An Efficient Approach for Transformation and Analysis of Streaming Data

A Thesis
submitted in fulfillment of the requirements
for the award of degree of

Doctor of Philosophy

by
Shruti Arora
(Roll No: 901803011)

Under the Guidance of
Dr. Rinkle Rani
(Professor)

Dr. Nitin Saxena
(Assistant Professor)

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT
THAPAR INSTITUTE OF ENGINEERING AND TECHNOLOGY
PATIALA – 147004

December 2023

Table of Contents

LIST OF TABLES

LIST OF FIGURES

Certificate

I hereby certify that the work which is presented in this thesis entitled **"An Efficient Approach for Transformation and Analysis of Streaming Data"**, in fulfillment of the requirement for the award of degree of **"Doctor of Philosophy"** submitted in Computer Science and Engineering Department of Thapar Institute of Engineering & Technology, Patiala, is an authentic record of my own work carried out under the supervision of **Dr. Rinkle Rani** and **Dr. Nitin Saxena** refers other research works which are duly listed in the reference section.

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

Shruti .

(Shruti Arora)

Regn. No. 901803011

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Rinkle 05/12/23

Dr. Rinkle Rani
(Professor)

05/12/2023

Dr. Nitin Saxena
(Assistant Professor)

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology, Patiala

Acknowledgement

I am deeply grateful to have the guidance and support of many individuals throughout my journey in completing this Ph.D. thesis. Their invaluable contributions have shaped my research and enriched my academic experience in profound ways.

First and foremost, I extend my heartfelt gratitude to my supervisor, Dr. Rinkle Rani whose unwavering dedication, insightful guidance, and immense expertise have been instrumental in shaping the trajectory of research. I am thankful to my co-supervisor, Dr. Nitin Saxena for his consistent encouragement, constructive feedback, and invaluable suggestions. The collaborative spirit and wealth of knowledge have broadened the horizons of my research and enhanced the rigor of my work.

My sincere thanks to Dr. Shalini Batra, Professor and Head, CSED, Thapar Institute of Engineering and Technology, Patiala, for providing me the necessary administrative assistance and infrastructure that helped me in the completion of my research work. I extend my sincere appreciation to the members of my doctoral committee, Dr. Ashutosh Mishra Assistant Professor CSED, Dr. Nidhi Kalra Assistant Professor CSED, and Dr. Kulbir Singh Professor ECED, Thapar Institute of Engineering and Technology, Patiala, for their thoughtful insights, critical feedback, and rigorous examination of my thesis.

behind my academic pursuits. I am also grateful to my sister Diya, for her belief and willingness to lend an ear, even during her own busy schedule, have been a source of strength. I also acknowledge the cooperation and encouragement by my friends all this while.

Last but not the least, I bow my head in front of my God for showering His grace on me, without which any of this, would not have come to life.

Patiala

(SHRUTI ARORA)

December, 2023

Abstract

The era of digitalization has ushered in an unprecedented deluge of data generated from various sources, leading to the emergence of data streams as a critical paradigm in data analysis. This PhD thesis delves into the intricate domain of transforming and analyzing data streams, addressing the challenges and opportunities presented by their high-velocity, dynamic, and often unbounded nature.

The primary objective of this research is to develop efficient methodologies for effective transformation and analysis of data streams, catering to the unique characteristics and demands of real-time data processing. This study encompasses a comprehensive review of existing techniques, algorithms, and tools pertinent to data stream processing, while also identifying the gaps and limitations that need to be addressed.

One of the major contributions of this thesis is the development of adaptive and scalable data stream transformation techniques. By integrating concepts from machine learning and statistical modeling, these techniques facilitate the automatic adaptation of transformation process according to the evolving nature of data streams. This adaptability not only enhances the accuracy of downstream analysis but also ensures the robustness of the transformation pipeline in handling dynamic data distributions.

Furthermore, this research explores advanced methodologies for the analysis of transformed data streams. The development of techniques for real-time analysis, capable of handling high-velocity data streams while providing timely insights, is a

key focus. These methods incorporate concepts from data mining, pattern recognition, and event detection to enable the extraction of valuable patterns, trends, and outliers from rapidly evolving data.

In addition to the technical contributions, this thesis also explores the practical implementation and deployment aspects of the proposed techniques. A framework for building scalable and resilient data stream processing pipelines is proposed, considering factors such as computational efficiency, fault tolerance, and resource optimization. Moreover, the integration of visualization techniques aids in the effective communication of insights derived from the analyzed data streams.

To validate the effectiveness of the proposed methodologies, a series of experimental evaluations are conducted using real-world data streams from IoT and social media domains. In addition to real world data streams, synthetic data streams are also generated using open source tool MOA for validation of proposed techniques. The results demonstrate the superiority of the developed techniques in terms of accuracy, efficiency, and adaptability in comparison to existing similar approaches.

In conclusion, this PhD thesis contributes to the advancement in the field of data stream processing by addressing the challenges associated with transforming and analyzing high-velocity data streams. The developed methodologies not only enhance the quality of insights extracted from dynamic data but also lay the foundation for more informed decision-making in various application domains. This research opens avenues for future work in the areas of adaptive analytics, real-time visualization, and integration of emerging technologies into data stream processing pipelines.

Chapter 1

INTRODUCTION

The proliferation of data in the years is due to a tremendous rise in the use of smart gadgets and technological breakthroughs. Dynamic advancements such as online transactions have replaced currency mode of trade, entertainment devices such as televisions are eclipsed by mobile streaming and over-the-top (OTT) platforms, smart home IoT devices have replaced traditional devices and, therefore, serve as a powerhouse of data these days. The data is increasing at jitter and is expected to rise 44 times as much volume as in 2009 [1]. This rise in data gives birth to a situation where handling and processing such voluminous data becomes tedious. The big continuously flowing data with characteristics such as high complexity and variability, unordered, unbounded sequence of items is known as Streaming Data or Data Streams [2]. This flavor of Big Data cannot be handled by traditional big data frameworks such as Hadoop. Therefore, the need for techniques and tools arises to draw valuable insights from such massive dynamic data. Data stream applications frequently need advanced processing capabilities for persistent input stream monitoring to identify events or changes. Hence, accessing and effectively interpreting streaming data is essential to extract relevant information from this constantly expanding dataset. In recent years, researchers have developed many data stream mining strategies to get knowledge from such data. However, the traditional methods were typically used for static datasets, where a pre-determined dataset was gathered and processed; currently, real-time data analysis is required. Hence, mining data streams is a notable area of research that examines algorithms and techniques for processing and extracting knowledge from streaming data.

1.1 Characteristics of Streaming Data

Data streams can be described in more generic terms as follows:

i) *Size*: A data stream may have an unbounded size.

ii) *Volume*: A continual stream of real-time data is being produced. The size of the dataset created from different sources is unknown and heterogeneous [3].

iii) *Volatility*: Given that the system's resources are limited and it cannot save all of the data received, once it has been analyzed, the data is either summarized or destroyed in data stream mining.

iv) *Drifting:* When the probability of data distribution alters with time and it becomes difficult to test the data on the previously trained model, it is said to suffer from concept drifts [4], [5].

1.2 Data Stream Challenges

When working with large data streams, there are numerous difficulties as stated below:

i) The processing of ever-increasing voluminous data using multiple passes is challenging. Therefore, algorithms required to process data streams should especially be implemented with the constraint that they could execute in a single pass [6].

ii) The techniques available for stream mining typically have an intrinsic time component due to the possibility of data evolution. Temporal locality is the term used to describe this characteristic of data streams [7]. As a result, a simple adaption of one-pass mining algorithms might not be the best way to

handle the problem. The development of underlying data must be a clear priority in the careful design of stream mining algorithms.

iii) The query workload may alter due to the stream's unpredictable and bursty nature, which can cause substantial fluctuations in stream arrival rates. For instance, the incoming stream rate may exceed 500000 packets per second during a distributed denial of service (DDoS) attack on an IP network [8]. It is crucial for Data Stream Management System to remain steady in certain hostile situations and to give useful user input.

iv) High-dimensional data streams may include text documents, audio or video data. Therefore, the distances between instances increase rapidly for certain types of data, which may have an impact on how well a stream classifier performs.

1.3 Transformation of Data Streams

Streaming data is captured from heterogeneous sources and contains inconsistencies and abnormalities. Hence, it is the need of the hour to transform the metadata as it becomes easier for business managers or owners to get insights into the data to make better decisions for the growth of their business. Moreover, it becomes easy to retrieve transformed data from virtual machines if standardized.

The role of data transformation is prominent in all industries ranging from healthcare to transportation. Therefore, it is the secondary step after data extraction in ETL (Extract-Transform-Load) process [9]. In this step, some tasks such as Filtering, Cleaning, Joining, Splitting, Sorting, etc. are performed. The overall process involves a directed graph known as a stream application that consists of a source stream as an initial node. Applying various transformations, the stream is processed

and examined before being transferred to a data sink such as a file or a database. The source data can be captured from Kafka data pipelines [10], files, heterogeneous sensors, NoSQL databases, etc. The data streams are transformed into a Stream object known as the source Stream. A single or many transformations are applied to the source Stream. Every transformed stream object calculates the average of the data stream while removing inaccurate data. The final stream, which contains the results, is transmitted to a data sink which is a component for saving the data to an external system.

1.3.1 Types of Transformation Techniques

Streaming Data entails for two types of transformation techniques that is Stateful and Stateless Transformation [11]. In Stateful transformation, a micro-batch of data is considered and used as it is during processing and depends partially or entirely on the processing of the preceding batches of data. In this research work, we have used Stateful transformation for the detection of drifts and events and to maintain the states in case of recurrent drifts. Some computations in this work use *updateStateByKey* operation.

1.3.1.1 Stateful Transformation

The incoming data is divided into small groups called Discretized Streams (DStreams) using micro-batch architecture. In order to track streaming data across batches, stateful DStreams are required. For instance, we could monitor a user's activity while they are on a website, or we could monitor a specific Twitter hashtag over time to see who is using it. Stateful DStreams come in two flavors: window-based tracking and full-session tracking. All incoming data should be converted to key-value pairs for stateful tracking to monitor the key states across modules.

1.3.1.2 Stateless Transformation

Each batch in a stateless transformation is completely independent of any batches of data that came before. For instance, some simple RDD transformation functions like map(), flatMap(), filter(), repartition(), reduceByKey(), groupByKey() are applied to each batch. Stateless and stateful concepts are not available in Kafka's KStream. Otherwise, it works with all other versions of stream processing engines, including Hadoop MapReduce, Apache Spark, Apache Flink, and Apache Storm.

1.4 Event Detection in Data Streams

Event detection is an intriguing topic, but the name itself doesn't reveal anything about what is really observed. Every tweet on Twitter might be viewed as its own independent event. It would be pretty simple to construct an event detection system that only looked for tweets, but that is obviously not what an event detection framework is designed to do. Everyone has an intuitive idea of what would be detected when discussing event detection; most frequently, news-worthy topics come to mind as examples of what the outcome would entail. Events are alerts to a status update. Potential users can subscribe and respond to notifications after they are published or issued. Usually, the system from where an alert is sent doesn't know what was done with it and doesn't get any confirmation that it was handled.

The primary driver of an event detection approach is an outlier detection technique. In streaming data, an event is defined as a series of anomalies with temporal and geographical correlation. Therefore, online and offline, or retrospective, strategies are both used in the literature for event detection. Our work involves detecting events online or, in other words, without storing them in memory. An assumption that a sudden spike in the volume of mentions of a specific subject is a reliable sign of an

event. Some consideration will be given to what can and will be regarded as event as part of this thesis.

It is crucial to understand the distinction between event detection and complex event processing (CEP) in order to clarify the scope of the work. Although the definition of the event is clear in event detection, its structure is not; in contrast, the event structure in CEP is complex but is defined through patterns. Hence, CEP is extracting instances that comply with the predefined pattern under low latency [12]. However, event detection is the main emphasis of this research.

For many years, low-latency and high-throughput stream processing platforms have been in use. High-performance event detection investigations are being set up on these platforms. A few studies [13] explore stream processing platforms along with event detection applications, however, there aren't many studies dealing with high-accuracy event detection techniques built on top of stream-based platforms.

In traditional techniques, an offline module is developed for research on several deep-learning models that work well for real-time event detection. In this thesis, trials are carried out with the datasets from PubNub sensors [14] that demonstrate how recurrent neural network's long short-term memory (LSTM) unit is the most effective model for anomalous event identification. To forecast the anomalous peaks, an online pipeline module is constructed using streaming data frameworks.

Event detection aims to identify instances at a specific time interval that act differently from other instances and is a crucial component of streaming data analytics. Wireless sensor networks, IOT sensors, social media, and financial transactions are widespread and produce enormous amounts of data every second in a variety of application fields, including communication networks, healthcare

monitoring, weather forecasting, etc., due to advances in sensor technologies. As a result, numerous application domains require event detection and streaming data analytics.

There are two essential categories for classifying events: unsupervised event detection, where events are found in unlabeled data and supervised event detection, where events are found in labeled data. Several density-based methods, such as K-Nearest Neighbour (K-NN), one-class support vector machines (SVM), autoencoders [15], Hidden Markov Models (HMM) [16], etc., can be used to detect events. The real-world example is in the case of natural disasters; adaptive monitoring is required, which calls for higher sampling. Such applications require that the event detection approach be rapid and carried out incrementally to ensure the discovery of data deviations in near-real-time.

1.4.1 Need for Event Detection

Event detection is an essential component of data stream processing that enables quicker response times and gives rise to the possibility of taking preventative action before a scenario is addressed. Several applications where event detection plays an eminent role are:

a) News Articles: These are details of textual news information that are trustworthy and relatively low-frequency. Therefore, critical event detection in long texts is important so that the concerned people take appropriate actions well in advance.
b) Twitter: It is a microblogging social media website that is prone to rumors and spam information. Many people share their personal opinions on Twitter;

therefore, the interpretation of their thoughts plays a vital role in applications such as Twitter Sentiment Analysis.

c) Fraud Detection: Some applications such as credit card fraud detection, phishing attack detection, and catching frauds on the fly helps in eliminating the risk in advance.

d) Heart-rate Event Monitor: This is one of the most crucial applications in the domain of medicine. Although this technology is not new, addressing the anomalies in the prediction of heart attacks can be fulfilled by detecting critical events in streaming healthcare datasets.

1.4.2 Categories of Event Detection Approaches

Detecting anomalous events in massive data poses various processing challenges, such as increased storage and time overhead. However, in literature, there are four broad categories of event detection techniques.: Statistical, Probabilistic, Artificial Intelligence and Machine Learning, and Composite [17]. The classification for event detection approaches is given in Figure 1.1.

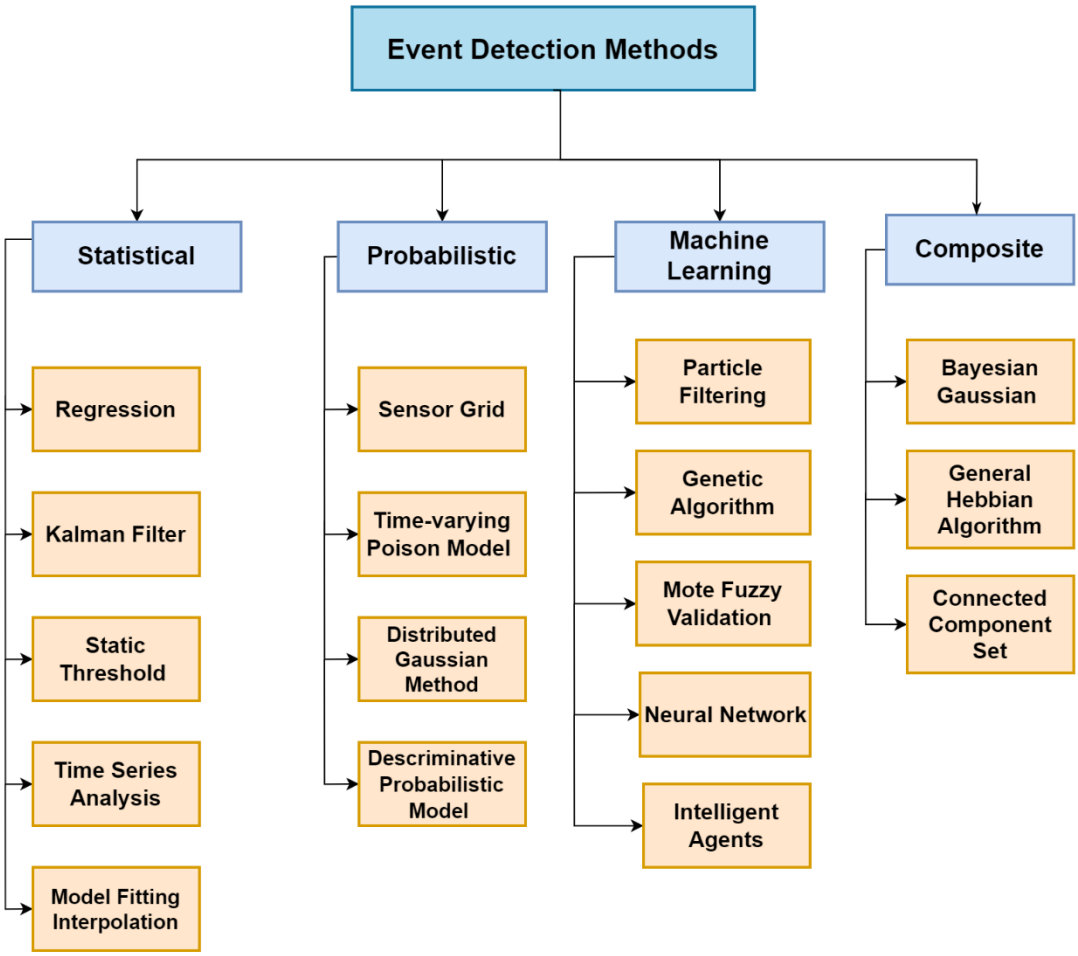


Figure 1.1: Classification of Event Detection Approaches in Streaming Data

This research focuses on processing, transforming, detecting, and classifying events in data streams. Further, other important categories are dependent on production, such as Online Learning and Offline Learning [18].

Online Learning: Unlike batch learning, the data is incrementally trained in mini chunks of data on the online server, where the data is updated continuously. For example, Chatbots, Alexa and Siri follow an online learning approach because they train the model simultaneously whenever new data is fed incrementally. The main application of online learning is in the scenario of concept drift, where the events are

detected on-the-fly. This learning method is more cost-effective as compared to batch learning.

Offline learning: The data is gathered over time in this category, and the machine learning model is periodically trained using the accumulated data in batches. As the model cannot learn sequentially from a stream of real-time data, it is the exact reverse of online learning. The machine learning algorithm does not modify its parameters until fresh data are consumed in offline learning.

Statistical Techniques for Event Detection

Real-time data necessitates online techniques; thus, it is necessary to comprehend how statistical techniques relate to offline versus online processing. Analysis, as described, is intrinsically offline as it involves human expertise, nature and education. It is tedious for human beings to perform highly challenging real-time reasoning tasks. The task of event detection can be accomplished using a few statistical techniques such as Regression, Kalman Filters [19], Model Fitting Interpolation [20], Time-series analysis [21], Static Threshold [22] and many more. In this work, a long short-term memory (LSTM) network based on regression is employed for event detection as a predictor that learns higher-level temporal information to predict potential values.

