: In most circumstances, concept drift adaptation and detection are carried out progressively as new data come in, resulting in a temporal complexity of O(1) per data point.

Space Complexity:

• Statistical Measures: The memory required to store intermediate outcomes, such as distance matrices or density estimations, is often the cause of a statistical measure's spatial complexity. In terms of the quantity of features and data points, it is often linear, or O(N*M).

• Model Parameters: The memory needed to store any model parameters that the method utilises (such as those for statistical tests) is included in the space complexity.

4. DATASET DESCRIPTION

We used three real world time series datasets that are publicly available at the UCI Machine Learning repository and Kaggle.

4.1 Appliances energy prediction

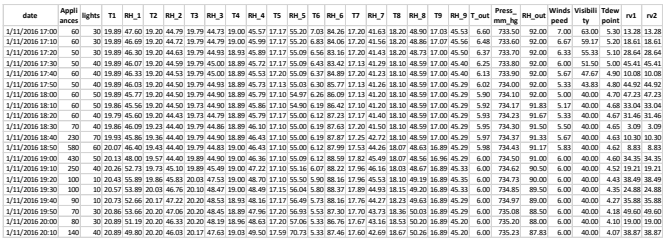
Table 2. Energy data description

Dataset	No of Feature	No of Record
Energy Dataset	29	19735

Appliances energy prediction dataset has data for 4.5 months with 10 minutes frequency. It has 19735 data points

and 29 features in it. The details of the features and a sample of data is given as shown in Table 2.

Sample data (Appliances energy prediction):



date	Appliances	lights	T1	RH_1	T2	RH_2	T3	RH_3	T4	RH_4	T5	RH_5	T6	RH_6	T7	RH_7	T8	RH_8	T9	RH_9	T_out	Press_mm_hg	RH_out	Windspeed	Visibility	Tdewpoint	rv1	rv2
1/11/2016 17:00	60	30	19.89	47.60	19.20	44.79	19.79	44.73	19.00	45.57	17.17	55.20	7.03	84.26	17.20	41.63	18.20	48.90	17.03	45.53	6.60	733.50	92.00	7.00	63.00	5.30	13.28	13.28
1/11/2016 17:10	60	30	19.89	46.69	19.20	44.72	19.79	44.79	19.00	45.99	17.17	55.20	6.83	84.06	17.20	41.56	18.20	48.86	17.07	45.56	6.48	733.60	92.00	6.67	59.17	5.20	18.61	18.61
1/11/2016 17:20	50	30	19.89	46.30	19.20	44.63	19.79	44.93	18.93	45.89	17.17	55.09	6.56	83.16	17.20	41.43	18.20	48.73	17.00	45.50	6.37	733.70	92.00	6.33	55.33	5.10	28.64	28.64
1/11/2016 17:30	50	40	19.89	46.07	19.20	44.59	19.79	45.00	18.89	45.72	17.17	55.09	6.43	83.42	17.13	41.29	18.10	48.59	17.00	45.40	6.25	733.80	92.00	6.00	51.50	5.00	45.41	45.41
1/11/2016 17:40	60	40	19.89	46.33	19.20	44.53	19.79	45.00	18.89	45.53	17.20	55.09	6.37	84.89	17.20	41.23	18.10	48.59	17.00	45.40	6.13	733.90	92.00	5.67	47.67	4.90	10.08	10.08
1/11/2016 17:50	50	40	19.89	46.03	19.20	44.50	19.79	44.93	18.89	45.73	17.13	55.03	6.30	85.77	17.13	41.26	18.10	48.59	17.00	45.29	6.02	734.00	92.00	5.33	43.83	4.80	44.92	44.92
1/11/2016 18:00	60	50	19.89	45.77	19.20	44.50	19.79	44.90	18.89	45.79	17.10	54.97	6.26	86.09	17.13	41.20	18.10	48.59	17.00	45.29	5.90	734.10	92.00	5.00	40.00	4.70	47.23	47.23
1/11/2016 18:10	60	50	19.86	45.56	19.20	44.50	19.73	44.90	18.89	45.86	17.10	54.90	6.19	86.42	17.10	41.20	18.10	48.59	17.00	45.29	5.92	734.17	91.83	5.17	40.00	4.68	33.04	33.04
1/11/2016 18:20	60	40	19.79	45.60	19.20	44.43	19.73	44.79	18.89	45.79	17.17	55.00	6.12	87.23	17.17	41.40	18.10	48.59	17.00	45.29	5.93	734.23	91.67	5.33	40.00	4.67	31.46	31.46
1/11/2016 18:30	70	40	19.86	46.09	19.23	44.40	19.79	44.86	18.89	46.10	17.10	55.00	6.19	87.63	17.20	41.50	18.10	48.59	17.00	45.29	5.95	734.30	91.50	5.50	40.00	4.65	3.09	3.09
1/11/2016 18:40	230	70	19.93	45.86	19.36	44.40	19.79	44.90	18.89	46.43	17.10	55.00	6.19	87.87	17.25	42.72	18.10	48.59	17.00	45.29	5.97	734.37	91.33	5.67	40.00	4.63	10.30	10.30
1/11/2016 18:50	580	60	20.07	46.40	19.43	44.40	19.79	44.83	19.00	46.43	17.10	55.00	6.12	87.99	17.53	44.26	18.07	48.63	16.89	45.29	5.98	734.43	91.17	5.83	40.00	4.62	8.83	8.83
1/11/2016 19:00	430	50	20.13	48.00	19.57	44.40	19.89	44.90	19.00	46.36	17.10	55.09	6.12	88.59	17.82	45.49	18.07	48.56	16.96	45.29	6.00	734.50	91.00	6.00	40.00	4.60	34.35	34.35
1/11/2016 19:10	250	40	20.26	52.73	19.73	45.10	19.89	45.49	19.00	47.22	17.10	55.16	6.07	88.22	17.96	46.16	18.03	48.67	16.89	45.33	6.00	734.62	90.50	6.00	40.00	4.52	19.21	19.21
1/11/2016 19:20	100	10	20.43	55.89	19.86	45.83	20.03	47.53	19.00	48.70	17.10	55.50	5.90	88.16	17.96	45.53	18.10	49.19	16.89	45.35	6.00	734.73	90.00	6.00	40.00	4.43	38.49	38.49
1/11/2016 19:30	100	10	20.57	53.89	20.03	46.76	20.10	48.47	19.00	48.49	17.15	56.04	5.80	88.37	17.89	44.93	18.15	49.20	16.89	45.33	6.00	734.85	89.50	6.00	40.00	4.35	24.88	24.88
1/11/2016 19:40	90	10	20.73	52.66	20.17	47.22	20.20	48.53	18.93	48.16	17.17	56.49	5.73	88.16	17.76	44.27	18.23	49.63	16.89	45.29	6.00	734.97	89.00	6.00	40.00	4.27	35.88	35.88
1/11/2016 19:50	70	30	20.86	53.66	20.20	47.06	20.20	48.45	18.89	47.96	17.20	56.93	5.53	87.30	17.70	43.73	18.36	50.03	16.89	45.29	6.00	735.08	88.50	6.00	40.00	4.18	49.60	49.60
1/11/2016 20:00	80	30	20.89	51.19	20.20	46.33	20.20	48.19	18.96	48.63	17.20	57.06	5.33	86.76	17.67	43.16	18.53	50.20	16.89	45.20	6.00	735.20	88.00	6.00	40.00	4.10	19.00	19.00
1/11/2016 20:10	140	40	20.89	49.80	20.20	46.03	20.17	47.63	19.03	49.50	17.59	70.73	5.33	87.46	17.60	42.69	18.67	50.26	16.89	45.20	6.00	735.23	87.83	6.00	40.00	4.07	38.87	38.87