Probabilistic Approaches for Event Identification

The techniques for event detection that are related to probabilistic algorithms involve Sensor Grid [23], Time-varying Poisson Process Model [24], Distributed Gaussian Model [25], and Discriminative Probabilistic Model [26]. In this work, we have used a time-varying Poisson process model for the detection of anomalous events in data

streams. We also demonstrated how statistical estimate methods may be used to infer the model parameters using time-series data.

Machine-Learning based techniques for Event Detection

We also conducted experiments for event detection in data streams by employing both supervised and unsupervised machine-learning algorithms. In the case of supervised machine learning algorithms, the learning algorithms such as Random Forest and Support Vector Machine are utilized. In contrast, some artificial intelligence-based algorithms like Isolation Forest [27], Local K-Nearest Neighbor and the Outlier Factor have been used in the unsupervised category. We have also explored the SparkML component of Apache Spark Streaming for testing our dataset and training the models. There are some techniques, such as Particle Filtering [27], Genetic Algorithms [28], Intelligent Agents [29], and Mote-Fuzzy Validation [30] that have been studied during the implementation phase of our research work.

Composite techniques for Event Detection

A composite event is made up of many elementary occurrences. In the past literature, Eigen Value Decomposition (EVD) based algorithms require large batches of measurements to accurately calculate principal components. Therefore, most event detection techniques are based on dimensionality reduction using the General Hebbian Algorithm (GHA) [31]. This approach permits the online calculation of the percentage contributions of individual attributes to detected events. There are other simple approaches for the detection of composite and global network events along with minimization of communication overhead.

1.4.3 Challenges in Event Detection

- Most existing solutions, while efficient, lack scalability and thus, do not match the context of streaming data. Furthermore, contemporary research in the area of event detection is focussed on common memory architecture, that limits the capability to deal with huge volumes of data.

- The cacophony on social media is overwhelming (advertisements, spam messages, hoaxes, URLs, etc.). As a result, there is a lot of noise, which eventually has a severe impact on event detection evaluation parameters such as accuracy and performance. Resultantly, identifying tweets/posts that depict real-world occurrences among the polluted contents has become an essential step for event detection from social networks.

- Data streams are examined for a specific period; however, most event detection systems compare this data with other previously saved datasets. Continuous data streams make it impossible to store every bit of the data.

- Existing event detection algorithms are focused on particular application areas. There isn't a method that can identify each type of event or drift across different fields. As a result, a generalized technique can be proposed for other domains.

- Existing methods for the identification of events using machine learning are primarily concerned with features or entities while ignoring the relationships between them.

With the tremendous growth of digital data in numerous areas in recent years, there has been a spike in demand for building scalable, consistent, and exact algorithms. As a result, distributed systems dealing with massive amounts of data accurately are in great demand.

1.5 Research Contributions

The research is primarily focused on design and implementation of efficient techniques for transformation and then detection of events from data streams. The significant contributions of this study are as follows:

- The event detection strategies are thoroughly reviewed. Approaches for concept drift detection in streaming data have been discussed in detail.
- The event detection research topic is thoroughly examined, and numerous solutions are suggested.
- Different strategies for detecting scalable events in dynamic datasets are proposed. The proposed techniques are:
    1. Social network data streams from Twitter were captured to detect rumors related to Covid-19 by using filtering techniques for hashtags, re-tweets and user's social networks. The filtered streaming data was fed into the spark streaming framework for further processing and detection of the highly rumored tweets. In this work, we proposed and designed a modified VADER algorithm to classify tweets into various rumor categories.
    2. The streaming data captured from ambient weather sensors is transformed to data frames to overcome the processing latency issue. Further, an offline component integrated with an online component is proposed to capture events in approximately real time. The concepts of early-stopping mechanism and contextual layer are introduced in the proposed model, resulting in better time efficiency and optimization. The proposed approach performs better than existing approaches for event detection in terms of classification accuracy and time efficiency.

3. To predict unusual peaks, the online pipeline module is constructed with streaming data frameworks. The experimental results show that the proposed approach detects abnormal occurrences continuously while also eliminating false positives.
4. Further, our research focused on the detection and classification of concept drifts. To achieve this objective, we utilized an ensemble classifier based on transfer learning technique. Experimental evaluation of the proposed approach is performed using various statistical parameters along with the Massive Online Analysis (MOA) tool under different scenarios.

1.6 Thesis Organization

The organization of the thesis is as follows:

Chapter 1: Introduction

This chapter introduces the concept of streaming data. It discusses various challenges of transformation and detection of anomalous events in massive data streams. Then, it proceeds to describe the significance of events/drifts in data streams, laying a foundation for the introduction of the concept of event detection in streaming data. In a nutshell, this chapter explains the fundamental concepts of streaming data and how they relate to the concept of event detection. This chapter partially addresses Objective 1.

Chapter 2: Background and Related Work

This chapter provides an in-depth review of the literature on data streams and event detection, with a focus on events in social networks and concept drift detection. It discusses existing classification and clustering-based approaches for event detection. A collection of case studies demonstrates that discovering a

correlation between characteristics is the most effective technique to cope with the problem of detecting abnormal events in streaming data. Finally, this chapter includes a few comparison studies that lay an emphasis on the benefits and limitations of the methodologies presented. Research Objective 1 is accomplished in this chapter.

Chapter 3: Problem Formulation

This chapter highlights the research gaps that were discovered while outlining the research objectives. These gaps illustrate the flaws in existing streaming data techniques. These approaches will be valuable for designing scalable applications. The four research objectives are established and addressed in various chapters of this thesis.

Chapter 4: Transformation and Analysis of Input Data Streams

In this chapter, two types of transformation schemes and an improved classifier for the detection of rumors during the time of pandemic are discussed. The two transformation approaches namely stateful and stateless approaches are mainly discussed as they are both used to regulate the incoming social media data streams before processing them into stream processing frameworks. Apache Spark based streaming model is used to design the proposed approach in the distributed setting. The improvised classifier for rumor detection, classification and analysis of twitter streams is proposed to accomplish the task of detection of misinformation in real-time as well as classify it by analyzing the source of misinformation. This chapter includes the details of the experiments performed on real-time data streams from Twitter to validate the proposed approaches.

Standard parameters are used to evaluate the performance of the proposed approach. This chapter partially accomplishes the research objective 2, 3 and 4.

Chapter 5: An Approach for Online Event Detection in Data Streams

In this chapter, an offline-online approach for event detection is proposed. The existing offline techniques mostly handle big data algorithms lack scalability and consistency while handling data streams. A novel 2-component approach is implemented by training machine learning models such as RNN and LSTM on few chunks of data and then the events are detected using distributed Apache Spark framework which is an online framework for processing streams of data. This approach is specifically designed to handle weather data streams but can also be validated on other data sources. To validate the proposed approach, tests are run on streams of real-time sensor data. The proposed approach is tested in the first round of experimentation, together with current state-of-the-art event detection algorithms. The proposed algorithm's performance is measured using common parameters. It succeeds in achieving objectives 2, 3, and 4.

Chapter 6: Detecting Concept Drifting Events in Data Streams

This chapter discusses dynamic ensemble based techniques leveraging selective transfer models for detection of concept drifting events in data streams. The proposed technique introduces a selective parameter for transferring the effective models in the ensemble to minimize the training time and avoid negative transfer challenges. The proposed approach obtains high prediction performance when tested using real and synthetic data streams. The comparison with other existing ensemble based approaches shows performance enhancement in contrast to other techniques.

Chapter 7: Conclusion and Future Directions

This chapter concludes the thesis by providing a summary of the proposed methods and techniques for event detection in data streams. In the chapter, insights regarding the future scope of labor are also provided. It highlights the contributions and identifies the areas that require further development.

1.7 Chapter Summary

This chapter covers the characteristics of streaming data and its associated challenges. Also, the concept and need for event detection in data streams are discussed in detail. Further, various techniques available in the literature for detecting events are explained. The Research Contribution section also includes a quick overview of the methodologies offered in this study. Finally, the thesis organization is discussed. The following chapter will go through various existing methodologies for event detection in detail.

Chapter 2

BACKGROUND AND RELATED WORK

This chapter comprises of a comprehensive survey of application areas, transformation techniques, approaches for event detection, and concept drift detection and adaptation techniques in data streams. Section 2.1 discusses various application areas of event detection in streaming data. Section 2.2 discusses the data stream transformation frameworks. Section 2.3 highlights some existing techniques for event detection in streaming data. Section 2.4 elaborates the field of event detection with a focus on concept drift detection techniques.

2.1 Application Areas of Event Detection in Streaming Data

Streaming data is unbounded and consistently generated data sequence that essentially arrive in Data Stream Management System at high staggering rate and contains unfathomable data. The widespread use of smart phones, social media and microblogging services have become the largest source of publicly accessible data. People can now use social media at any time and from any location, and it has become the primary platform for people to express their thoughts and respond to current issues of a particular event. As a result, social media plays a crucial role in the detection and analysis of events. There are other use cases of streaming data such as location intelligence, fraud detection, sales and market analysis, real-time stock market analysis, user activity monitoring, log monitoring, cab tracking service, warehouse inventory management, database migration, real-time elderly patient monitoring, etc summarized in Table 2.1 that consists of a technique used and desired outcome for each application area.

Social (network) Analysis

Contemporary studies on data stream analytics have presented that the field of rumor detection is one of the most crucial fragments of prediction-based and data-driven approaches. The analysis of Castillo et al. [32] denotes the beginning stage for some subsequent studies. Castillo et al. [33], stated the process of defining elements for Twitter credibility evaluation.

Intrusion Detection

The widespread use of internet in daily life has made network security the most critical foundation for all web-based applications such as online transactions, online retail, and businesses. Data confidentiality is lost as a result of internet intrusion via various means of internet access [34]. In some cases, network traffic intrusion is referred to as a cyber-attack or a malicious attack [35]. The primary objective of [36] is to detect anomalies in mobile network signaling traffic. A mobile network for example where an anomaly detection technique is critical for identification of abrupt changes in signal traffic. The characteristics of signal traffic, such as data in terms of TBs, the number of multidimensional data events per second, and the speed of data events per second, are used to identify traffic events. Previous studies [37]–[39] on intrusion detection system primarily focus on system or user behavior modelling from monitored system log or accounting log data, such as CPU utilization, login time, user session duration, and so on.

Fraud Detection

Design and implementation of scalable learning techniques that are capable of ingesting and analyzing enormous amounts of streaming data is necessary for

detecting frauds in (almost) real-time settings. The field of fraud detection is now exploring new perspectives due to recent developments in analytics and the accessibility of open source Big Data storage and processing solutions. Carcillo et al. [40] introduced a Scalable Real-time Fraud Finder (SCARFF) that combines machine learning techniques for dealing with imbalance, non-stationarity, and feedback latency with Streaming Data tools (Kafka, Spark, and Cassandra).

Borgne et al. [41] investigated the accuracy of various active learning techniques in fraud detection. For querying unlabeled transactions, they compared various techniques based on supervised, semi-supervised, and unsupervised fraud detection. Also, they highlighted the fact that active learning in the context of fraud detection has a trade-off between exploitation and exploration and has been ignored in the literature till date.

Khine et al. [42] introduced an Online Boosting (OLBoost) approach by using Extremely Fast Decision Tree (EFDT) as a base (weak) learner first before combining it with other online weak learners to create a single online strong learner for the application of credit card fraud detection.

Healthcare

Real-time event processing is more important in the healthcare industry than in other sectors because of need for early diagnosis, and efficient patient treatment through the large amount of data. Application of CEP method in this area is introduced to draw actionable insights, forecast anomalies, and improve healthcare quality. Rahmani et al. [43] presented an event-driven IoT architecture, including context, event, and service layers, for data analysis of trustworthy healthcare applications. The above said CEP method is a novel approach with the integration of automated

intelligence in the event layer, where dependability parameters are taken into account in each layer. Nair et al. [44] implemented machine learning techniques to achieve an objective of building a real-time remote health status prediction system. They designed an Apache Spark based framework deployed in the cloud. In their work, they leveraged Spark MLlib library to form a decision tree model for the prediction of health status by using tweets captured using Twitter API.

Weather Prediction using IoT sensors

Kanavos et al. [45] designed and implemented prediction model for weather data focussing several types of winter precipitation leveraging Apache Spark Streaming framework. Their work was validated using a number of classification models encompassing the Bayesian, decision trees, and ensemble methods of classification. The regularization technique implemented by them significantly improved forecasting performance.

Table 2.1: Application Areas focusing Event Detection in Data Streams

Author(s)	Application Area	Summary	Outcomes
Maarala et al. [46]	Transportation	Apache Flume is suitable for ingestion of real-time data to Spark Streaming framework for analysis when appropriately configured.	Data storage to Impala is fast even though it includes merging the data tables.
Ameer et al. [47]	Smart Urban Planning	An architecture for pollution monitoring in a smart city to solve the challenge of pollution tracking.	Processing using Spark is 10 times faster as compared to data frames.
Amen et	Collective	A novel distributed	The experimental

al. [48]	Anomaly Detection	collective anomaly detection approach for large-scale IoT event sensor streams. Leveraged a snapshot model over the sequence of event stream tuples to compute anomalous events.	results are evident of collective snapshot model detection as well as capability of the proposed distributed anomaly detection method.
Jin et al. [49]	Adaptive Language Learning System	Presented a novel approach using complex event processing as a behaviour tracker to process the events of interactive learning.	Complex event processing is designed for massive amount of data flow, it is not effective for processing archived data.
Shah et al. [50]	Disaster Management	A conceptual reference model for the implementation of BDA and IoT-based disaster management was put forth by the authors.	Provided quick emergency response and prevent unavoidable disasters
Jain [51]	Telecommunication Industry	The authors implemented detection of telecommunication fraud using neural network classification-based data mining.	The accuracy of both the techniques have been compared.
Castillo et al. [32]	Social Network Analysis	Identified groups of semantic and syntactic micro posts that correlate with the supporting documents by selecting	Easily scales with massive incoming micro posts.

		important feature combinations from frequent pattern mining	
Sovilj et al. [52]	Intrusion Detection	Empirically compares deep learning frameworks for the problem of intrusion detection.	RNN variation surpasses a popular non-sequential deep auto-encoder that is often used for unsupervised intrusion detection system.
Carcillo et al.[40]	Fraud Detection	Developed scalable real-time fraud finder by integrating machine learning techniques with Big Data tools.	The system is scalable, effective, and accurate over a large stream of genuine credit card transactions.
Rahmani et al. [43]	Healthcare	An event-driven architecture is proposed that includes context, event, and service layers. Each layer takes into account dependability parameters, and the event layer uses the CEP method.	CEP approach improves healthcare quality by decreasing expenses and increasing reliability.
Kanavos et al. [53]	Weather Prediction	It aims to classify the weather type into rain, freezing rain, and snow.	The classification results produced by OzaBag and HoeffdingOption Tree [54] were optimum.

2.2 Streaming Data Transformation Frameworks

The primary goal of stream transformation frameworks is to process an enormous volume of data streams and to make decisions instantly. As the streaming data has become more prevalent, numerous organizations have started using stream frameworks to address the critical big data issues relevant to smart ecosystems, healthcare services, social media, etc. For instance, in smart cities scenario, a variety of sensors, including GPS, weather stations, smart cards for public transportation, and traffic cameras, are installed on various regions (e.g., water lines, utility poles, buses, trains, traffic lights) [55]. Leveraging in-stream architectures with iterative learning and processing capabilities enables an efficient implementation of particular tasks like social network analysis, stock-market prediction, etc. Finding an appropriate framework for data stream applications becomes a challenging task given the significance of the mentioned real-world scenarios. A variety of stream processing frameworks along with their limitations have been briefly summarized in Table 2.2. The literature contains several state-of-the-art proposed systems and some of them are discussed below:

a) Apache Spark

The robust data stream processing framework Apache Spark [56] offers an ease of use architecture for effective analysis of multi-dimensional data. It was initially developed in 2009 at UC Berkeley [57]. Resilient Distributed Datasets (RDDs) is a fundamental concept in Apache Spark (RDDs). An RDD is a distributed and immutable collection of objects in Spark clusters [58][59]. There are two different ways to operate on RDDs in Spark: transformations and actions. The implementation of transformation operations like map, filter, union, and join involves creation of

new RDDs from historical ones. The final result of RDD computations makes up actions.

b) Apache Storm

An open source framework called Storm [60] allows large amounts of structured and unstructured data to be processed in real-time. Storm is a framework that can handle faults and is appropriate for sequential and iterative computation, machine learning, and real-time data analysis. A directed acyclic graph is used to represent a Storm program (DAG). Data transfer is represented by the edges of the program in DAG. Spouts and bolts are the two different types of nodes that make up DAG. The data sources are represented by the spouts (or entry points) of a Storm program. The operations that are performed on data are represented by the bolts. To process the data concurrently, Storm distributes bolts across a number of nodes.

c) Apache Flink

Flink is a popular framework for batch and real-time data processing [61] that offers fault tolerance characteristic. Flink, as opposed to MapReduce, provides extra high-level functions like join, filter, and aggregation, however, the programming model is similar. The streaming data that is gathered using various tools like Flume and Kafka can be processed iteratively and in real-time using Flink [62]. It provides a number of abstract APIs that allow users to implement distributed computation.

d) Apache Samza

An open source distributed stream processing framework Apache Samza [63] was developed by LinkedIn to offer a variety of stream processing capabilities, including data tracking, service logging, and data ingestion pipelines. Large messages can be handled using Samza, which also provides file system persistence for them. It uses

Hadoop YARN [64] for distributed resource allocation and scheduling and Apache Kafka as a distributed broker for messaging. Samza uses the YARN resource manager daemon to provide the cluster with fault tolerance, processor isolation, security, and resource management.

e) Apache Kafka

The most popular mechanism for importing data streams into processing platforms is called Kafka. This framework has the feature of sending messages at end-points using a distributed public-subscribed messaging system because of its ability to handle a large volume of data [65]. It is highly scalable and reliable because it can be partitioned, replicated, and distributed. Every topic in Kafka's data storage system can further be subdivided into one or more partitions. A partition is kept as an immutable log, or series of records. Consumer clients can read partitions while producers can append new ones continuously.

Table 2.2: Stream processing frameworks proposed in existing literature

Author(s)	Input Data	Nature of Data	Proposed Work	Limitation(s)
Patel et al. [66]	Streaming IoT Sensor Data	Streaming	Raw data processing technique for streaming IoT sensor data.	Lack of spatial and temporal reasoning.
Osman [67]	Heterogen e-ous dataset	Streaming	Flume, Kafka, Sqoop	Just a review of challenges of big data analytics.

P. Talagala et al. [68]	Synthetic and real-world datasets	Synthetic and dynamic	Framework to compute the boundary for the system's behavior using extreme value theory. To look for anomalous series, a sliding window is employed.	Security issues not addressed.
M. Hasan et al. [69]	Twitter Data Stream	Streaming data	Incremental clustering based approach.	Temporal Summarization in event cluster not considered.
B. Leang et al. [70]	Sensor Data	Spatio-temporal data	Hadoop and HBase for storage, Kafka for pipelining and Spark for stream processing.	Security of database in the server is not considered.

2.3. Event Detection Techniques

Event detection in data streams employs methods from numerous fields that have been extensively covered in the literature, including machine learning and data mining [71], [72], natural language processing [73], [74], information extraction [75], text mining [76], and information retrieval [77]. Event Detection techniques are broadly classified as supervised, un-supervised or semi-supervised. Further the classification is carried out into 4 categories namely Statistical, Probabilistic, Machine Learning and Composite Techniques.

2.3.1 Statistical Approaches

Sakaki et al. [78] proposed earthquake event detection in real-time by implementing a dynamic classifier for analyzing tweets semantically. In this study, each user is viewed as a sensor, and tweets serve as the sensor data along with the user's location and timestamp. The probability of exponential distribution density function, that deals with time-series data, was used for event detection. The location of tweets is estimated using Bayesian filters, such as Kalman and particle filters.

Other statistical approaches, exponential weighted moving average (EWMA), and cumulative sum (CUSUM) were studied for events such as concept drift detection in data streams.

2.3.2 Probabilistic Approaches

In addition to the statistical-based methods, a lot of other work addresses event detection in various ways. The probabilistic models have developed systems that are primarily based on non-parametric Bayesian methods [79], [80], [81]. Additionally, these methods require more computational resources than alternative methods, which essentially make them even less appropriate for processing large amounts of data, particularly in an online environment. In contrast, probability - based approaches have a natural probabilistic interpretation and can present data to the user in a more structured way. For instance, Ahmed et al. [80] presented a system that can support structured browsing and creation of storylines from a stream of headlines. Masud et al. [82] proposed an essential component of novel class detection that is an adaptive threshold for concept drift detection. Additionally, effective approach leveraging discrete Gini coefficient to detect novel classes.

A novel method for the identification of abnormal activity has been suggested by Alarfaz et al. [83] . To detect and follow the features, they used the Kalman filter [84] and the Gaussian Mixture Model (GMM) [85].

2.3.3 Machine Learning Approaches

Spanos et al. [86] presented a method for identification of events in IoT devices installed in smart homes that was supported by statistical analysis and machine learning methods. They used a process called sniffing and collection to gather the sensed traffic data that was transmitted between gateways and sensor devices. Further, using a set of statistically predefined features, they attempted to categorize the data into various behavioral frameworks. Later, based on a predetermined threshold, these frameworks were used to identify events that deviate from normal outcome.

Shukla et al. [87] proposed a scalable approach for event detection in streaming data. They designed a three-stage technique. In the first stage, the data streams were passed through noise filter component and then the dimensionality reduction process using PCA in offline mode. To reduce the impact of data distribution, in the second stage an offline segmentation process was carried out on the time series data. Long Short-Term Memory (LSTM) neural network trained from known dataset using the ARIMA model for time series data analysis was implemented in the third stage. Furthermore, they used m-estimator for statistical analysis, and the output entered in the event detector module was used to detect events in online mode.

Alomari et al. [88] outlines a technique for automatically detecting events related to traffic from twitter posts in the Saudi dialect using big data and machine learning approaches. First, to categorize tweets into relevant and irrelevant, a classifier was

trained using three machine learning algorithms: Naive Bayes, Support Vector Machine, and logistic regression. Additional classifiers [89][90] were trained to recognize a variety of events, such as fire, weather, social events, roadwork, road closures, road damage, accident, etc.

Walther et al. [91] proposed Geo-spatial event detection technique in twitter stream for determining whether the geospatial clusters contain actual events. To achieve this objective, machine learning algorithms (Naive Bayes, Multilayer Perceptron, and Prune C4.5) were used. According to the individual tweet ranking score in descending order, the detected events (candidate clusters) were shown on a map with their locations in real-time.

2.4 Concept Drift Detection Techniques

There are various concept drift detection algorithms proposed in literature, that may be divided into three classes: ensemble based, single learner based, drift detectors based.

2.4.1 Ensemble Based Techniques

By adjusting their aggregation techniques, ensembles offer a mechanism to adapt to the changes. Ensembles that are modular can adapt to change by altering their structure, retraining ensemble members, or updating the decision-making procedures. To produce the final result, single classifier decisions are often aggregated using a voting mechanism. The aggregate output is somewhat more reliable than that of an individual expert [92].

In the literature, a variety of ensemble-based algorithms that deal with concept drifts have been described [93]–[96][97][98]. The ensemble based systems for handling

concept drift can operate in one of main methods, depending on how the data received is processed: block-based mode or online mode.

a) Block-based Mode

Block based ensembles, also known as chunk-based techniques handle the instances in blocks of data. A fixed or variable block size is established for a specific technique, which may necessitate several iterations over the training instances [99], [100]. Some of these are described below:

Streaming Ensemble Algorithm [101]*:* It is one of the earliest drift handling technique that is based on chunk-based ensembles. Each time a batch or chunk of instances enters the algorithm, a new learner is developed and added to the ensemble. An existing learner with low quality score is replaced by a new learner. Accuracy and learner variety are taken into account while determining the quality scores. Final ensemble predictions are made via simple majority voting. However, the SEA algorithm might provide out-of-date concepts in some instances where older learners outperform the newer ones.

Accuracy Weighted Ensemble [102]: AWE is a unique algorithm that uses ensemble reconstruction similar to SEA. It is based on the idea of giving weights to the underlying learners based on how well they predicted outcomes from a recent chunk of instances. Every learner is given a unique weighting function based on mean square error. Weight-based pruning is used to maintain ensemble size, and with every iteration a newly trained learner on the most recent chunk is added. A new learner added uses k-cross validation for evaluation, which has a high computational cost. However, AWE is one of the most effective technique for larger streams and works well for recurring concepts.

Accuracy Updated Ensemble [103]: This approach permits incremental modifications to its learners with each incoming chunk. New weighting algorithms and accuracy-based pruning contribute in outperforming competitors in computational cost. AUE2 was proposed for handling both sudden and gradual drifts. For both new and existing learners, this method applies a different weighing function. Weights are updated as a new batch of instance is evaluated, allowing for iterative concept adaption. Re-weighting facilitates by evolving over changes, allowing high precision to be maintained when addressing gradual drifts.

b) Online Mode

When processing instances one at a time, these ensemble approaches learn incrementally. Each instance arrives here and is handled separately. Applications that deal with a lot of incoming data and have strict memory and processing time requirements are processed online.

Online Bagging

Oza and Russell [104] presented an online bagging algorithm, sometimes referred to as OzaBag, as an online implementation of the well-known bagging technique. A new instance can be reproduced zero, one, or more times in OzaBag during the updation process while the base learner is being trained. 'k' copies of the new incoming instance are updated for each learner in the ensemble. With 'k' Poisson, this procedure derives the value of k from the Poisson distribution [105][106]. The underlying learners contribute their decisions for the output class, which are combined by implementing majority voting technique.

Online Boosting

This is an online ensemble-based approach for incremental learning and drift handling. A fixed set of learners in an ensemble are updated with each new instance, and the ensemble is preserved. Each instance starts with a weight, $\lambda = 1$. With this learner, the first learner is updated $k = \text{Poisson}(\lambda)$ times. According to the learners' predictions, instances that are incorrectly labelled are given more weight than ones that are accurately classified. To broaden the scope of the weighting process, more boosting variant techniques such as Online Coordinate Boosting [107] and AdaBoost [108] were also presented.

Other Techniques

In order to handle abrupt and frequent drifts, Roberto et al. [109] suggested a strategy that improved the accuracy of online boosting techniques. By using ADWIN instead of DDM, they looked at the impact of modifying the drift detection approach. In order to identify data drifts, Raquel et al. [110] presented a windowing strategy that compares the data distributions by applying the idea of fading histograms. Some ensemble techniques, such as BLAST [111], temporarily reduce rather than replace the importance of the learners who perform poorly. The concept of model reuse is covered in a technique called CONDOR [96], where learners' weights are adaptively altered in accordance with their performance.

2.4.2 Single-Classifier based Techniques

Single classifier based approaches are well-known in static learning and can be extended to deal with dynamic data streams. They integrate forgetting mechanisms and online learners. Methods for forgetting assist in removing data instances from outdated concept distributions, retaining the models consistent with concept

evolution. The single classifiers that may be adapted to respond to evolving concepts includes Neural Networks, Hoeffding Tree, Naive Bayes, etc.

- Neural Networks

  Neural networks work by using neurons that are linked to one another and given a weight for each link to learn. Each neuron's output is described by its activation function [112]. These networks are utilized in stream-based input processing in addition to static settings. Neuron weights are updated almost continuously because each instance is only seen once, making them suitable for stream management. In order to respond to concept drifts, the core networks adapt to the new instances.

- Hoeffding Tree Classifiers

  One of the earliest static learning techniques to use a Hoeffding bound for data streams was Decision Trees. A Hoeffding tree (HT), sometimes known as a Very Fast Decision Tree (VFDT) [113], is a stream-specific incremental technique also known as an anytime decision tree. A small sample is sufficient to determine the best value for the splitting attribute that forms the basis of the VFDT learning process. It is dependent on a mathematical bound value called the Hoeffding bound, which specifies how many instances (observations) are needed to estimate an optimal value of attribute. According to this principle, the true mean (Ɛ) of a random variable with range R must not deviate from the estimated mean after n independent observations (number of instances) by more than a probability of 1 -δ. This value is defined by Equation (2.1)

$$\text{Ɛ} = \sqrt{\frac{R^2 ln(1/\delta)}{2n}} \qquad (2.1)$$

Since VFDT was created for stationary data streams, it initially lacked a forgetting mechanism. Hulten et al. [114] introduced Concept-adapting Very Fast Decision Tree (CVFDT), an enhanced version of VFDT that addressed the issue of fluctuating data streams. Most data stream algorithms use Hoeffding trees as its formative learners to deal with concept drifting environments [115].

- Naïve Bayes

  Naive Bayes is a learner based on bayesian prediction that makes a naive claim that all data points are distinct. It calculates the class-conditional probability for each new instance based on the Bayes theorem. It is a fundamental learner with negligible computing overhead during learning. A forgetting mechanism is introduced to the learner and is paired with a sliding window to eliminate outdated instances in order to address concept drifting issues. Decision trees used for data streams frequently include one Naive Bayes learner [116].

From the literature review mentioned above, it can be concluded that there are several flaws with the current techniques, which have been triggered by the reasons listed below. The data streams are temporal however, various existing techniques do not take into consideration the temporal aspect of data hence, may not be able to track events in real-time. To address the above issues, a scalable approach for event detection has been proposed in Section 4.4 of Chapter 4 that considers supervised, semi-supervised and unsupervised nature of classification approach.

In conclusion, despite much research in the field of event detection in streaming data, increasing effectiveness, dependability, and performance in different domains

such as rumor detection, abnormal events in weather data and concept drifts is need of an hour.

2.5 Chapter Summary

In this chapter an overview of application areas in the field of event detection in data streams is discussed. Also the detailed discussion on data stream transformation frameworks is conducted. Furthermore, various event detection techniques based on statistical, probabilistic and machine learning approaches are discussed. Finally, a brief survey on concept drift detection techniques is done. In the next chapter, research gaps in existing techniques based on the literature survey will be identified. Further based on the research gaps, various research objectives will be formulated.

Chapter 3

PROBLEM FORMULATION

This chapter highlights the research gaps discovered while reviewing the literature. These gaps recognize the challenges in the existing event detection techniques. They are determined to develop novel approaches for the detection of anomalous events in streaming datasets. Also, there is a need for an approach capable of event and drift detection in the data streams. There is a rising demand for an approach that fits in the memory space of a single system and is highly scalable. Four research objectives are developed and addressed in various chapters of the thesis based on the research gaps identified in this chapter.

3.1 Research Gaps

Following a thorough review of the literature, there were several research gaps, as mentioned below:

- The tremendous data growth due to the emergence of many new IoT devices made existing stream processing techniques incapable of handling data streams efficiently. Therefore, a need for an intelligent framework arises which is capable of accessing, analyzing and representing valuable insights from large-scale real-world streaming data.
- Stream processing needs new computational tools to evaluate diverse datasets into suitable results. It entails data visualization and data analytics of massive datasets. The old-fashioned mining methodologies need to modify themselves as per in-stream processing to give dynamic results.

- Existing techniques for the detection of events in high-dimensional streaming datasets process all the features for finding outliers. However, some of the features have no significance in event detection. Hence, in streaming data, analysis can be performed more efficiently using data reduction techniques.
- Data streams contain temporary data instances that are only in memory for a particular period. However, most event detection techniques compare these current data instances with old data sets stored earlier. Since, instances in data streams are dynamic therefore, it is impossible to store all the data.
- State-of-the-art techniques available for the detection of drifts in streaming data can identify a specific type of drift. It may be gradual drift or abrupt drift thus, there is a need for a unified framework that can detect all kinds of drifts.
- Several methods have been suggested for drift detection, including block-based drift detectors, window-based methods, incremental learner methods, etc. In the majority of techniques, these methods have been applied individually. However, approaches that integrate adaptive methods with drift detectors need to be explored.
- There is a considerable research gap in deciding thresholds for events. The techniques available for deciding static thresholds for event detection are efficient for fixed datasets, but for scalable data, dynamic thresholds must be used that are selected based on values in the data streams.

3.2 Research Objectives

The following are the objectives of the proposed research:

- To study various techniques and tools available for handling big streaming data.

- To design and implement technique(s) for transformation and analysis of input data streams to receive regulated data streams.
- To design and implement efficient technique(s) for detection of events from regulated data streams.
- Testing and validation of the proposed approach using real-world data.

3.3 Research Methodology

A specific methodology was used to achieve our research goals; the workflow of this methodology is shown in Figure 3.1. The following process is used to accomplish each goal:

Objective 1:

An extensive survey of existing approaches for handling streaming data is done to gain valuable insights and a better understanding of the domain. The bibliographic study is conducted considering the following aspects for streaming data analysis:

- Gained understanding of the fundamentals of Streaming Data, its characteristics, usage, inference from streaming events and various tools.
- Gathered knowledge about the existing data stream processing algorithms. Additionally, studied some existing techniques for cleaning, filtering, simple and complex event processing.
- Reviewed various frameworks for data transformation and feature extraction from unstructured data streams.
- Studied some machine learning techniques for drift detection and adaptation in streaming data.

Objective 2:

- A sliding window-based stateful transformation technique is designed, which divides the incoming data streams into micro-batches called Discretized Streams (DStreams).
- Developed an improved Valence Aware Dictionary and Sentiment Reasoner (VADER) classifier for lexicon and sentiment analysis of tweets. It accurately classifies the tweets into four categories and detects whether they are rumorous or non-rumorous. The technique detects rumor events on the top of Apache Spark framework so that the data streams are processed using in-memory processing frameworks.

Objective 3:

- The implementation and experimentation was carried out on Dell machine simply by setting up Jupyter Notebooks with Python libraries. Spark Streaming Framework is configured on Eclipse IDE as well as VM to compare the performance of the proposed approach on different machines.
- The proposed approaches are implemented using Python and Scala programming languages on the top of Spark.
- Real-world and artificial datasets are used to implement the proposed methods which are extracted from Twitter, PubNub sensors, available APIs to fetch the data streams.

Objective 4:

- The efficiency of the proposed approaches is verified by comparing their performance with existing state-of-the-art techniques.

- Tested the techniques using different datasets and performance is evaluated in terms of performance metrics such as Accuracy, F1-score, Precision, Recall, etc.
- The proposed techniques are validated by considering data streams from different sources and application domains such as weather datasets, tweets, SPAM email datasets, etc.

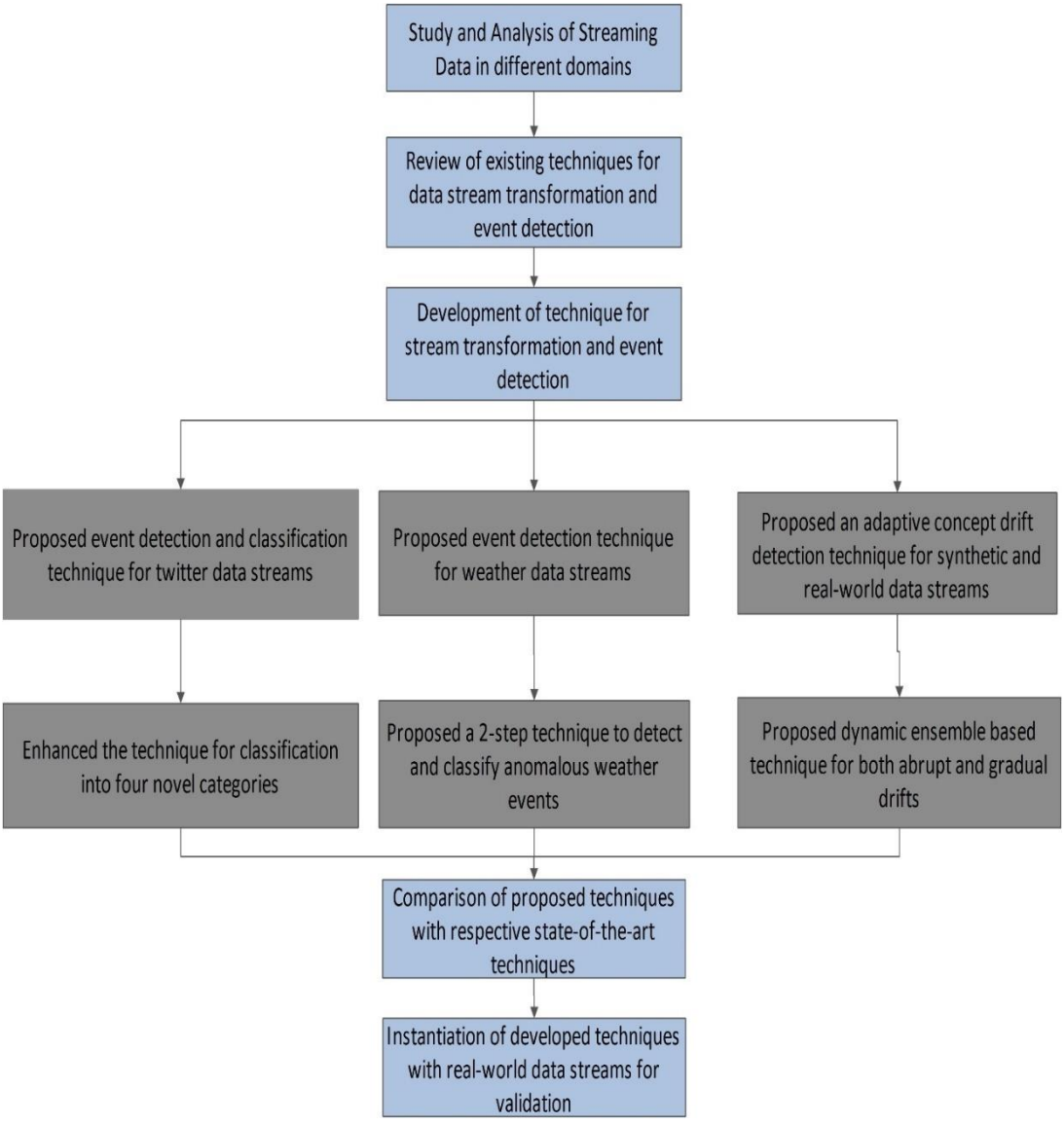


Figure 3.1: Research Methodology

3.4 Chapter Summary

In this chapter, different research gaps are highlighted based on the literature survey. To address some of the gaps in the existing research, few research objectives are formulated. Further, to achieve each objective a research methodology is designed which is discussed in the last section of the chapter. In the next chapter, the proposed approach for transformation of data streams for event detection in social media data that is Twitter will be discussed in detail.