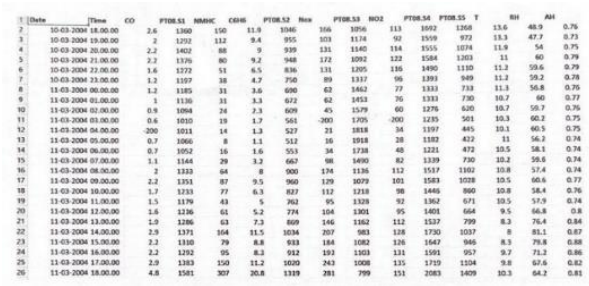
4.2 Air quality prediction

Air quality dataset has 9358 instances and 15 features. Data spans for one year starting March 2004 and ending in Feb 2005 as shown in Table 3. The missing values in the dataset are indicated with -200 value.

Table 3. Air quality data description

Dataset	No of Feature	No of Record
Air quality Dataset	15	9358

Sample data (Air quality data description**):**



	Date	Time	CO	PT08.S1	NMHC	C6H6	PT08.S2	Nox	PT08.S3	NO2	PT08.S4	PT08.S5	T	RH	AH
2	10-03-2004	18.00.00	2,6	1360	150	11,9	1046	166	1056	113	1692	1268	13,6	48,9	0,76
3	10-03-2004	19.00.00	2	1292	112	9,4	955	103	1174	92	1559	972	13,3	47,7	0,73
4	10-03-2004	20.00.00	2,2	1402	88	9	939	131	1140	114	1555	1074	11,9	54	0,75
5	10-03-2004	21.00.00	2,2	1376	80	9,2	948	172	1092	122	1584	1203	11	60	0,79
6	10-03-2004	22.00.00	1,6	1272	51	6,5	836	131	1205	116	1490	1110	11,2	59,6	0,79
7	10-03-2004	23.00.00	1,2	1197	38	4,7	750	89	1337	96	1393	949	11,2	59,2	0,78
8	11-03-2004	00.00.00	1,2	1185	31	3,6	690	62	1462	77	1333	733	11,3	56,8	0,76
9	11-03-2004	01.00.00	1	1136	31	3,3	672	62	1453	76	1333	730	10,7	60	0,77
10	11-03-2004	02.00.00	0,9	1094	24	2,3	609	45	1579	60	1276	620	10,7	59,7	0,76
11	11-03-2004	03.00.00	0,6	1010	19	1,7	561	-200	1705	-200	1235	501	10,3	60,2	0,75
12	11-03-2004	04.00.00	-200	1011	14	1,3	527	21	1818	34	1197	445	10,1	60,5	0,75
13	11-03-2004	05.00.00	0,7	1066	8	1,1	512	16	1918	28	1182	422	11	56,2	0,74
14	11-03-2004	06.00.00	0,7	1052	16	1,6	553	34	1738	48	1221	472	10,5	58,1	0,74
15	11-03-2004	07.00.00	1,1	1144	29	3,2	667	98	1490	82	1339	730	10,2	59,6	0,74
16	11-03-2004	08.00.00	2	1333	64	8	900	174	1136	112	1517	1102	10,8	57,4	0,74
17	11-03-2004	09.00.00	2,2	1351	87	9,5	960	129	1079	101	1583	1028	10,5	60,6	0,77
18	11-03-2004	10.00.00	1,7	1233	77	6,3	827	112	1218	98	1446	860	10,8	58,4	0,76
19	11-03-2004	11.00.00	1,5	1179	43	5	762	95	1328	92	1362	671	10,5	57,9	0,74
20	11-03-2004	12.00.00	1,6	1236	61	5,2	774	104	1301	95	1401	664	9,5	66,8	0,8
21	11-03-2004	13.00.00	1,9	1286	63	7,3	869	146	1162	112	1537	799	8,3	76,4	0,84