Chapter 4

TRANSFORMATION AND ANALYSIS OF INPUT DATA STREAMS

The objective of this chapter is to discuss an approach for transformation and analysis of input data streams. A transformation and classification technique for rumored events on social media during the pandemic is proposed in this chapter. This work is accomplished by extracting diffused information from Covid-19 tweets. Further, the extracted tweets are pre-processed and transformed into explainable statements and finally, an objective of a lexicon-analysis-based approach for the classification of rumor events is achieved. The experimental results illustrate that the proposed classifier is extremely effective in identifying and categorizing rumor tweets that appear under critical circumstances.

4.1 Background

With the ubiquitous use of smartphones, tablets, and other handheld devices, the rise of social media and microblogging services which first served as a growing means to disseminate information and news on the internet has evolved into the widely used open-source data. Social media is accessible 24/7 and has become the primary medium for people to express their thoughts and respond to the emerging trends. People frequently broadcast the topics of their immediate environment or present circumstances. As a result, social media can play a significant part in the identification and analysis of real-world events. Twitter is a well-known social networking website for microblogging which allows 140-character texts to be posted, with roughly 340 million tweets posted by over 300 million monthly active

users [117]. Twitter provides geo-locational data tags that come from proximity sensor devices, such as mobile phones to target widespread audience around the globe. Twitter data is incredibly beneficial for a wide range of fields, from market research to current trend analysis.

Microblogs posted on Twitter and other social media platforms have been used to track critical events such as earthquakes, natural disasters, sociopolitical commentary, traffic, etc. in the past. However, using social media to detect such events is still a viable and growing research area. There are numerous definitions of the concept of an 'event' in streaming data literature. An event can be defined as a "significant happening at a particular time and location". This definition can be extended by saying that "when a group of people discuss about a happening which is different from regular levels of conversations about a topic". In other words, it can also be treated as "trending topic". In some studies events are characterized as activities that take place at a given location and time and draw attention immediately [118]. In this study, we adopt the definition of an event from Topic Detection and Tracking (TDT) research field, which employs textual document data mining approaches [119].

In this chapter, the ultimate goal of event detection is to identify swinging rumors in social media posts during the times of pandemic in the world. In 2020, the world was talking about Covid-19, and millions lost their lives to this widespread disease. The scientists and doctors were rigorously performing experiments to discover medicine for this disease. On the other side, common people were trying to find remedial treatments at home for the cure of this disease. In such a situation, any breaking news needs to be verified to stop the dissemination of false information because it serves as the primary source of rumors. This type of information can be detrimental

to the society if it turns out to be untrue in the future. Hence, a research gap is discovered in the existing literature that most event detection techniques do not consider the relationship among data attributes and are not scalable to streaming data. Additionally, this calls for a significant amount of manual labor of referring debunking services for numerous tweets every second. A system to record, verify, and classify the statement as true or false is necessary since there are no restrictions on the amount of information that any user can post on social media.

To address the above issues, we propose an approach to handle the aforementioned challenges by:

- Collection and transformation of tweet feature set such as profile, spread, temporal, and microblog features.
- Detection and validation of tweets being classified as a rumor.
- Find all instances of events (also known as "bursts") in a stream of microblog posts with high accuracy. A burst can be distinguished by an increase in word occurrences or a rise in message cluster sizes, and it may indicate anything from breaking news to natural disasters.
- Rumor classification into various types, such as amalgamated, unauthoritative and exaggerated.

Some existing supervised and unsupervised methods for detection and classification of rumors from tweets taken as the background of the proposed approach are discussed as follows:

4.1.1 State-of-the-art Approaches for Rumor Detection

The famous Chinese social media tool utilized by Yang et al. [120] for their research is Sina Weibo and the features such as client program and event location are the two novel characteristics of their work that are not discussed in the existing literature

[120], [121]. Prior studies concentrated on other aspects such as propagation, account, and content-based event detection. However, based on specific subsets of the characteristics above, they trained and implemented a Support Vector Machine (SVM) classifier. The remaining three elements yielded a median accuracy score of 72.5%. By including two different characteristics in the trial, the median accuracy was increased to 77%. Dayani et al. [122] contributed more data instances of Twitter accounts to the dataset [123]. Low estimation accuracy with KNN was achieved by combining user-based features with a Naive Bayes (NB) classifier and content-based features with a k-nearest neighbor (KNN). The researchers discovered that pre-processing improved NB's ability to recognize rumor elements. With both supported and disputed rumor tweets, precision and correctness were up to 86%, while rumor information that was questioned showed 74% accuracy. The findings of this study, in contrast to other studies, as discussed in [124], was that the identification of rumor tweets had no link with the user-based attributes.

Additionally, Reis et al. [125] conducted research on the issue of false and misleading news on social media by analyzing data files of 2,282 news content uploads from BuzzFeed related to the 2016 US elections. The evaluation and grouping of 141 textual characteristics were categorized as sentence level, lexical, psycholinguistic, semantic, and subjectivity. Then, for research perspective, a set of characteristics were added. The study included KNN and NB classifiers, random forest (RF), SVM with RBF kernel, and XGB (XGBoost). A comparison between multi-step and single-step classification was done [126]. The authors extended a handful of the characteristics using social and realistic features while working with the public dataset [123]. The models were trained and tested using the Weka platform [127] using the J48 algorithm, a java equivalent of the widely used C4.5

mechanisms. Their analysis revealed that the mono-step strategy for detecting rumors lagged well behind the multi-step approach.

Under the CIVIC project (Intelligent characterization of the reliability of the information connected to COVID-19; http://civic.etsisi.upm.es/) [128], more recent activities related to the identification of rumors and fake news related to the COVID-19 pandemic have been undertaken. Its primary goal is to identify and restrict the spread of false information by using proactive approaches that combine deep learning and natural language processing to combat misinformation.

4.1.2 State-of-the-art Transformation Techniques

Raw data streams are mostly sent to stream processing systems through Kafka pipelines before being consumed in real-time or fed to long-term storage. In the data preparation stage, the initial responsibility is of the stream consumer that is Kafka, a Spark Streaming operation, or a Storm architecture to capture and prepare the data for processing framework. There are a variety of operations to conceptualize the data preparation procedure, and the three most prevalent ones are: filter, extract, and transform.

a) *Filtering*: Limiting the data to be sent to the next stage of a stream processing pipeline is known as filtering. This process eliminates sensitive information that has to be treated carefully or that only a small group of people should see. It is frequently used for schema matching and maintaining data quality. Lastly, routing a raw stream to stream processing engines for additional analysis is sometimes referred to as filtering [129].

b) *Extraction*: Unstructured or partially structured records or fields are frequently present in raw data streams. The requirement is to extract specific structured fields from these raw records before doing structured data analysis

operations like aggregation or model development. Regular expressions and libraries like Grok are frequently used for parsing and extraction. Re-assembling records using a serialization library that support schemas like Apache Avro is beneficial after extraction [130]. These structured records generate a new stream for downstream consumers to ingest.

c) *Transformation*: It may still be necessary to use projection or flattening operations to reconstruct a stream after it has been converted into structured records. The transformations are most frequently used to change records having a specific source schema into records with a specified destination schema. This may be used to write a stream to long-term storage with a particular schema for offline analysis or to combine many streams into a single stream with a common schema.

4.2 Transformations for Streaming Data

The process of changing the configuration, structure, or characteristics of data into a new format is known as data transformation. Additionally, the growing complexity of the real-world necessitates straight-forward and adaptable solutions to fuel commercial rivalry. An Extract-Transform-Load (ETL) process involves extraction, cleaning, transformation and loading of data streams from the source in order to integrate and build a unified source for Business Intelligence (BI). This research primarily focuses on data stream transformation that is the secondary step of ETL.

Raw Twitter streams are captured using Twitter Streaming API and then filtered considering the main characteristics such as profile, spread, microblog and temporal features. These feature categories are further described as:

a) Profile Features

This category mainly contains information about a particular user's account. Some of these are Username, Activity, Posts, Followers, Following, etc.

b) Spread Features

This category comprises of information about the spread of a particular post. For instance, the features such as Reply, Forwarding, Spread_Intensity.

c) Microblog Features

This category discusses the features of a particular post such as Length, Image, Hashtag, URL, News_C, URL_Value, Content.

d) Temporal Features

This feature indicates whether the posting date falls on a workday. Weekdays are represented by 1 and holidays by 0.

Once all the tweets are captured along with their features. Some transformation techniques are applied to real-time streaming tweets so that the transformed tweets are fed to the stream processing engine (Spark) for further processing. Some of these are discussed below:

Stateful Transformation

In twitter data streams, a state must be maintained to track user activity so that rumor spread can be identified. The user activity is tracked through "user session" as a persistent state and continuously updates the change in the state based on user's actions. These type of transformations are performed in Apache Spark using an operation known as mapWithState.

The history of user activity for each user is preserved. In order to translate this into code, initially, the state data structure is created and then the method to modify the state is proposed as depicted in code snippet given in Figure 4.1.

‘mapGroupsWithState’ applies the stateful transformation to the streaming data and maintains track of each group's state. A GroupState specification (group_state_spec) is also used to regulate the state's timeout and other settings as shown in Figure 4.2.

```
def stateUpdate(
userId: UserId,
newData: UserAction,
stateData: State[UserSession]): UserModel = {

val currentSession = stateData.get() // Get current session data
val newSession = ...       // Compute updated session using newData
stateData.update(newSession) // Update session data

val userModel = ... // Compute model using updatedSession
return userModel // Send model downstream
}
```

Figure 4.1: Code snippet for method to modify the state of a data chunk in a stream

```
result_df = parsed_streaming_df \
    .groupByKey("key") \
    .mapGroupsWithState(
        updateAcrossBatches=state_update_function,
        outputMode="update"  # Specify the output mode based on your require
    )

# Start the streaming query
query = result_df.writeStream \
    .outputMode("update") \
    .format("console") \
    .start()

query.awaitTermination()
```

Figure 4.2: Stateful transformation using mapGroupsWithState

In stateful transformations, a window-based tracking is used where all the incoming streaming data is transformed into key-value pairs DStream[(K, V)], so that the states of the key can be tracked in all batches of data.

Using the *findspark* library, Apache Spark is detected on the local workstation before importing the required packages from pySpark. A module named pyspark.sql is used

to save the top 10 tweets in a temporary SQL table after we clean off the tweets that start with #. In this work, a named tuple object to save the hashtag counts is developed. Secondly, an array of tokenized tweets is created using flatmap(). Lambda functions are used because they utilize less memory and execute more quickly. Then, the tweets that don't start with # are eliminated. The foreachRDD() function of PySpark is used for facilitating rapid RDD processing. Each RDD is subject to be transformed into a dataframe.

The tweets are pre-processed to filter out only the clean text of the tweet. In every micro-batch, the received tweets are split at the string t_end. The tweet's text is cleaned by removing the blank rows and applying regular expressions.

There are some modules implemented for Twitter streams, such as Hashtag Analysis, Social Network Graph analysis of all the users posting the trending hashtags and region-wise analysis of the trending hashtags. All the functions mentioned above take the transformed tweets and produce valuable results through Spark Streaming Engine. In this work, we have used the combination of both stateful and stateless transformations on social media data streams to achieve favorable outcomes.

4.3 Event detection from streaming data

This section provides detailed explanation of the three pivotal event detection methodologies comprising of keyword-oriented, clustering-oriented and a hybrid of keyword and clustering-based technique. All three methods begin with a preprocessing stage utilizing the Stanford NLP parser to improve the performance metrics such as accuracy and consider geolocation while processing the data. Data streams are processed through sliding windows of defined time intervals known as rounds. In each iteration of a round, the stream is processed and a random state is produced. The historical state is maintained at every iteration, and a state transition

occurs between subsequent rounds. An event is said to have occurred when a state is reached in accordance with a change between two subsequent states.

4.3.1 Keyword-Oriented Event Detection

This technique intends to identify the widespread terms that exhibit high occurrence exclusively in specific rounds. Highly frequent words used in several tweets or the words that are rarely used are eliminated in this algorithm. The primary interest is in the identification of unexpected rise in word occurrences as compared to the previous round. Such type of *bursty* words are said to be expressive of an event. This technique further involves three operations in each iteration: wordcount, word_weight and event detection. Therefore, at the final stage of every iteration, a list of event keywords is collected.

4.3.2 Clustering-Oriented Event Detection

The core principle of the clustering-oriented approach is that an event corresponds to a group of tweets with a rapid growth rate. Each iteration is processed one at a time, and the output clusters are analyzed for event detection, similar to the keyword-based method. This approach begins with one cluster and concludes with an infinite number of clusters, creating Twitter clusters hierarchically. This technique further involves two operations that is cluster formation and event detection.

4.3.3 Hybrid Technique

The final technique combines the first two to improve the effectiveness of clustering by pre-elimination using word TF-IDF values and filtering out tweets that do not contain bursty phrases. The methods utilized in the

keyword-oriented event detection approach are first used to identify bursty terms, and then clustering is performed on tweets containing those keywords. The hybrid technique, like the earlier two techniques also involves data processing using three operations namely: Tweet filtering, Clustering and event detection using clusters.

4.4 Proposed Approach for Event Detection: *A case study of rumor detection during COVID-19 Pandemic*

The proposed work is divided into 4 phases as shown in Figure 4.3. The detailed explanation of each phase is discussed as follows:

i) *Phase 1: Data Extraction*

Studies have identified that Twitter is the primary source for all popular events because of its self-correcting open-source features, which allow people to post opinions, facts, and theories. Being the source of all news, Twitter was initially created to unite all reliable sources on a single searchable platform where users could learn about current events. Twitter was created as a social media platform for microblogging, but it eventually integrated the more sophisticated functionality of "re-tweeting," or the quicker sharing of the published microblogs and freely accessible to all on the globe. This accelerated not just the transmission of genuine information, but also the propagation of false rumors.

Twitter offers REST APIs and Streaming APIs that provide a large chunk of public user tweets. Information requests on tweets, people, locations, or other elements of Twitter data are sent via the REST API, and replies are often returned in JSON or XML format. The Streaming API offers a stream of Twitter data that may be accumulated according to a particular condition.

In this research work, the Twitter Streaming API implemented in Python library Tweepy [131] is employed. The tweets are filtered using keywords related to "Covid-19 virus" and no specific location is targeted. The tweet information includes Twitter post id, the text of the post, the rumor label (True, False, or Unverified), the amount of likes, retweets, and comments, as well as user reactions across time, response times, and user response instances. For experiments, this study primarily focuses on content from Twitter posts and the reactions each post received from users within a specified time frame. The proposed Algorithm 4.1 gathers subject tweets from Twitter based on popular hashtags and then forms the clusters by topic.

ii) Phase 2: Data Pre-processing

Pre-processing includes several steps, such as data cleaning, polarity detection, determining opinions, and merging datasets. The first step here is to use the public opinions to derive several techniques, such as labeling the parts of speech, lemmatization, stemming, eliminating stop words, and locating the keywords. The second step includes data preparation and exploratory data analysis. During the process of data cleaning to identify useful features, raw tweets are unable to produce objective outcomes in sentiment prediction. The major obstacles are #tags, @mentions, URLs, and stop words in tweets. The #tags, @mentions, and URLs from the tweets were substituted using regular expressions for sentiment analysis of the opinion on COVID-19. The NLTK library in Python handles the stop words.

The TextBlob module in Python is employed to determine the polarity value of every tweet in the dataset. Using TextBlob, the cleaned tweets from the earlier phase are submitted for metric evaluation using ML models, and a general polarity score for every tweet is obtained. The word types such as trigrams, bigrams, and unigrams, impacts the polarity score

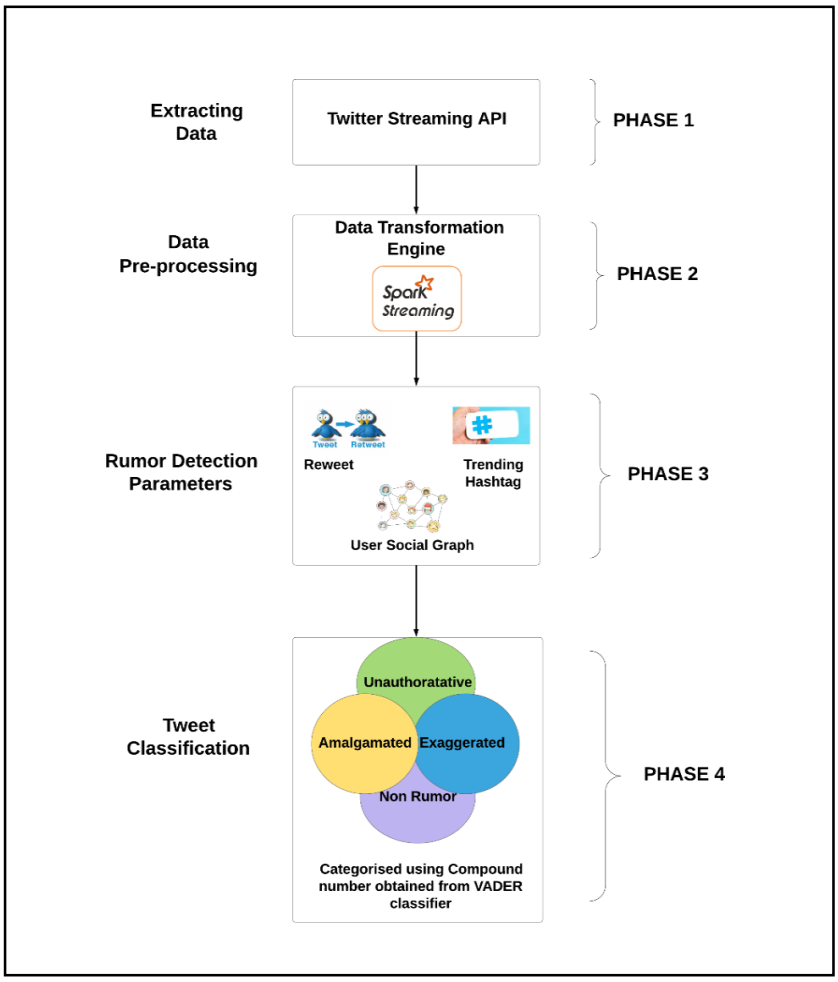


Figure 4.3: Workflow of Proposed Methodology

Algorithm 4.1 collectTopicStreamSet(TrendSet)

```
Input:    HashtagSet consisting of Trending Hashtags(Hashtags) with
count
Output:   Topic wise Tweet Set
  for each Hashtag ∈ HashtagSet do
1 TweetStream ← getTweets(Hashtag)                  //retrieve tweets by passing Hashtag
2     if ambiguousTweets(TweetStream) then
3     TopicStream ← getTopic(TweetStream)//retrieve distinct Topics of Hashtag
4          for each Topic ∈ TopicStream do
5             TopicStreamSet ← getTweets(Topic)      //retrieve tweets of a particular topic
6             TopicStreamSet has structure T (Topic, tweets)
7             end for
8     end if
9           end for
10    return TopicStreamSet
```

Window Transformation

The window transformation module defines window-based aggregations of column attributes in data streams. In this module, the incoming micro-batches are aggregated in time intervals, that is every 'x' seconds. Further, the computations on such batches are performed using slide intervals. Several aggregations are performed based on data or time frames (SQL OVER clause) with the Expression Builder, including LEAD, LAG, NTILE, CUMEDIST, RANK, and others. The output is a new field or column after these aggregations.

For instance, to keep a streaming average of 20 seconds for some data, a batch interval of five seconds and a window length of 20 seconds is required. In this research, a batch of 1 second of data, a window interval of 3 seconds, and a sliding interval of 2 seconds is initialized. This means that every 2 seconds computation of result is done by looking back for up to 3 seconds worth of data as shown in Figure 4.4. The batch interval used in this work is set up as a part of

the streaming context, the discretized stream (DStream) which comprises chunks of 1 second of Resilient Distributed Dataset (RDD). A new RDD is obtained, and the historical backdrop of the RDDs which existed in the window are captured.

Apache Spark does not support the idea of a tumbling window; instead, all windows are sliding.

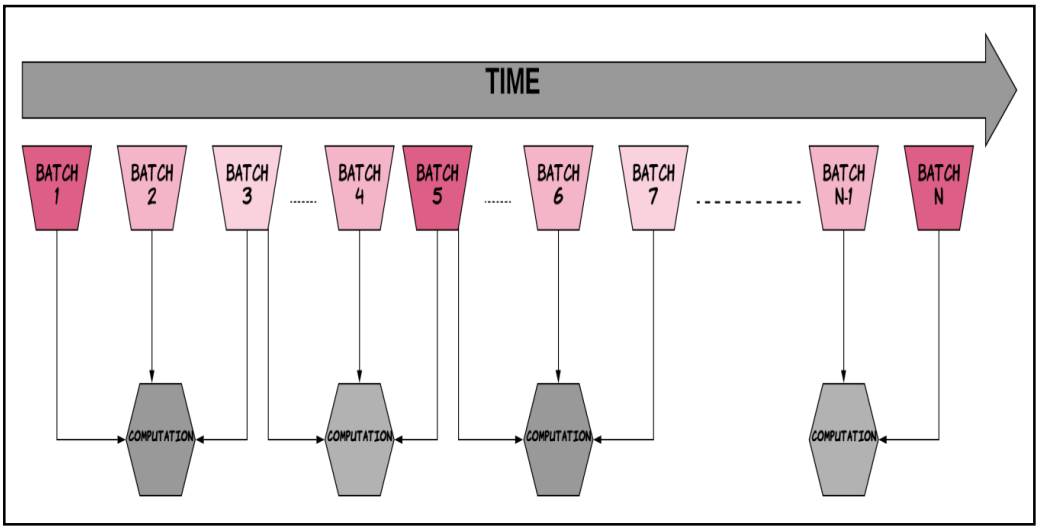


Figure 4.4 Streaming Data Window Transformation using Spark

iii) *Phase 3: Setting up Rumor Detection Parameters*

Following the data ingestion, preprocessing, and transformation of the streaming data, the below factors are considered before a rumor is deemed to exist:

a) Re-Tweets

Techniques for detecting rumors are based on propagation criteria and concentrate on the features of how rumors spread, including how users of social media platforms use retweets and comments. These comments and retweets represent users' opinions. The comments and retweets typically appear like a tree structure and conversational threads are made up of rumors and retweets. Several studies [132], [133] investigated the user answers and

retweets tree structure to assess the credibility of rumors using the propagation path structure. However, the subject of user reactions is not focused upon much in the existing work.

b) User Social Network Analysis

By following other users, a person creates a social network, which they share with other people. Therefore, it is crucial to determine if a rumor is being propagated by a small number of users or a vast network of people. This is accomplished by gathering a user's followers' information via retweets of their tweets. The followers of the individual who retweeted anything are further analyzed and classified into verified and non-verified users. The user social network graph extracted during the implementation of this module is shown in Figure 4.11 and Figure 4.12.

c) Trending Hashtags

In addition to the textual data, other significant content aspects are postag, emotion, and particular hashtags like "#COVID19" and "#Vaccine". Hashtags act as concise summary of the tweet's data [134]. Within the collection of labeled data, hashtags are extracted. The hashtag feature dimension of tweets is given a value of 0 for those without hashtags, and a value of 1 for those with at least one hashtag. Furthermore, each observed hashtag is incorporated as a feature dimension, making it possible to recognize tweets containing relatively similar hashtags. A straight-forward heuristic for compound hashtags is introduced in the case of an uppercase letter that appears amid a hashtag word. The idea is to divide the word before the uppercase letter making it split into two meaningful words to get meaningful insights. For instance, hashtags such as “#coronavirusWuhan” are separated into “coronavirus” and “Wuhan”.

Further, by using hashtag feature dimension, compound and separated hashtags are modeled.

iv) Phase 4: Tweet Classification

Real-time classification of a tweet into rumor and non-rumor is quite challenging. The existing literature reveals that conversations having neutral sentiment words and text with high distribution of punctuation is typically vulnerable to fundamental frame of reference. This study intends to evaluate the characteristics of rumors discovered on Twitter by considering the factors such as tweet length and frequency, tweet circulation via Twitter, and tweet phrase pattern. The primary objective of this research is to comprehend the lexical characteristics that are utilized to classify distinct rumor types. An unsupervised approach is used to categorize tweets based on the determined properties. The technique for distinguishing rumor-related tweets from non-rumor-related tweets includes redefining attributes like tweet-type and tweet-text using feature engineering, manually labelling tweet sentiments, and sentiment analysis. Furthermore, machine learning approaches are required for self-regulated rumor detection and classification. The selection of the tweet set is based on the criteria listed in Table 4.1 for the parameters that were taken into consideration.

Table 4.1: Criteria used for the selection of tweets

Sr. No.	Criteria	Explanation
1.	Rumor Integrity	The metric computes tweets based on it being either news or rumor. It is highly estimated that the dataset consists of mostly news.
2.	Contextual Integrity	This metric computes the number of contexts in a dataset, which varies depending on the search query used to build the dataset.
3.	Number of Tweets	The number of tweets is required to be sufficient for experimentation. Having an insufficient number of tweets (<100) may halt the outcome extracted from the dataset.
4.	Number of Unique Tweets	The number of unique tweets must be more than 50.

4.5 Experimental Analysis

4.5.1 Tweet Analysis

The data is collected via Twitter's API service. There are two different types of APIs accessible for the collection of dataset: REST API and stream API. The frequency of gathering data is the only key difference between the two. REST API is used for one-time data collection, whereas streaming API is used for ongoing data collection over a pre-determined duration. With the use of Twitter's streaming API, tweets containing the words or hashtags "Covid-19", "Corona Virus", "Corona", "COVID", "covid19," and "sarscov2" from March 1, 2020, till December 2020 were gathered for this study. The tweets are collected from users of 182 different countries. The non-english tweets in the tweet collection are eliminated, leaving behind 8,430,793 English tweets. Thousands of verified accounts tweet about the coronavirus, while there are more than 20,000 tweets from regular accounts. The rise in new account creation during the coronavirus epidemic demonstrates how social isolation affects people's desire to find others to interact and find a source of news.

During the process of data cleaning for identification of useful features, the raw tweets are unable to produce objective results for sentiment rumor detection. The major barriers are #tags, @mentions, URLs, emoticons and stop words which were substituted using regular expressions. Stop words are processed with the Python NLTK package.

Feature extraction and data pre-processing is done using the Apache Spark streaming framework. Word2Vec is used as an embedding model for the feature extraction process. It allows learning the jargon words included in the tweets to produce the

similar representation of terms in context. The word embedding model is specially trained to overcome the noise in tweets. The classification step made use of a VADER classifier.

4.5.2 Hashtag Analysis

To discover popular topics for event detection, tweets with similar content are grouped in a process known as clustering. It is feasible to form a viewpoint on the trending events on Twitter by grouping tweets with similar hashtags. Different hashtags may be used to describe the same occurrence, though. Moreover, grammatical errors might occur when typing a hashtag.

To detect related hashtags, an analysis of word co-occurrence on tweet contents is performed. The effectiveness of event detection is achieved by employing a lexico-semantic extension to hashtags. In this work, hashtag analysis is performed to capture the changing sentiments and trends of people during the pandemic. The word cloud module as visualized in Figure 4.5 is implemented to fetch all the words trending on Twitter in that time period. Trending hashtags as on April 26, 2020 were captured for experimental purpose screenshot of which is shown in Figure 4.6. A network of top 50 trending hashtags as shown in Figure 4.7 was again captured on April 29, 2020 which further proves that the hashtags such as “fake news”, “coronavirus”, “media literacy”, “who can you trust” were among the most popular hashtags that sounded appropriate for this research. Hashtag Analysis is one of the most prominent methods for the detection of rumors. Another module implemented to verify the trending hashtags is Hashtag Network module that leveraged in visualizing the popular words and their evolution as shown in Figure 4.10. For instance, during the pandemic there was a trending hashtag of North Korean leader #

KIMJONGUNDEAD with 214,982 tweets. However, later it was debunked using Snopes.com that declared news as Unconfirmed and the leader was found alive later. This rumorous instance was one of the most critical news as it was concerned with the leader of a particular nation and the proposed work proved to be effective in such as situation.

Figure 4.5: Word cloud for trending tweets

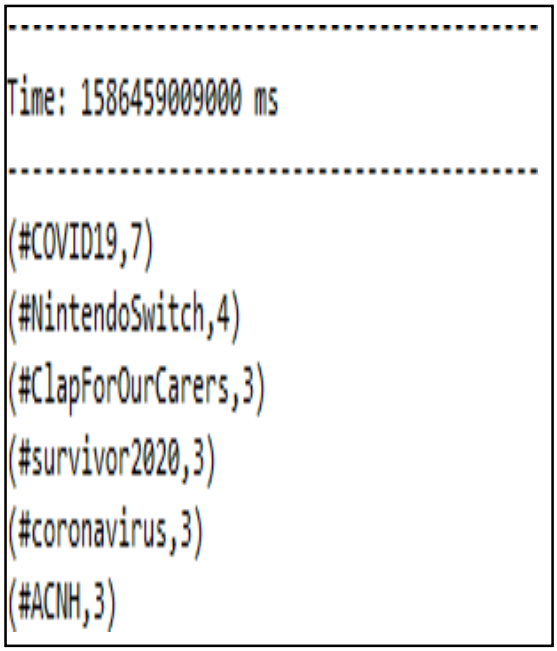


Figure 4.6: Trending hashtags in decreasing order of count

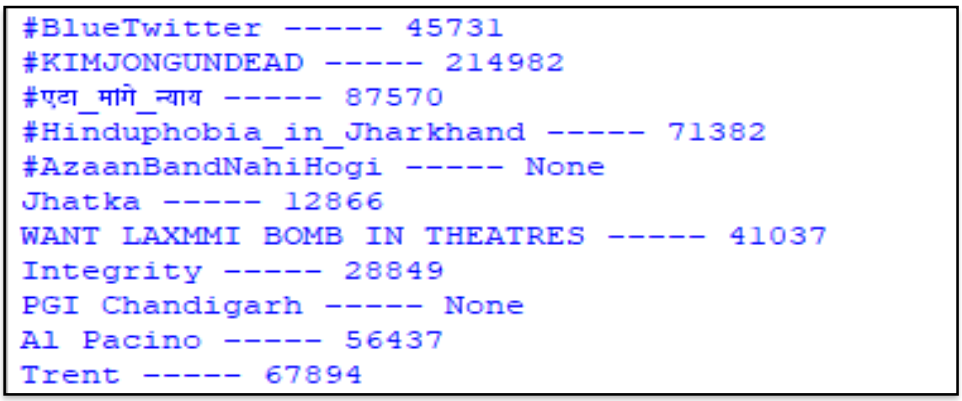


Figure 4.7: Trending Hashtags as on 26th April 2020

Algorithm 4.2: Modified VADER-Sentiment Algorithm for Classification

```
#def senti_scores (self, sentiments)
unauth = 0
amal = 0
exag =0
neu_count = 0
for senti_scores in sentiments:
        if senti_scores > 0 and senti_scores < 0.5:
                unauth += ((senti_scores) + 1)
            else if senti_scores > -1 and senti_scores < -0.5:
                amal += ((senti_scores) - 1)
                else if senti_scores > -0.5 and senti_scores < 0:
                    exag += ((senti_scores) – 0.5)
                  else if senti_scores == 0:
                      neu_count+=1
                  End else if
              End else if
            End else if
         End if
  return unauth, amal, exag, neu_count
End for
Threshold = 0.5
#analyzer = IntensityAnalyzer
text_score = []
for tweet in json_data:
      if ‘text’ in tweet.keys():
          text = tweet[‘text’]
          tweet count+=1
      if text:
          vs = analyzer.polarity scores(text)
      if vs[‘neg’] >= NEG THRESHOLD and vs[‘compound’] < 0:
          return (tweet count, vs, rumor)
      else
          return (tweet count, vs, non rumor)
        End if
      End if
    End if
End for
```

4.5.3 Rumor Classification

A sentiment analysis technique to find the emotions hidden underneath the false tweets on COVID-19 is used. The complete dataset is analyzed to evaluate the VADER classifier, a rule-based approach for broad sentiment analysis implemented

using NLTK package [135] (https://github.com/cjhutto/vaderSentiment). Based on a set of lexical characteristics adjusted particularly to sentiment analysis, this model calculates scores for the valence of sentiment. It is based on conventional lexicons but is enhanced by emotive elements seen on social media, such as capitalization, emoticons (frequently used to indicate intensity), and acronyms (like lol! omg!). The model is selected because it is explicitly designed for contexts resembling microblogs, addressing and demonstrating strong performance (such as text-length limitations and the use of shortened linguistic conventions to convey emotions). The model generates four distinct scores: a final compound score, a negative score, a neutral score, and a positive score. Each word that also exists in the lexicon has its valence scores added together, modified in accordance with the rules, and standardized to lie between -1 and +1. This last metric is beneficial in assessing overall sentiment value throughout the whole sample as well as any potential variations in different forms of misinformation since it offers a single sentiment score for each particular line or tweet. Moreover, the targeted tweets are in the English language to help in comprehension of the tweets in a better way. Hence, after the tweets are filtered in English, an approximate of 90K tweets are targeted from different nations. The comparison of pandemic-hit countries is done with respect to the spread of rumors related to Covid-19 as shown in Figure 4.8. The comparison is further elaborated with respect to the rise in number of Twitter users during the pandemic time and it is clearly visualized in Figure 4.9.

The VADER classifier is enhanced further (Algorithm 4.2) to classify the rumors into 4 categories that is Unauthoritative, Amalgamated, Exaggerated, and Non-Rumor. The tweets that are not from an official news source and have a compound score between 0 and 0.5 are considered unauthoritative. The tweets with a compound

score between -1 and -0.5 are classified as amalgamated tweets since they are a blend of truth and hoax. The tweets with a compound score between -0.5 and 0 are classified as exaggerated because they present the situation worse than it actually is. A tweet is deemed non-rumor if its compound score exceeds 0.5. Table 4.2 lists the classification results for a sample of tweets and it's category distribution into four mentioned categories. Further, the categorization results can be seen in the bar graph shown in Figure 4.13.

Additionally, a platform known as MedISys (http://MedISys.newsbrief.eu) [136] is used to verify medical-related rumors. It is a method for tracking and evaluating the information on global public events. The site was built on the foundation of the Europe Media Monitoring engine (EMM, http://emm.newsbrief.eu), but it was subsequently modified to track health-related origin in addition to conventional media to categorize a wide range of public health hazards. This website offers a feature that effectively addresses news stories in more than 40 diverse languages [137].

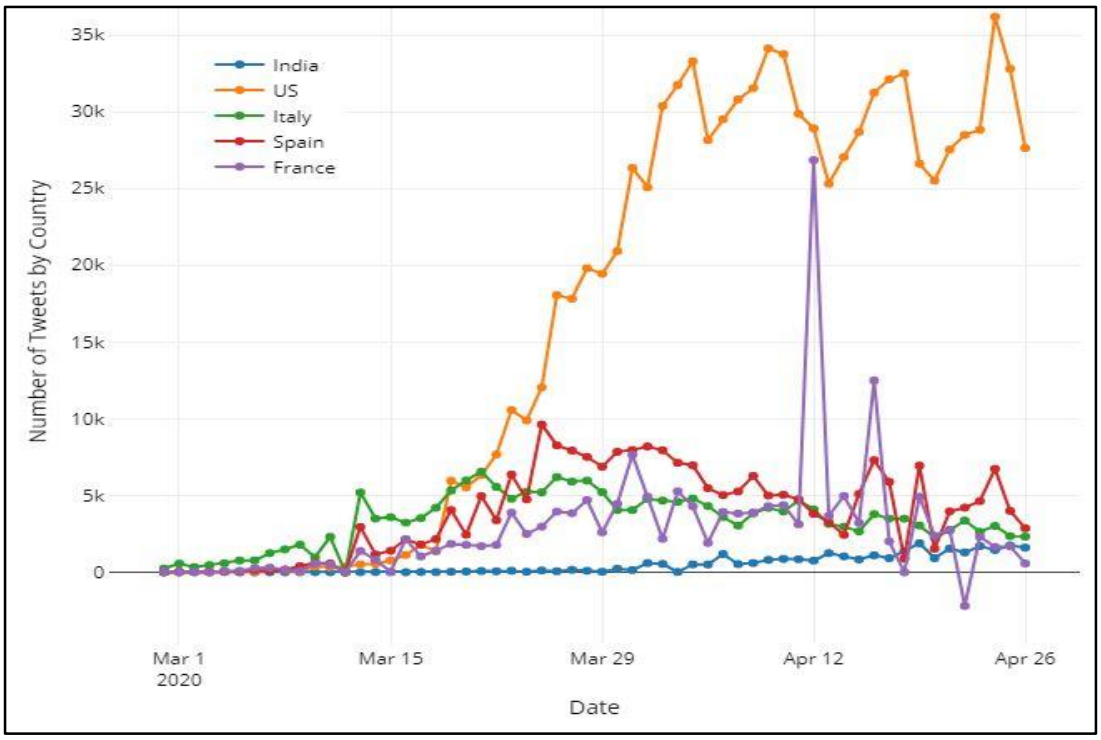


Figure 4.8: Comparison of number of tweets on Corona Virus by highly affected countries in 2020