22	11-03-2004	14.00.00	2,9	1371	164	11,5	1034	207	983	128	1730	1037	8	81,1	0,87
23	11-03-2004	15.00.00	2,2	1310	79	8,8	933	184	1082	126	1647	946	8,3	79,8	0,88
24	11-03-2004	16.00.00	2,2	1292	95	8,3	932	193	1103	131	1591	957	9,7	71,2	0,86
25	11-03-2004	17.00.00	2,9	1383	150	11,2	1020	243	1008	135	1719	1104	9,8	67,6	0,82
26	11-03-2004	18.00.00	4,8	1581	307	20,8	1319	281	799	151	2083	1409	10,3	64,2	0,81

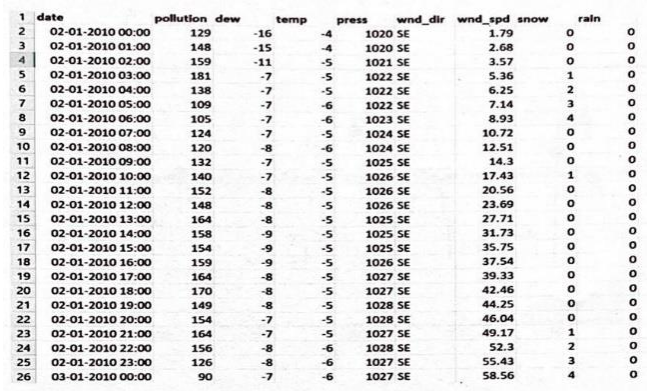
4.3 Air pollution prediction

The air pollution data has 43801 instances and 9 features as shown in Table 4. It has 5 years of data collected at an hourly basis at the US embassy in Beijing, China.

Table 4. Pollution data description

Dataset	No of Feature	No of Record
Air Pollution	9	43801

Sample data (Air pollution prediction):



	date	pollution	dew	temp	press	wnd_dir	wnd_spd	snow	rain
2	02-01-2010 00:00	129	-16	-4	1020	SE	1.79	0	0
3	02-01-2010 01:00	148	-15	-4	1020	SE	2.68	0	0
4	02-01-2010 02:00	159	-11	-5	1021	SE	3.57	0	0
5	02-01-2010 03:00	181	-7	-5	1022	SE	5.36	1	0
6	02-01-2010 04:00	138	-7	-5	1022	SE	6.25	2	0
7	02-01-2010 05:00	109	-7	-6	1022	SE	7.14	3	0
8	02-01-2010 06:00	105	-7	-6	1023	SE	8.93	4	0
9	02-01-2010 07:00	124	-7	-5	1024	SE	10.72	0	0
10	02-01-2010 08:00	120	-8	-6	1024	SE	12.51	0	0
11	02-01-2010 09:00	132	-7	-5	1025	SE	14.3	0	0
12	02-01-2010 10:00	140	-7	-5	1026	SE	17.43	1	0
13	02-01-2010 11:00	152	-8	-5	1026	SE	20.56	0	0
14	02-01-2010 12:00	148	-8	-5	1026	SE	23.69	0	0
15	02-01-2010 13:00	164	-8	-5	1025	SE	27.71	0	0
16	02-01-2010 14:00	158	-9	-5	1025	SE	31.73	0	0
17	02-01-2010 15:00	154	-9	-5	1025	SE	35.75	0	0
18	02-01-2010 16:00	159	-9	-5	1026	SE	37.54	0	0
19	02-01-2010 17:00	164	-8	-5	1027	SE	39.33	0	0
20	02-01-2010 18:00	170	-8	-5	1027	SE	42.46	0	0
21	02-01-2010 19:00	149	-8	-5	1028	SE	44.25	0	0
22	02-01-2010 20:00	154	-7	-5	1028	SE	46.04	0	0
23	02-01-2010 21:00	164	-7	-5	1027	SE	49.17	1	0
24	02-01-2010 22:00	156	-8	-6	1028	SE	52.3	2	0
25	02-01-2010 23:00	126	-8	-6	1027	SE	55.43	3	0
26	03-01-2010 00:00	90	-7	-6	1027	SE	58.56	4	0