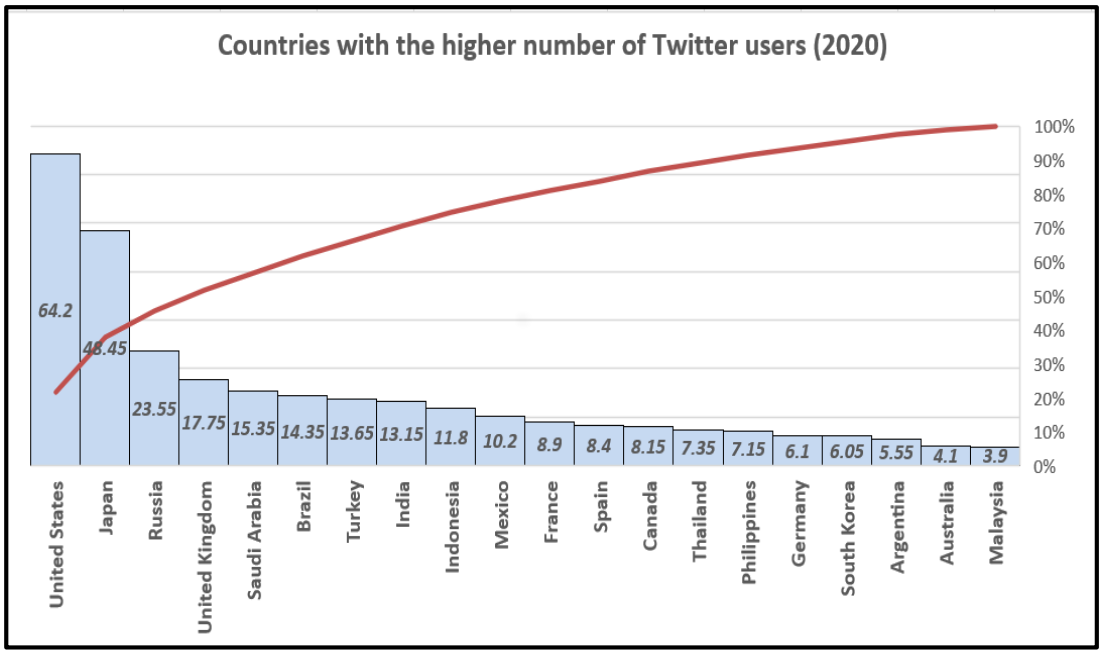


Figure 4.9: Country-wise comparison of the number of Twitter users (January-April, 2020)

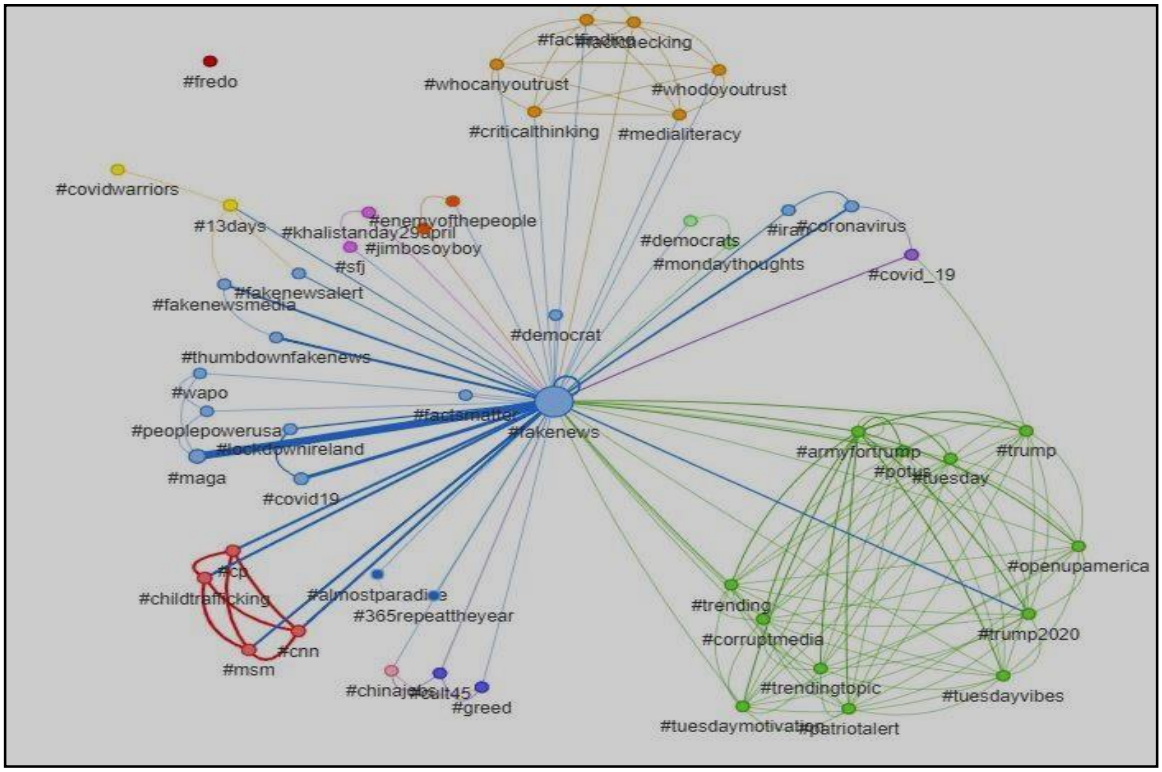


Figure 4.10: Hashtag network of trending hashtags in the world (in the year 2020)

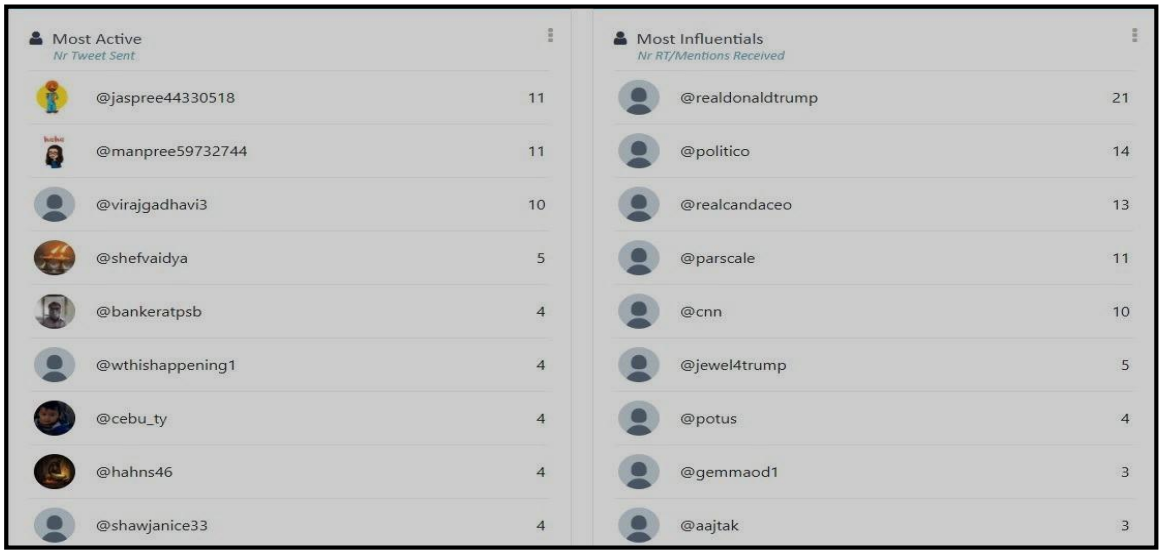


Figure 4.11: Snapshot of Top Users using the trending Hashtag

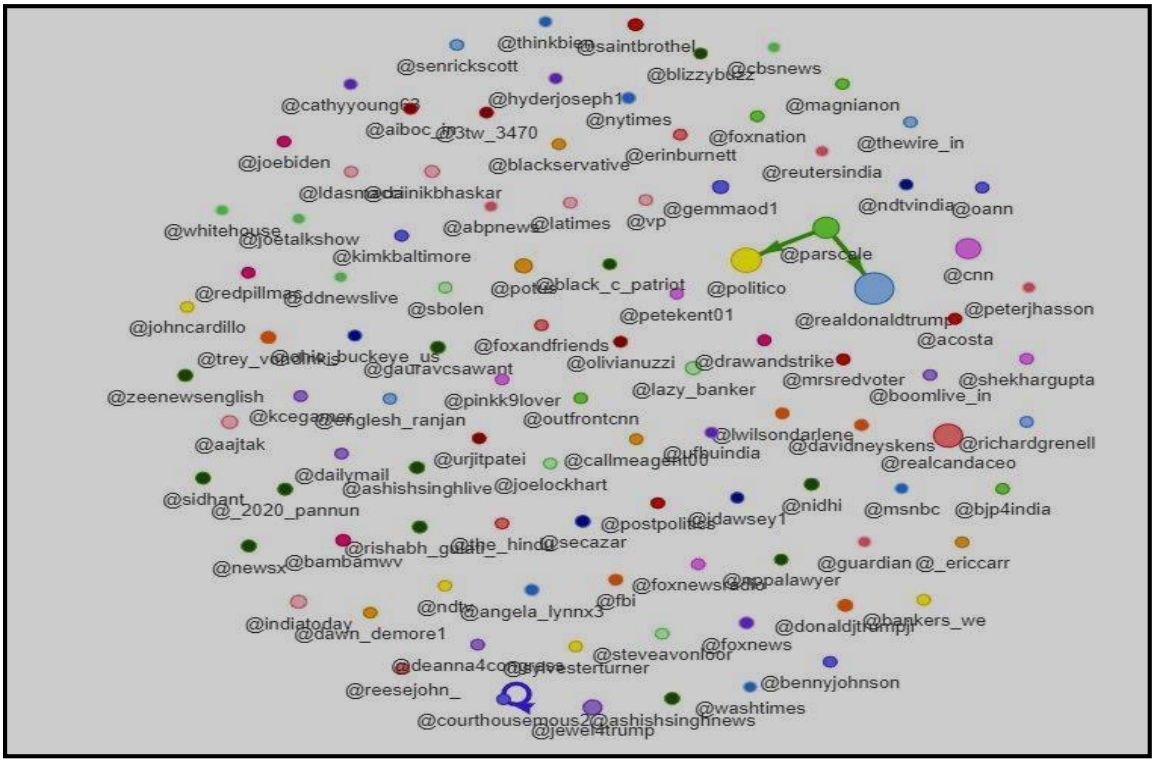


Figure 4.12: Social network graph of the Verified and Non-verified users tweeting on the trending topic

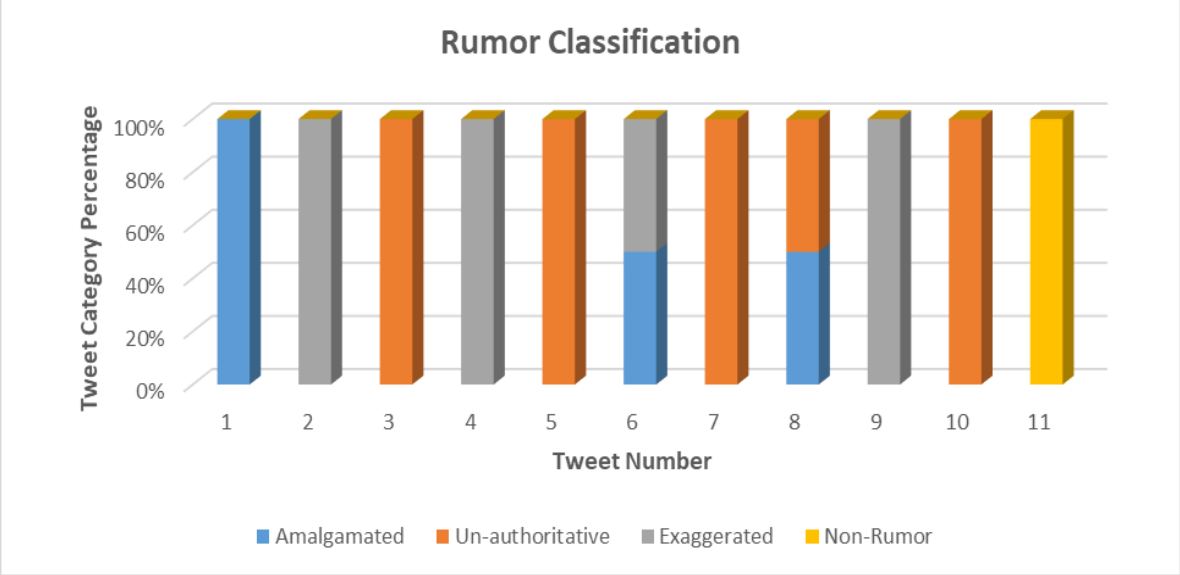


Figure 4.13: Classification of tweets based on VADER classifier compound score

Table 4.2: Tweet Classification based on Modified VADER Compound Sentiment Scores

Sr.	Tweet	Un-authoritative	Amalgamated	Exaggerated	Non-Rumor	Compound
1	1. TAKE good care of yourself. Mask and gloves okay? Don't go outside without them. The #CoronaVirus is airborne. 2. State of Emergency Declared for Black America as Public Health Experts Reveal Coronavirus is Airborne	0	1	0	0	-0.79
2	Trump's 'disinfectantinjection' claim has actually caused people to poison themselves	0	0	1	0	-0.49
3	@narendramodi Honourable Sir,please tell us why are people being deprived of Alcohol especially when it has immunity booster and disinfectant capability. If open fruits and vegetables can be sold why not packed and sealed liquor? Please look into this.	1	0	0	0	0.31
4	if u want help with corona my advice would be to turn of 5G networks the masts and the phones are radioactive causing	0	0	1	0	-0.4215

	illness to spread wuhan turned of 5G corona arrested c my point					
5	Coronacannotbetreated, moron. No cure or vaccine is present. People recover from corona. Get ur facts right n ur IQ tested.	1	0	0	0	0.1098
6	Covid-19 is dry in nature therefore no running nose. Do self-check every morning by holding ur breathfor10secs. If you dont cough then good. Coz sometimes there is no symptoms untill your lung is already affected. Keep ur throat wet every 15 mins w warm water.	0	1	1	0	-0.67
7	Guess what's in vaccines? Thimerosol and formaldehyde! Both are dangerous disinfectants! And now all those so called scientists, doctors, and moronic socialists have admitted vaccinesareunsafe!	1	0	0	0	0.417
8	Shocking #corruption in #Corona times !! must clarify why faulty #rapidtestkits were bought at hugely inflated prices !! Who are the culprits in this huge #scam ?	0	1	0	0	-0.8

	#Nation Wants to know -#Nationalist #TV walas					

4.6 Chapter Summary

An event is a significant occurrence at a specific time or location, and event detection is a statistical technique that allows the tracking and identification of critical occurrences in various domains. The ability to instantly comprehend current events and people's opinions through social media streams is a vital field of study. In this research, we have taken case study of twitter data streams to detect rumor events during COVID-19 pandemic. Any pandemic has an adverse impact on people's conduct. Hence, an efficient technique is designed and implemented to detect and classify them in real-time.

The first phase of the proposed technique involves a stateful stream transformation scheme that resolves the noisy and ambiguous terms in Twitter streams to improve the event detection process. A system is designed to extract slang, acronyms, and abbreviations from the local language. This makes it easier to comprehend the implicit meanings that are immersed in twitter streams and enhances the event detection process. The proposed method is compared and evaluated against other strategies, including locality-sensitive hashing [138], cluster summarization [139], entity-based approach [140], and Repp framework [141].

The transformed data streams are then analyzed using several Twitter based feature characteristics such as profile, spread, microblog and temporal features. If the profile is inactive for an extended period and suddenly the user posts a tweet, it can be said

to be a suspicious post. Contrastingly, if a profile contains the word “news” in its username, it can be said to be from some reliable news agency and may be classified in the non-rumor category. Similarly, the tweets are even analyzed considering the factors such as spread_speed, likability, region of re-tweets, etc. Using these parameters, rumors and non-rumors are classified and finally, the proposed approach (VADER Classifier) is designed to categorize the tweets into four categories that is Un-authoritative, Amalgamated, Exaggerated, Non-Rumor. This is implemented utilizing the compound value obtained from VADER Classifier for each filtered rumor. The proposed approach is enhanced to detect rumors in both supervised and unsupervised datasets. The main advantage of the enhanced approach is that it extracts meaningful information from Twitter data streams. The experimental evaluation validates high event detection rate and low false alarms compared to other techniques. Also, the scalability of the approach has been validated by varying the data size. The data stream transformation and event detection approach proposed in this chapter is specifically designed to handle social media data. In the next chapter, we propose a framework for critical event detection in weather data.

Chapter 5

AN APPROACH FOR ONLINE EVENT DETECTION IN DATA STREAMS

In this chapter, a hybrid approach for event detection is proposed. The existing offline techniques that mostly handles big data algorithms, lack scalability and consistency while handling data streams. The proposed approach integrates the existing offline approach with online streaming framework to handle data streams in real-time. The proposed approach is specifically designed to detect and handle weather data streams but can also be validated on other data sources. In Section 5.1, the background and preliminaries are discussed. Then, the proposed offline-online approach is discussed in Section 5.2. Further, the working of the proposed approach is elaborated in sub-sections of 5.2. The performance evaluation and comparative analysis of an approach is discussed in Section 5.3. Finally, some standard parameters are used to evaluate the performance of the proposed approach.

5.1 Background and Preliminaries

In this section, an intuitive explanation of the state-of-the-art techniques is given. Firstly, the batch models for streaming data are discussed in Section 5.1.1. Then, existing techniques for online event detection are discussed in Section 5.1.2. Finally, a brief explanation of the working of online approaches is given in Section 5.1.3. The various notations used in this chapter are mentioned in Table 5.1 along with their context.

Table 5.1: Notations and symbols used

Notation	Description
C_i	Hidden state or memory unit
X_i	Input to the memory unit
Y_i	Output from memory unit
f_1	Activation function-1
f_2	Activation function-2
W_s	Weight matrix
S_t	Sliding Window at time t

5.1.1. Batch Models for streaming data

Event detection is a significant component of data stream mining and has been implemented using several efficient models. The most popular approaches for event identification are to uncover data points with clearly unique characteristics from their neighbors or to extract anomalies from a dense cluster. In case of an imbalanced dataset and unavailability of target labels, unsupervised techniques are used to detect events in streaming data. Additionally, sometimes the batch models are used which are usually one-class event detection models. The most commonly used one-class model is One-class Support Vector Machine (OCSVM) [142] . The fundamental principle of OCSVM is to utilize a kernel function to map the data from the available class into high dimensional feature map and further to find the most suitable decision boundary for differentiating the observations from the source [143]. This batch model is typically trained using a collection of batch data gathered over a predetermined time period and processed all at once. If the distribution of the data varies with time and differs from the training data, this model usually underperforms.

5.1.2. Online event detection in data streams

The concept of detecting critical events from streaming data in real-time is called online event detection. Time series modelling faces unique difficulties with streaming applications. Such applications involve sensor data and require real-time analysis of a constant stream of data. The whole dataset is unavailable, in contrast to offline event detection also known as batch processing. The main objective of online event detection is that as the data records arrive, the system must observe them in chronological order. Mathematically, it can be supposed that X is the streaming time-series with inputs as $<. . . , x_{t-2}, x_{t-1}, x_t, x_{t+1}, x_{t+2}, . . .>$. It assesses whether the system offers an event at each time t. This decision must be made as soon as the x_{t+1} input is shown. To determine whether the system behavior is abnormal or an event is detected, the algorithm must take into account both the present and prior states. Additionally, the models must be updated with the change in the data patterns and re-trained incrementally.

For instance, in case of data streams from IoT sensors that are constantly evolving and large, it becomes a challenge to analyze it as they require expert and manual intervention. Thus, it is required for an online event detection model to perform in unsupervised and automated fashion.

In existing literature, the event detection methods are classified as model-driven or data-driven. Data-driven approaches evolve on the principles of machine learning, in contrast, the model-driven techniques are more conventional and associated with statistics.

Model-driven approaches

The fundamental model-driven approaches rely on one of the following methods to model temporal data: moving average, volatility, empirical mode decomposition

(EMD), state-space, autoregressive models, or linear regression. Other techniques such as, exponentially weighted moving average (EWMA) [144], generalized ARCH (GARCH) [145], and seminal change point (SCP) [146] all stand out as useful approaches for the detection of trend anomalies, volatility anomalies, and change points, respectively .