5. RESULTS AND ANALYSIS

In order to analysis our proposed model and showcase its results we used the three real world datasets selected ie, Energy, Air quality and Pollution. The result section is structured as follows, first we provide the model specifications such as hyper-parameters for each dataset along with the pre-processing done. Next to showcase the Lightweight Nature of our proposed model we analyse the memory consumption and time consumption of our model with feature selection and without feature selection. Next, we compare the RMSE of predictions on test data for our model both with and without feature selection. After that we analyse our concept drift detection method and show case the outcomes on both the datasets.

Table 5. Result of random forest based feature selection

Dataset Name	Total Number of Features (Excluding Date and Time)	Number of Features Selected	Selected Features
Energy	28	13	'Appliances', 'lights', 'RH_1', 'RH_2', 'T3', 'RH_3', 'RH_5', 'RH_6', 'RH_7', 'T8', 'RH_8', 'Press_mm_hg', 'RH_out'
Air quality	13	3	'CO', 'PT08.S2', 'NOX'
Pollution	8	5	'Pollution', 'Dew', 'Temp', 'Press', 'Wind_spd'

5.1 Pre-processing and feature selection of data

At the starting step of we set the date and time as index for visualization purpose. Since Air Quality dataset has some missing values, we imputed the missing values using the previous valid data value. In case of Pollution data, we used Label Encoding to transform categorical feature to numerical feature.

We applied the Random Forest based Feature Selection algorithm to both the datasets to select only the important features and the selection were as shown in Table 5.

Based on the selected features we performed outlier detection using Z score value, Z score denotes the distribution of each individual data points with respect to the mean of the population in terms of standard deviation. For example a Z score of 2 denotes that the selected data point for selected feature lies within 2 standard deviations from the mean of the population of that feature. We used Z score to filter out data points with extreme outliers. |Z score| greater than 3 means those data points are likely to be outliers and greater than 4 is definitely outliers. We exclude all the data points that contains outliers for any feature except target feature from the dataset before proceeding with model building.

We applied the selected features on our RNN model and computed the RMSE (Root Mean Squared Error) and compared it with a recent work named AIS-RNN [2] model.

5.2 Train-test split

For Energy and Air Quality datasets we used 80% of the data for initial training, 10% of that 80% were used as a validation set while training. Rest 20% of the data were used as a test set in a streaming manner feeding one point at a time to the trained model. For Pollution data our main objective is to show how the model detects and adapts to different concept drifts hence we used 30% of the data as training set (10% of that used as validation set), and 70% of the data as test set. Since our problem is regression one, we use RMSE (Root Mean Square Error) as an error metric and measure the final RMSE on the test set.

5.3 Model specifications

The RNN model used for all the datasets have 2 hidden layers and each layer had 12 units. The lag parameter was chosen to be 12. The optimizer was Adam with learning rate of 0.001 for the training. We used sigmoid activation for hidden layers and linear activation for output layer of the network. The batch size was taken as 1500 and we ran the model for 5000 epochs. We also used L1 regularization in the hidden layer of the value 0.0001. The hyper-parameters were chosen using HyperOpt Bayesian hyper-parameter tuning using a range of values for those hyper-parameters.

5.4 Memory and time consumption analysis

A lightweight model takes less memory and less time to process the data compared to the conventional model. To validate the effectiveness of our model being lightweight we measured both the memory consumed and time consumed while training by the data with and without feature selection. To check scalability, we artificially generated a large version of the Energy and Air Quality datasets with 120000 instances each to measure the memory and time consumption as well. All the experiments were conducted on a machine with 11th Gen Intel® core ™ i5-processor with clock speed 2.60GHz, 16GB of RAM and 4 Core(s), The results are as shown in Table 6 and Table 7:

Table 6. Memory consumption by the datasets without feature selection and with feature selection

Dataset Name	Dataset Size Before Feature Selection (MB)	Dataset Size after Feature Selection (MB)	Percentage of Reduction in Size
Energy	4.4	2.1	52.27%
Air Quality	1	0.29	71.39%
Energy Generated	26.6	12.8	51.88%
Air Quality Generated	12.8	3.7	71.09%
Pollution	3	2	33.33%

Table 7. Time consumption for training by the datasets without feature selection and with feature selection

Dataset Name	Time without Feature Selection	Time with Feature Selection	Percentage Reduction of time
Energy	10 min 7 sec	9 min 19 sec	7.91%
Air quality	4 min 53 sec	4 min 30 sec	7.84%
Energy Generated	1 hour 10 min	51 min 41 sec	26.20%
Air Quality Generated	52 min 25 sec	41 min 40 sec	20.51%
Pollution	18 min 15 sec	16 min 41 sec	8.58%

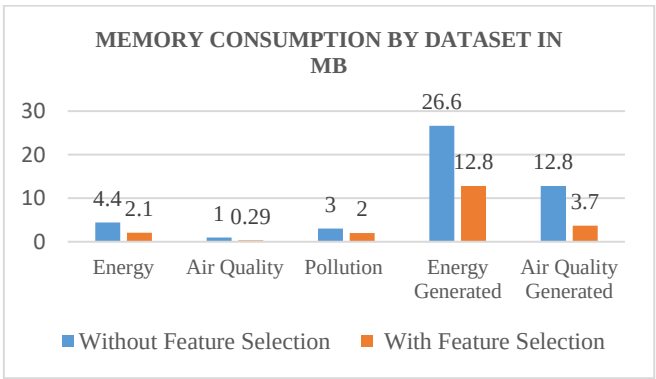


Figure 4. Memory consumption in MB without feature selection and with feature selection

From the results as shown in Figure 4, we see that by using Random Forest based feature selection we can reduce the memory consumption by about 52% for Energy data, 71% for Air Quality data and 33.33% for Pollution data. This can save a lot of storage when the data size grows as large volume is a well-known characteristic of real world time series data. The Time consumption to train the RNN model is also seen to be reducing since the model needs to process a smaller number of features. Comparing the reduction of time consumption between the original data and the larger generated version of the data, we observe that the reduction of time scales well as the data size grows, hence while its only about 8% reduction for original data size, the reduction is significantly more about 26% for Energy and 20% for Air Quality generated data. So, we can conclude that as the data size grows even more the reduction of time consumed will be more significant as shown in Figure 5.

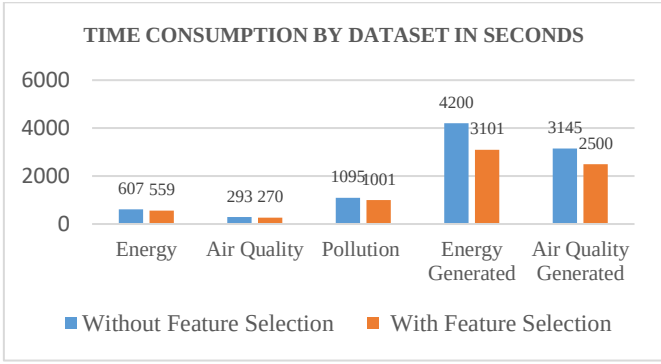


Figure 5. Time consumption in sec without feature selection and with feature selection

5.5 Model performance with and without feature selection analysis

We measured the performance of the RNN model using same set of hyper-parameters for the 3 datasets on their test sets once with full feature and once with only selected features from the Random Forest based feature selection procedure. The results are as shown in Table 8:

Table 8. Model performance on test datasets without feature selection and with feature selection

Dataset Name	RMSE without Feature Selection	RMSE with Feature Selection
Energy	61.34	58.74
Air quality	0.604	0.606
Pollution	28.15	27.79

As we see for Energy and Pollution data the performance is better when we used feature selection using Random Forest to select only the important feature. Whereas for Air Quality dataset the performance is almost same in both the cases.

Hence, we see that by using Random Forest based feature selection procedure we are not only saving memory and time consumption by the model training but the model performance itself is also improving or remaining similar to what it was while using all the features.

6. CONCEPT DRIFT DETECTION ANALYSIS

Our model detects any possible drift in the data stream by comparing the model performance over a recent window of data against a threshold. The threshold is set using the model performance on the training set itself. For all the datasets the performance metric used is Root Mean Squared Error (RMSE). The training RMSE as measured with the specified model specifications are as shown in Table 9:

Table 9. Training RMSE on the datasets

Dataset Name	Training RMSE
Energy	68.03
Air quality	0.615
Pollution	29.29

One important note is that RMSE depends on the dataset and the range of target feature hence, RMSE can take any value and does not have any specific range. We set the performance threshold as 120% of training RMSE for further experimentations. Hence the performance threshold for the experiments are as shown in Table 10:

Table 10. Performance threshold for the datasets

Dataset Name	Performance Threshold
Energy	81.636
Air quality	0.738
Pollution	35.148

Thereafter since the datasets itself does not have any concept drift behaviour we inject artificial concept drift into the datasets to examine if the drift gets detected by our model. The error window size was chosen as shown in Table 11:

Table 11. Error window size for the datasets

Dataset Name	Error Window Size
Energy	144
Air quality	144
Pollution	288

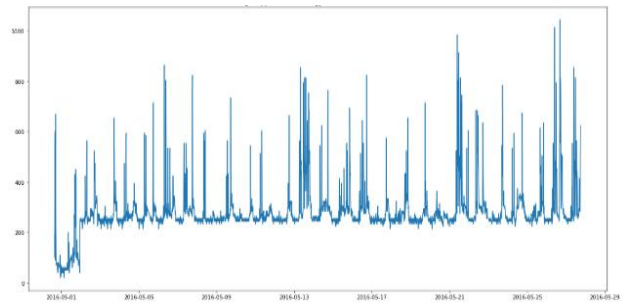


Figure 6. Test data for energy dataset with added concept drift

We injected drift by adding a shift to the target feature in Energy and Air Quality dataset as shown in Figure 6 and Figure 7. The amount of shift was chosen to be two times the mean value of the feature and the shift was added from 200 data point onwards in the test set.