Data-driven approaches

The data-driven techniques are based on machine learning concepts and are not application specific and therefore, they are not limited to a particular type of event detection technique. There are several data-driven approaches available that are used to detect events such as K-Nearest Neighbors Conformal Anomaly Detector (KNN-CAD) [147] , Feed-Forward Neural Network (NNET) [148] , Convolutional Neural Networks (CNN) [149] , Support Vector Machine (SVM) [150] , Extreme Learning Machine (ELM) [151] , and K-MEANS [152].

5.1.3. Relevant Theory for the Proposed Approach

The existing studies proposed several methods for the task of event detection in data streams. A tremendous amount of work is done in this domain using Neural Networks (NN), however, these are not compatible with sequential data modelling. Neural Networks are mostly suitable when the data samples exist independently. The type of data instances that depend on individual elements with time including speech, language, time series, video, etc. Concatenating a fixed number of successive data samples and treating them as one data point, comparable to moving a fixed size sliding window across the stream, is one of the methods to account for sequential dependency. This method was implemented in [153] to forecast time series using NNs and in [154] to model audio. However, as indicated in [153], the

success of this strategy depends on determining the ideal window size. A window size that is too narrow would not record the extended dependencies, while a window size that is too wide will introduce extra noise. A window-based approach would not scale if there are long-range relationships in data that span hundreds of time steps.

Sequential data can be modelled using a Hidden Markov model (HMM)[155] without the need for a static window size. By defining probability distributions for transition between hidden states and relationships between observed values and hidden states, HMMs map a recorded sequence to a set of hidden states.

a) Need for RNNs

Recurrent neural networks (RNN) have current outputs that depend on both their current value and their historical inputs, whereas in feed-forward networks current outputs depend on their current input. For instance, we write “Anil is a versatile actor” and “Anil is an Indian Cricketer”. If we ask a human who is Anil, he/she will say as per the statements it seems the name Anil is for two different people, one is an Indian Cricketer and another is an Actor. Now, if the same task is given to a machine to distinguish it may give ambiguous results until it is given some more features or past instances to understand the full context. Such tasks are implemented using Bi-LSTM [156] or LSTM as they are capable of learning the text context and variants of RNN. There are other tasks that LSTM units of RNN can handle. For example, sentimental analysis, machine translation, speech to text conversion, Apple Siri, etc.

RNNs operate by processing each element of the input sequence one at a time while maintaining a hidden state vector that serves as a memory for previous data. They develop the ability to selectively preserve important information, which enables them to detect dependencies over a range of time steps. This enables them to use both data from the present and the past to anticipate the future. Without the model

having much prior understanding of the cycles or time dependencies in the data, all of this is learned automatically. RNNs can accommodate variable length sequences and eliminate the requirement for a fixed size time frame.

Training of RNNs

Training of RNNs is accomplished by unfolding it and making a clone of the model at every input sequence. The unfolded RNN can be trained equivalent to back-propagation by treating it as a multi - layer NN. This method of training RNNs is known as back-propagation over time.

The recurrent node end has the same weight for each time step if the standard RNN is considered (Figure 5.1). Therefore, back-propagating the loss entails repeatedly multiplying the error gradient with the same value again. Resultantly, the gradients can grow hugely big or go away completely. Exploding gradients and vanishing gradients are the terms used to describe these issues. The challenge of exploding gradients can be addressed by scaling it down the pre-defined threshold and this technique is known as gradient clipping [157]. The L1, L2 regularization penalties used to avoid overfitting in NNs are comparable to the penalty term which is introduced in [158] to address the vanishing gradients problem. However, as stated, using a constraint to prevent vanishing gradients increases the likelihood of exploding gradients.

However, the problem of vanishing gradients was finally eliminated by Hochreiter and Schmidhuber [159] by introducing Long-Short Term Memory (LSTM). Comparing LSTM networks to standard RNNs, they have shown to be significantly more effective in learning long-term dependencies, making them the most often used variant of RNN.

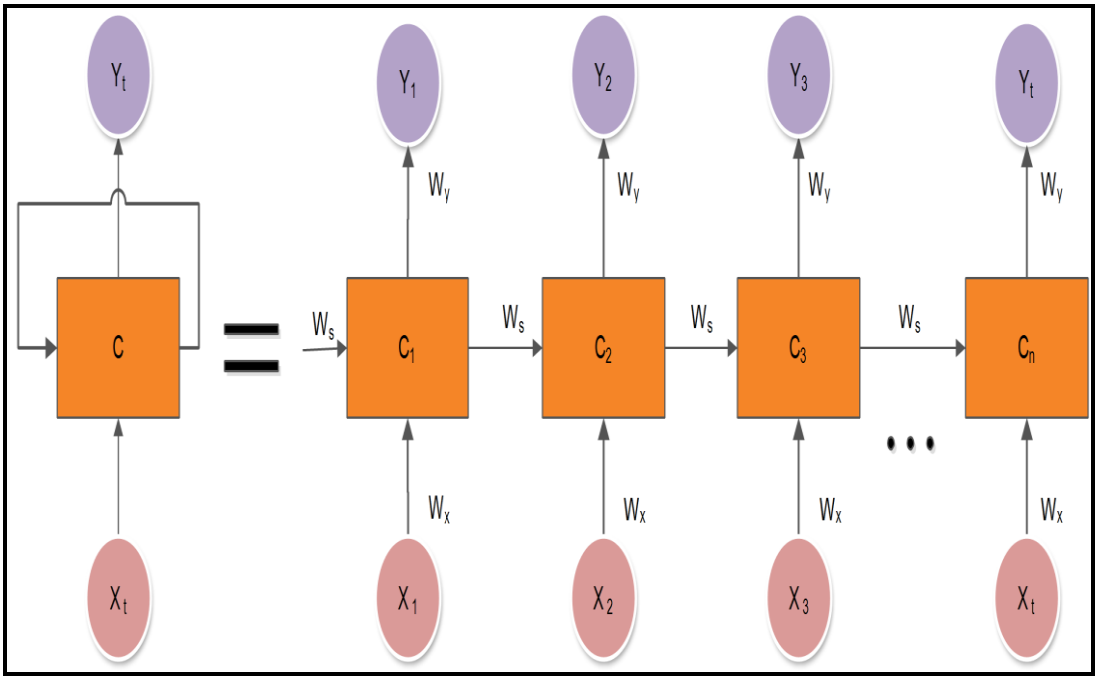


Figure 5.1: Structure of Unfolded RNN

b) Need for LSTM

Dependencies that span arbitrary over long time periods may be learned by the LSTM. The LSTM unit or block solves the problem of vanishing gradients by substituting a basic neuron with complex structure. An LSTM unit is composed of simpler nodes that are connected in a certain manner. The following are the primary elements of the LSTM architecture described as in Table 5.2.

Table 5.2: Description of LSTM architecture

Component	Description
CEC	CEC also known as Constant Error Carousel is a fundamental unit making a recurrent connection with unit weight. It's activation is the internal state that mimics as the memory for the historical information.
Input	It is a multiplicative unit that secures the data stored in CEC from

	noise by biased inputs.
Output	Similar to input it is also a multiplicative unit that secures other components from noise due to information stored in CEC.

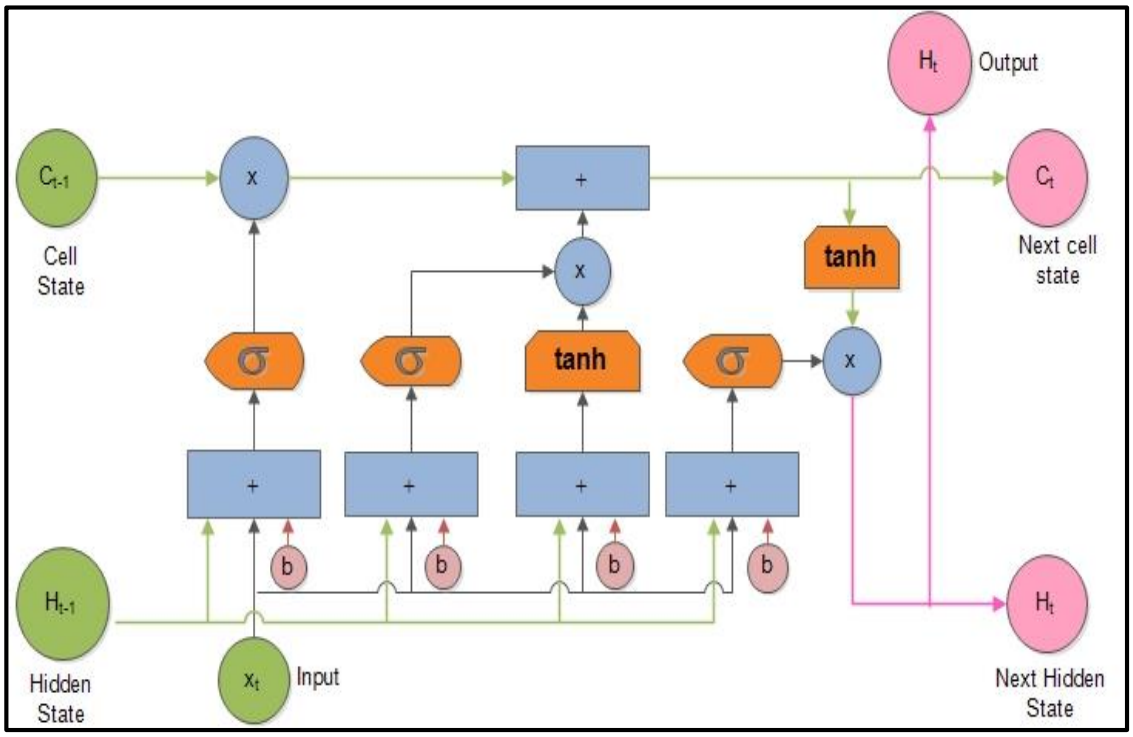


Figure 5.2: Detailed Structure of LSTM Model

Although, there are several characteristics of LSTM that overcome the limitations of standard RNN, still there is a shortcomings of standard LSTM that it becomes unstable on long sequence of input streaming data that is not having any start or end points marked explicitly. LSTMs even cannot learn loops or cycles in the data. Moreover, it is not possible to reset the state in LSTM unless the input streams are segregated into a required sized sequences. It is necessary for LSTMs to work efficiently if they learn to reset the memory component as soon as they complete the processing of one micro-batch stream and before the arrival of a new stream. To mitigate this issue, a novel architecture proposed by Cummins et al. [160] includes

the forget gates. The new concept of forget gates learns to reset the LSTM before the beginning of new sequence. In existing literature [161], a few more modifications have been introduced in the standard LSTM architecture, however, it makes it more complicated with the performance being unaffected. This research work incorporates simple LSTM architecture with forget gates for the task of event detection in weather data streams.

LSTM eliminates the problem of vanishing gradient and therefore, it can store some past inputs in memory. Figure 5.2 illustrates the detailed structure of the LSTM model. It is majorly composed of four components, namely, inputs: X_t, C_{t-1}, H_{t-1}; outputs: C_t, H_t; non-linearities: σ, tanh, b; and vector operations: ×, +. LSTM model transfers two states to the next cell; the cell state and the hidden state. In LSTM, the memory component is used to memorize the factual information and discard the information that is not required. The manipulation functions are implemented using gates: forget gate, input gate, output gate. The forget gate deletes an irrelevant information from the cell state. The main working of the sigmoid function starts here as it filters out the information to be discarded. An input gate is responsible for the addition of information to the cell. An output gate filters the required information that is to be passed as an input to the next cell state. This gate employs tanh function for creating a vector to filter that needs to be stored in memory from that is to be ignored. The functions employed to calculate the weights are defined in the equations below:

$$\text{Input: } \boldsymbol{y_t} = \mathbf{tanh}(\boldsymbol{Q^y x_t} + \boldsymbol{S^y j_{t-1}} + \boldsymbol{c^y}) \quad (5.1)$$

$$\text{Input gate: } \boldsymbol{z_t} = \boldsymbol{\sigma}(\boldsymbol{Q^z x_t} + \boldsymbol{S^Z j_{t-1}} + \boldsymbol{c^z}) \quad (5.2)$$

$$\text{Forget gate: } \boldsymbol{f_t} = \boldsymbol{\sigma}(\boldsymbol{Q^f x_t} + \boldsymbol{S^Z j_{t-1}} + \boldsymbol{c^f}) \quad (5.3)$$

$$\text{Output gate: } \boldsymbol{u_t} = \boldsymbol{\sigma}(\boldsymbol{Q^u x_t} + \boldsymbol{S^u j_{t-1}} + \boldsymbol{c^u}) \quad (5.4)$$

Cell state: $\boldsymbol{CS_t = y_t \odot z_t + CS_{t-1} \odot f_t}$ (5.5)

Output: $\boldsymbol{o_t = \tanh(CS_t) \odot u_t}$ (5.6)

The LSTM unit takes the current input vector denoted by x_t and the output from the previous time step (through the recurrent edges) denoted by j_{t-1}. The weighted inputs are summed and passed through tanh activation, resulting in y_t. The input gate reads xt and j_{t-1}, computes the weighted sum, and applies sigmoid activation. The result is multiplied by y_t , to provide the input flowing into the memory cell. The forget gate is the mechanism through which an LSTM learns to reset the memory contents when they become old and are no longer relevant. This may happen for example when the network starts processing a new sequence. The forget gate reads x_t and j_{t-1} and applies a sigmoid activation to weighted inputs. The result, f_t is multiplied by the cell state at previous time step (CS_{t-1}) which allows forgetting the memory contents that are no longer needed. This comprises of the CEC, having a recurrent edge with unit weight. The current cell state CS_t is computed by forgetting irrelevant information (if any) from the previous time step and accepting relevant information (if any) from the current input. Output gate takes the weighted sum of x_t and j_{t-1} and applies sigmoid activation to control the information that would flow out of the LSTM unit. The output of the LSTM unit, o_t , is computed by passing the cell state CS_t through tanh and multiplying it with the output gate, u_t.

The sigmoid function in the gates of an LSTM enables the network to learn to selectively store, neglect, and output information based on the sequence of inputs. The sigmoid function lets the gates generate information between 0 and 1 that show strength of the activation or the probable outcome of information flow.

The key offerings of LSTMs make it useful for the application of critical event detection in data streams:

a) LSTM networks implement a memory cell structure to accumulate and remember long-term dependencies. As a result of this, they can be used to model sequential data like time series or sensor information.
b) The memory cell can store important historical information about the data stream. This facilitates the model to learn and adapt to the changing patterns.
c) After the model learns the regular patterns and relationships in the data, it can find cases that are very different from the patterns it has learned, which indicates the existence of notable events. Due to this, LSTMs are particularly suitable for identifying events like system failures, network intrusions, or unusual behavior across a variety of domains.

c) Need for Apache Spark

Apache Spark was chosen as the data processing framework in the online pipeline component of the proposed approach for a variety of reasons. The first and foremost is its stream processing capabilities. Spark processes the data in batches as well as micro-batches serving as one of the most competing frameworks amongst others. Secondly, it offers wide range of components such as SQL, graph, integrated data ingestion pipeline, and support for machine learning algorithms. To explicate, Spark MLib component offers support for all machine learning tasks such as clustering, classification, regression and predictions. Finally, the most significant feature of Apache Spark is its in-memory processing capabilities. This characteristic is one of the most vital reasons while choosing this framework as it helps in low latency processing and is suitable for voluminous data. Last but not the least, Spark streaming consists of windowing operations, enabling to specify time or space-based

processing and data-aggregation frames. This makes it possible to perform different time-based calculations, like sliding or tumbling windows. The reference architecture of Apache Spark can be visualized as given in Figure 5.3.

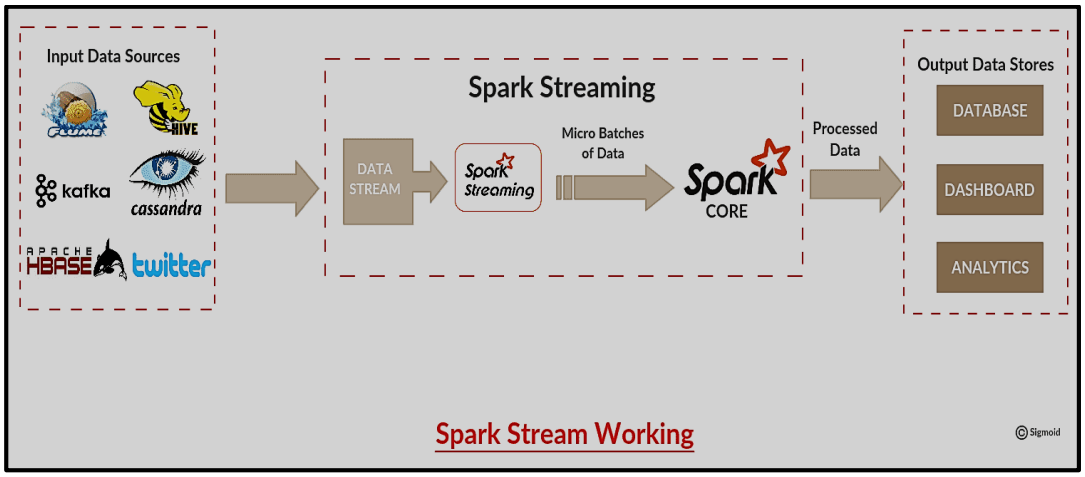


Figure 5.3: Reference Architecture of Apache Spark [44]

5.2 Proposed Approach

The proposed approach for critical event detection is based on the dual mode of data processing that is online and offline. It can be analyzed from the literature that online event approaches tend to outperform offline approaches in terms of several performance metrics. An online event detection approach can prove to be efficient if it processes the data in real time or near real-time as well as has strict temporal constraints. Moreover, it is also essential for an event detection algorithm to update the learning patterns incrementally as it ingests only limited labelled data. As soon as new data streams are ingested, online event detection algorithms maintain and update their models accordingly. This includes mechanisms for drift detection, model retraining, and parameter tuning. To accomplish this in online event detection settings, techniques like online learning, ensemble methods, anomaly detection, and model adaptation are frequently used.

The proposed approach consists of offline-online pipelines to firstly train the neural networks with data streams and further, use the trained model in online pipeline that is Spark Streaming to precisely detect anomalous events in streaming data. The data utilized to train the neural networks is a real-time weather data stream extracted using PubNub API. The data is refreshed in micro-seconds and the model learns the patterns in the data incrementally. Pandas Library of Python is used to convert the collected data into data frames, with the specified characteristics that is timestamp (s), ambient_temperature (C), humidity (g/m3), photosensor (k), sensor__uuid (uniqueidentifier), and radiation_level (Gy). The snapshot of the ingested data is shown in Figure 5.4. Figure 5.6 represents the architecture of the proposed approach.