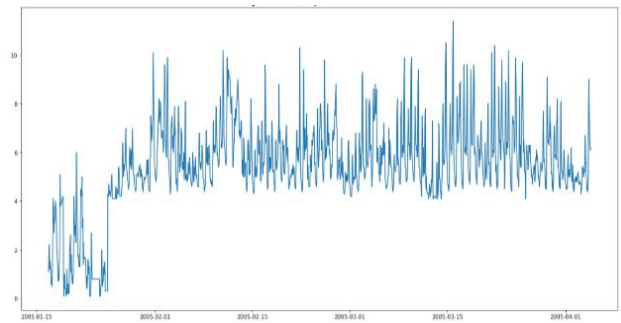


Figure 7. Test data for air quality dataset with added concept drift

Then we applied our model on these data and detected concept drift using our method without retraining the model, the results are as follows:

The RMSE is measured on the error-window. As shown in Figure 8 the Red line denotes the error threshold set. The Model detects concept drift whenever the RMSE crosses the red line hence the green blobs denote the instance when our model detects concept drift.

Whenever the RMSE of model prediction is crossing the threshold, our method is detecting the concept drift as can be seen from the method, note that we set a minimum gap between two concept drift detection to avoid continuous drift alert. The gap was set as the same as error window size. Also, since RMSE is measured on the error window, the graph of error shown in Figure 9 starts when the model has predicted at least the same number of instances as the error window size.

The details of the test datasets and concept drift deleted instances are as shown in Table 12:

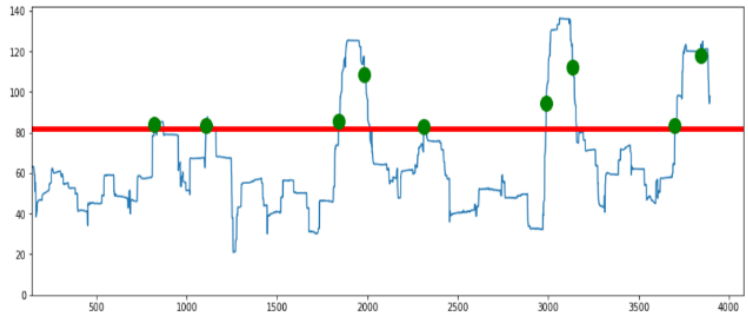


Figure 8. The RMSE of predictions made by the model on test data for energy dataset

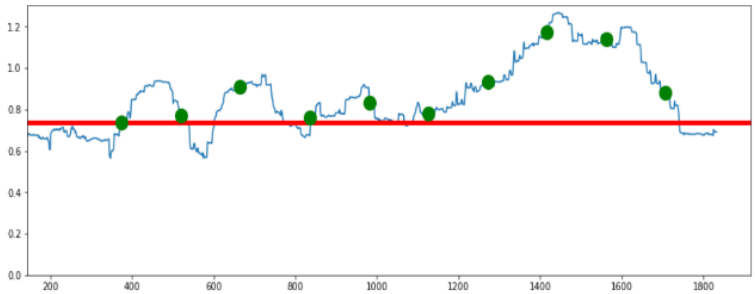


Figure 9. RMSE curve and concept drift detection on test data for air quality dataset

Table 12. Concept drift detection instances

Dataset	# of Instances in Test Data	Instances where Concept Drift Detected
Energy	3895	[820, 1106,1839, 1984, 2310, 2989, 3134, 3701, 3846]
Air quality	1832	[375, 520, 665, 837, 982, 1127, 1272, 1417, 1562, 1707]

Hence, we see that since drift is present in the data, our method successfully detects drift on a regular basis as without retraining the model is not adapting to the concept drift.

For the other data set Pollution, we applied all 4 kinds of drift ie. Sudden, Gradual, Recurring, and Incremental and tested our model performance against it.

For the other data set Pollution as shown in Figure 10 before drift inclusion, we applied all 4 kinds of drift i.e., Sudden, Gradual, Recurring, and Incremental and tested our model performance against it.

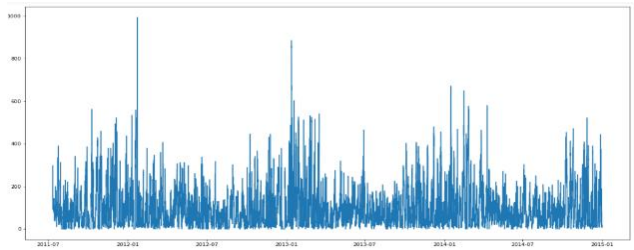


Figure 10. Pollution test set before drift inclusion

We added sudden drift followed by gradual drift and recurring the previous drift and finally incremental drift in the test data as shown in Figure 11.

The RMSE is measured on the error-window. The red line denotes the error threshold set. The Model detects concept drift whenever the RMSE crosses the red line or RMSE is higher than the red line after the minimum gap between two drift detection, hence the green blobs denote the instances when our model detects concept drift.

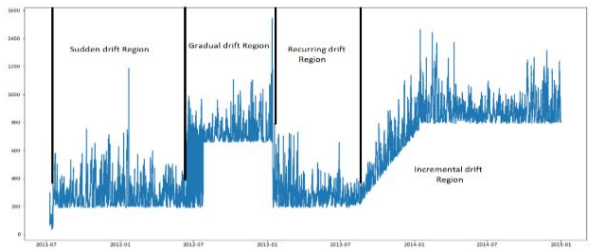


Figure 11. Pollution test set after applying all 4 kinds of drifts as indicated

We applied our model to detect concept drift while predicting on this modified test data but without any retraining, the concept drift detection result is as shown in the Figure 12 graph.

We can see that whenever a concept drift is faced in the test data the model prediction RMSE shoots higher than the threshold and remains high thus triggering the concept drift detection successfully.

Need of Injecting Artificial Drift:

• Model Validation: Using artificial drift, you may examine how well your models and algorithms respond to shifting data patterns. You can check to see if your model quickly, accurately, and without raising too many false alarms identifies drift.

• Scenario exploration: You might not always have access to historical data with identified drift points in real-world applications. You can explore numerous scenarios and evaluate how your systems manage different types of drift, such as slow or rapid shifts, by injecting false drift.

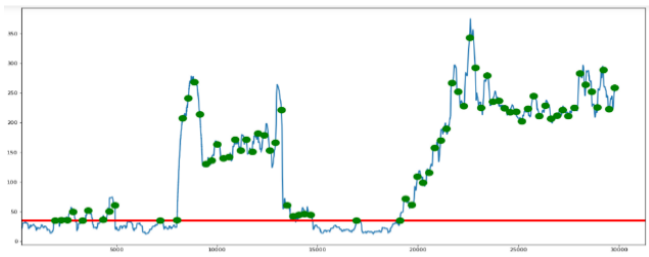


Figure 12. The figure denotes the RMSE of predictions made by the Model on the concept drift injected test data for Pollution dataset

Methodology for Injecting Artificial Drift:

Step 1: Target Features: Identify the features in your dataset that the artificial drift will have an impact on. It's crucial to pick qualities that are pertinent to the issue you're researching.

Step 2: Selecting Drift Types: Choose the forms of idea drift you want to model, such as progressive drift (data that changes gradually) or abrupt drift (data that changes suddenly and noticeably).

Step 3: Timing Window: Specify the times or intervals during which the artificial drift will take place. Indicate the beginning, duration, and end dates of the drift.

Step 4: Create Drifted Data: Alter the target features within the designated time frames in accordance with the selected drift type.

Step 5: Mix with Real Data: To produce a more realistic dataset, if at all possible, mix the artificially drifted data with real data. This integration makes sure that the simulated drift matches the distribution of the actual data.

Step 6: Run tests: Utilize the dataset that has artificial drift injected into it to run tests and assess how well your idea drift detection approaches work. Calculate the false positive and detection accuracy and time-to-detect rates.

Step 7: Fine Tune: The concept drift detection methods or models should be fine-tuned based on the findings of your studies to increase their effectiveness in spotting false drift.

7. CONCLUSIONS AND FUTURE WORK

In this paper we discussed about a lightweight RNN based model with concept drift detection using a sliding window. Since our model with RF based feature selection uses only important features, its lightweight compared to other models in the literature using full set of features. Detecting concept drift helps the model to adapt to it and maintain the model performance over the time. It is highly important to keep the model performance above expected level. We further showed our concept drift detection method on three real world time series datasets, and from results it is clear that our method successfully detects concept drift in the data stream Hence the proposed model is suitable to apply for evolving real world time series data streams. This approach improves real-time concept drift detection, enabling more effective pollution mitigation strategies and optimized energy use, ultimately promoting cleaner environments and sustainable energy practices. It is the proposed method for selecting relevant features in the air quality, pollution, and energy sectors.

Key Achievements of the Lightweight Model:

• Efficient resource consumption, making it suited for real-time or resource-constrained applications.

• Adaptability to evolving data patterns, decreasing false alarms and delays in idea drift detection.

• Scalability for managing large-scale time series data streams.

• Robust performance in numerous application domains, including air quality, pollution, and the energy sector.

The limitation of Lightweight Model:

• Potential conflicts between forecast accuracy and model simplicity.

• The need of choosing clustering and statistical methods carefully to ensure efficacy.

• Sensitivity to starting parameters that might need to be adjusted for best results.

• Compared to more complicated models, limited capacity to capture complex, non-linear relationships in data.

In future work, the lightweight method can be improved further by combining other techniques with dynamic feature section method to reduce time and memory complexity along with feature selection. In concept drift detection part, the detection can be improved to separate sudden noise from actual concept drift more efficiently and furthermore the mode can be made aware of the type of concept drift when drift is detected by the proposed method which may improve the performance further.

REFERENCES

[1] Paramasivan, S.K. (2021). Deep learning based recurrent neural networks to enhance the performance of wind energy forecasting: A review. Revue d'Intelligence Artificielle, 35(1): 1-10. https://doi.org/10.18280/ria.350101

[2] Munkhdalai, L., Munkhdalai, T., Park, K.H., Amarbayasgalan, T., Batbaatar, E., Park, H.W., Ryu, K.H. (2019). An end-to-end adaptive input selection with dynamic weights for forecasting multivariate time series. IEEE Access, 7: 99099-99114. https://doi.org/10.1109/ACCESS.2019.2930069

[3] Choi, J.Y., Lee, B. (2018). Combining LSTM network ensemble via adaptive weighting for improved time series forecasting. Mathematical Problems in Engineering, 2018: 2470171. https://doi.org/10.1155/2018/2470171

[4] Du, Y., Wang, J., Feng, W., Pan, S., Qin, T., Xu, R., Wang, C. (2021). Adarnn: Adaptive learning and forecasting of time series. In Proceedings of the 30th ACM International Conference on Information & Knowledge Management, pp. 402-411. https://doi.org/10.1145/3459637.3482315

[5] Xie, H., Zhang, L., Lim, C.P. (2020). Evolving CNN-LSTM models for time series prediction using enhanced grey wolf optimizer. IEEE Access, 8: 161519-161541. https://doi.org/10.1109/ACCESS.2020.3021527

[6] Mohapatra, U.M., Majhi, B., Satapathy, S.C. (2019). Financial time series prediction using distributed machine learning techniques. Neural Computing and Applications, 31: 3369-3384. https://doi.org/10.1007/s00521-017-3283-2

[7] Sahraei, M.A., Duman, H., Codur, M.Y., Eyduran, E. (2021) Prediction of transportation energy demand: Multivariate adaptive regression splines. Energy, 224: 120090.

[8] Riyaz, L., Butt, M.A., Zaman, M. (2022). A novel ensemble deep learning model for coronary heart disease prediction. Revue d'Intelligence Artificielle, 36(6): 825-832. https://doi.org/10.18280/ria.360602

[9] Song, Y., Lu, J., Liu, A., Lu, H., Zhang, G. (2021). A segment-based drift adaptation method for data streams. IEEE Transactions on Neural Networks and Learning Systems, 33(9): 4876-4889. https://doi.org/10.1109/TNNLS.2021.3062062

[10] Lu, Y., Park, Y., Chen, L., Wang, Y., De Sa, C., Foster, D. (2021). Variance reduced training with stratified sampling for forecasting models. In International Conference on Machine Learning, pp. 7145-7155.

[11] Fields, T., Hsieh, G., Chenou, J. (2019). Mitigating drift in time series data with noise augmentation. 2019 International Conference on Computational Science and Computational Intelligence (CSCI), Vegas, NV, USA, pp. 227-230. https://doi.org/10.1109/CSCI49370.2019.00046

[12] Read, J. (2018). Concept-drifting data streams are time series; the case for continuous adaptation. arXiv preprint arXiv:1810.02266. https://arxiv.org/abs/1810.02266

[13] Khamassi, I., Sayed-Mouchaweh, M., Hammami, M., Ghédira, K. (2018). Discussion and review on evolving data streams and concept drift adapting. Evolving Systems, 9: 1-23. https://doi.org/10.1007/s12530-016-9168-2

[14] Yang, L., Shami, A. (2021). A lightweight concept drift detection and adaptation framework for IoT data streams. IEEE Internet of Things Magazine, 4(2): 96-101. https://doi.org/10.1109/IOTM.0001.2100012

[15] Montiel, J., Mitchell, R., Frank, E., Pfahringer, B., Abdessalem, T., Bifet, A. (2020). Adaptive xgboost for evolving data streams. In 2020 International Joint Conference on Neural Networks (IJCNN), pp. 1-8. https://doi.org/10.1109/IJCNN48605.2020.9207555

[16] Fekri, M.N., Patel. H., Grolinger, K., Sharma, V. (2021). Deep learning for load forecasting with smart meter data: Online adaptive recurrent neural network. Applied Energy. 28: 116177. https://doi.org/10.1016/j.apenergy.2020.116177

[17] Guo, Y., Han, S., Shen, C., Li, Y., Yin, X., Bai, Y. (2018) An adaptive svr for high-frequency stock price forecasting. IEEE Access, 6: 11397- 11404. https://doi.org/10.1109/ACCESS.2018.2806180