5.2.1. Data Pre-processing

Data preparation is an essential stage in deep learning. It is difficult to simply input raw data into a model and then expect it to perform efficiently. When the input raw data is used to train the model, its performance improves gradually. Processing raw data is crucial to improve the performance of the model. The model might regard the features with larger values as more relevant and ignore some important characteristics having Boolean data if the dataset is not scaled. If the features have same scale, the model operates effectively and may show rapid convergence. Direct standardization is the data pre-processing technique that is typically thought to be the answer to this issue. Assuming the variables are columns, further the column is standardized, and each value has the column mean value deducted from it.

	timestamp	ambient_temperature	humidity	photosensor	sensor_uuid	radiation_level
0	1609914313	22.75	76.9555	814.18	probe-1db19108	200
1	1609914313	24.62	75.6012	772.75	probe-64d619e5	195
2	1609914313	19.99	87.4187	787.29	probe-b52b4a12	205
3	1609914313	32.35	82.9612	818.89	probe-8ad26973	198
4	1609914313	26.11	70.0489	699.46	probe-59c77f12	198

Figure 5.4: Snapshot of the data frame of a streaming dataset

5.2.2. Data Stream Transformation

As the primary objective of our research is transformation of streaming data, various novel combinations of stateful transformation operations are performed on this moving data such as Sliding window transformation, map, filter, flatMap, reduceByKey, reduceByKeyAndWindow, countByValueAndWindow, updateStateByKey, etc. A set of complex stream pipelines are built to gain insights from real-time data. Using the window transformation, streaming data can be processed in windows of different lengths and sliding intervals. The elements within each window are subject to modifications or aggregations by the windowed rules. The window transformation can be represented as given below in a code snippet:

```
windowedDStream = inputStream.window(windowLength, slideInterval)
```

Here, inputStream: Original DStream

windowLength: Size of the window (time or number of batches)

slideInterval: time period between start of consecutive windows

```
import org.apache.spark.streaming._
val ssc = new StreamingContext(sparkConf, Seconds(1))
val inputStream: DStream[Int] = ... // Input stream of
integers
val windowedDStream = inputStream.window(Seconds(10),
Seconds(5))
```

For a given weather data stream, the above code snippet shows window size of 10 seconds and a sliding interval of 5 seconds. Further, the transformation operations such as ‘reduceByKey’ and ‘countByValue’ are implemented to process the data within each time window. Finally, the Bloom filters [162][163] are applied on streaming data to quickly query whether or not a given element is present in a sizable dataset. A code snippet for Bloom Filter is given in Figure 5.5. However, Apache Spark provides transformation integration and suitable transformations can be applied online on Resilient Distributed Datasets (RDDs).

5.2.3. Model Training and Optimization: Offline Phase

Data Splits

The data stream taken from PubNub ambient sensor is firstly captured and fragmented into data frames for offline data processing pipeline. The captured dataset is divided into splits of 80:20 that is 80 % for training the model and 20% for testing it. RNN and LSTM are the two candidate forecasting models chosen for training using weather data streams.

```
class BloomFilter:
    def __init__(self, size, hash_functions):
        self.size = size
        self.bit_array = [False] * size
        self.hash_functions = hash_functions
    def add(self, element):
        for fn in self.hash_functions:
            index = fn(element) % self.size
            self.bit_array[index] = True
    def contains(self, element):
        for fn in self.hash_functions:
            index = fn(element) % self.size
            if not self.bit_array[index]:
                return False
        return True
def hash_function_1(element):
    return int(hashlib.sha256(element.encode()).hexdigest(), 16)
def hash_function_2(element):
    return int(hashlib.md5(element.encode()).hexdigest(), 16)
size = 100
hash_functions = [hash_function_1, hash_function_2]
bloom_filter = BloomFilter(size, hash_functions)
```

Figure 5.5: Code Snippet of Bloom Filter

Proposed RNN architecture:

There are 3 layers in RNN structure: Input, Recurrent and Output layer.

Input Layer: At each time step, the input layer receives the streaming data as input.

Recurrent Layer: The number of recurrent layers depend on the complexity of the data. However, in this experiment up to 3 layers are selected as sometimes the data is not constant for a period of time.

Output Layer: The output layer uses the recurrent layer's hidden state to generate the desired prediction or classification.

The activation function used in the recurrent layer is Rectified Linear Unit (ReLU) as it keeps the positive values unaltered while negative values are reset to zero. ReLU is frequently employed in deep neural networks, particularly RNN based architectures, and can offer faster training convergence.

Weather data frequently displays complex and nonlinear correlations. The neural network may successfully capture and model these nonlinearities by employing

ReLU as an activation function. The addition of ReLU to the network allows it to learn and express more complicated patterns and relationships in the data.

Moreover, ReLU has a sparse activation characteristic which means it can activate only required amount of neurons and deactivate those which are not in further use. Hence, reducing the computational overhead and improving efficiency.

Further, the number of hidden layers to be added in the LSTM model is decided with the help of combination of keras library *KerasClassifier* and sklearn library *GridSearchCV* or *RandomizedSearchCV*.

The number of learnable parameters in LSTM are calculated using the formula given in Eq. 5.7.

$$4*[(n+u+1)*u] \quad (5.7)$$

Assuming LSTM has *u* hidden units and *n* is the input dimension vector. There are 4 functional units in LSTM, 3 *sigmoid* and 1 *tanh*, therefore, it is multiplied by 4 for each functional unit to compute the number of learnable parameters. The selected optimization function is Adaptive Moment Estimation (ADAM) optimizer as the main focus of weather datasets and the objective of this research revolves around faster computation with best accuracy. Stochastic Gradient Descent (SGD) could have also been used, however, it would require more iterations and computation time to compete the accuracy of ADAM optimizer. Further, ADAM also requires less parameters for fine-tuning the model [164][165].

Model Evaluation:

The performance of RNN and RNN-LSTM models is evaluated by choosing Mean Absolute Error (MAE) as an evaluation metric because the task of event detection in weather data streams is a regression problem and it proves to be a measure of

continuous variations in data streams. MAE is computed by using Eq. 5.8, where y_i: Prediction Value, x_i: True Value and n: Total number of data points.

$$\text{MAE} = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n} \quad (5.8)$$

5.2.4. Model Training and Optimisation: Online Phase

The online event detection pipeline component is divided into two steps: ingestion and event detection.

Each component is discussed below:

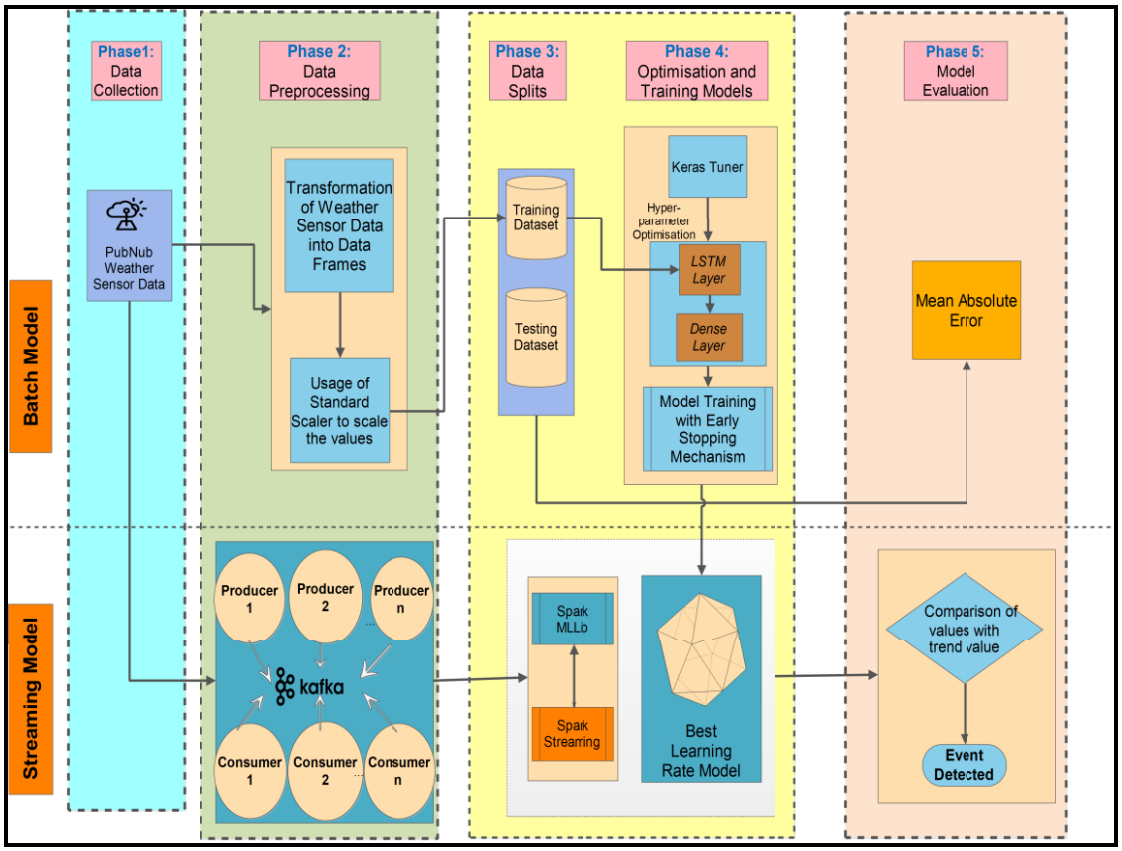


Figure 5.6: Architecture of the proposed framework for Real-time event detection in Weather Data

Data Pipelines

Apache Kafka 3.3.0 is installed on the system with appropriate configuration and further, a connection is established with PubNub weather sensor data using HTTP

protocol. According to the publish-subscribe approach used by Kafka, data is produced and consumers subscribe to certain topics in order to access it. In Kafka, the topics serve as a primary organizational units of data. They represent the published and consumed records streams. The partitioning of each topic enables parallel processing and data distribution over numerous servers and a seamless integration between Kafka and Spark is done using Kafka Connect framework. The connectors in this framework allow data to be streamed in and out of Kafka and simplifies the integration process with variety of data sources.

Online Data Stream Processing

The further integration of Kafka is done with Spark Streaming by following the mentioned steps:

1. The Kafka dependency is included in the Spark Streaming project where MAVEN or SBT acts as the dependency manager.
2. The Kafka consumer is configured in Spark project to accept the data frames from Kafka topics as seen in Figure 5.7.
3. Using the Spark Streaming transformations, the ingested data streams are processed for further task. The stream.foreachRDD command allows the processing of each RDD of Kafka data streams as seen in Figure 5.8.
4. The Spark Streaming context starts processing after configuring the Kafka consumer and embedding the processing logic and synchronizes the data frames as seen in Figure 5.9.
5. Next, the data is converted into Data Frames which is a format that SparkML Library follows for any model to accept the data.

6. The trained LSTM model in Offline phase is now trained again on the transformed DataFrames using fit() method to detect events. The projected values based on the trained LSTM model are included in the Data Frame predictions that follow.
7. Finally, MAE is again computed to evaluate the performance of the model.

```
import org.apache.spark.ml.evaluation.RegressionEvaluator

val evaluator = new RegressionEvaluator()
  .setLabelCol("label")
  .setPredictionCol("prediction")

val rmse = evaluator.evaluate(predictions)
println(s"Root Mean Squared Error (RMSE) on test data: $rmse")
```

The above steps repeat for each window of processed streaming data. Further, the integration of online learning component improves the precision of event detection.

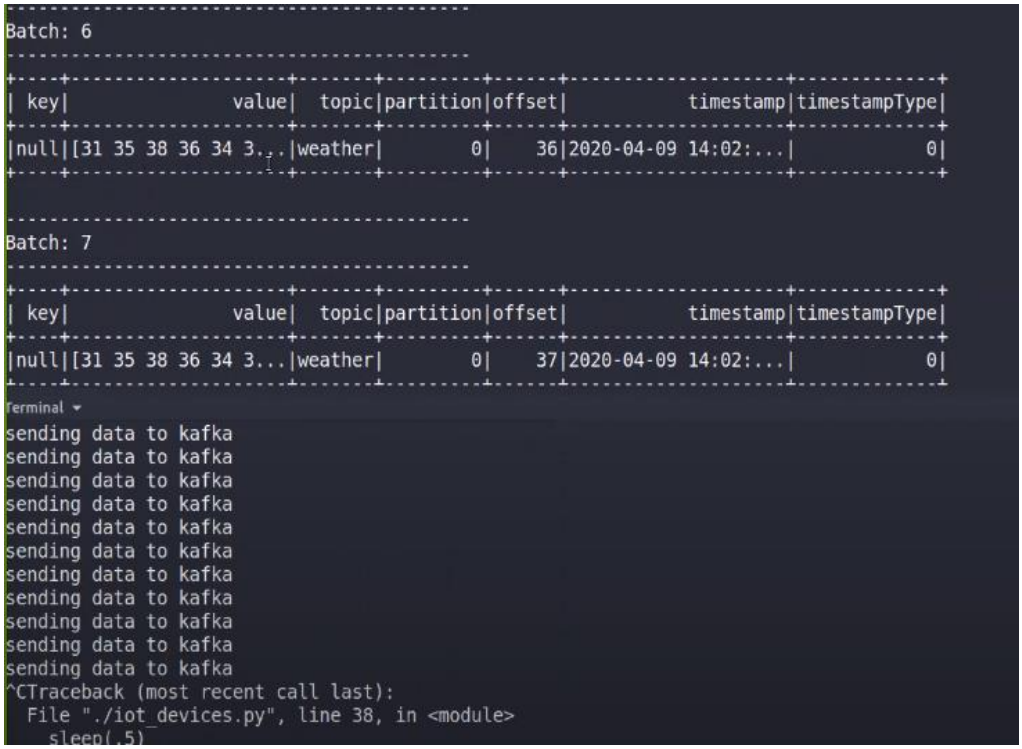


Figure 5.7: Snapshot of Kafka Integration with Spark

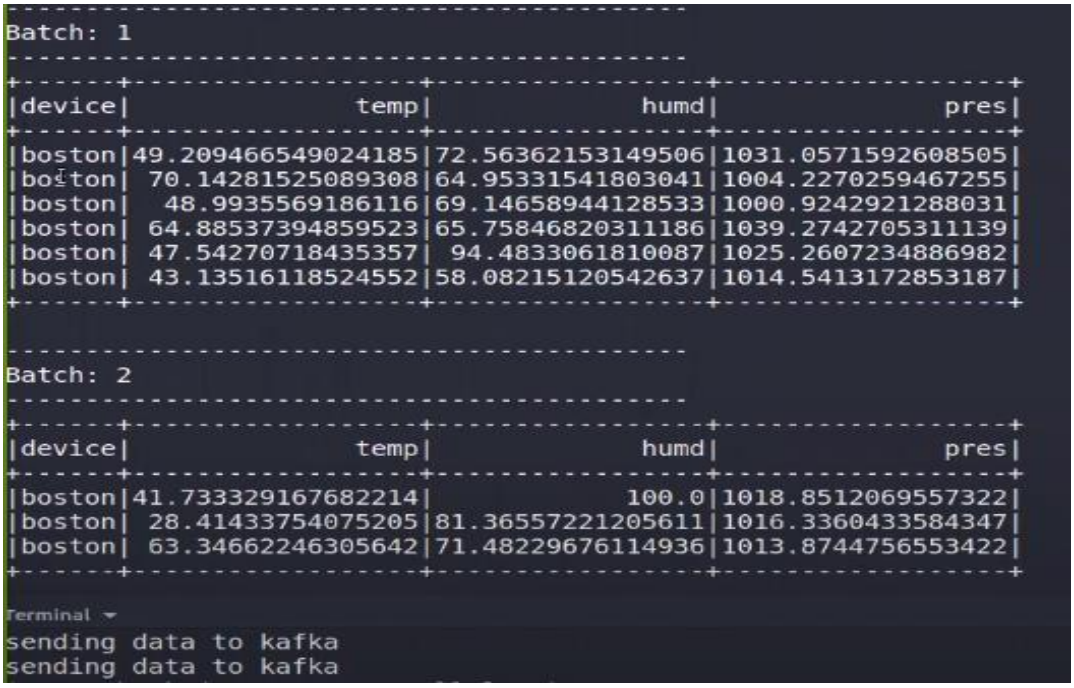


Figure 5.8: Snapshot of Data Streams ingested by Kafka

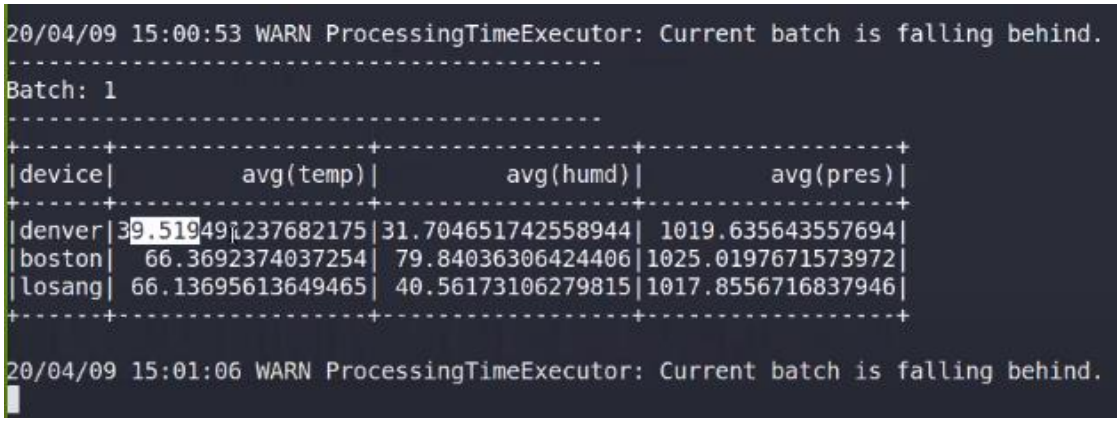


Figure 5.9: Maintaining synchronization between micro-batches of data received by Kafka by raising warnings

Updating Dynamic Thresholds in LSTM model for Event Detection

An agile universal, and unsupervised approach is required for the automated monitoring of thousands of telemetry channels where the expected values change in response to fluctuating environmental conditions and input sequences. There are several approaches discussed in the existing literature to address the detection of critical thresholds in dynamic environment. One such approach is making Gaussian

assumptions [166][167][168] about the variances of previous normalized errors since it enables quick inferences between concise representations of old errors and new errors. However, this method encounters some constraints when the parametric assumptions are not addressed. A novel technique to select the thresholds dynamically is proposed in this research so that the critical events detected are precise and have minimum false alarms. In this study, a novel unsupervised approach that demonstrates exceptional performance while maintaining minimal computational cost is proposed. Notably, this dynamic threshold method does not rely on annotated data or make any statistical assumptions regarding error patterns. Given the current factors, it is evident that the threshold ϵ is chosen from a given array where $\in$ is the threshold from a given Eq 5.9:

$$\in = \boldsymbol{\sigma}(\boldsymbol{t_x}) + m\,\boldsymbol{\mu}(\boldsymbol{t_x}) \tag{5.9}$$

where ϵ is computed using:

$$\epsilon = argmax(\epsilon) = \frac{\frac{\Delta\boldsymbol{\sigma}(\boldsymbol{t_x})}{\boldsymbol{\sigma}(\boldsymbol{t_x})} + \frac{\Delta\boldsymbol{\mu}(\boldsymbol{t_x})}{\boldsymbol{\mu}(\boldsymbol{t_x})}}{|\boldsymbol{t_a}| + |\boldsymbol{S_{seq}}|^2}$$

$$\Delta\boldsymbol{\mu}(\boldsymbol{t_x}) = \boldsymbol{\mu}(\boldsymbol{t_x}) - \boldsymbol{\mu}(\{\boldsymbol{t_x} \in \boldsymbol{t_x} | \boldsymbol{t_x} < \in\})$$

$$\Delta\boldsymbol{\sigma}(\boldsymbol{t_x}) = \boldsymbol{\sigma}(\boldsymbol{t_x}) - \boldsymbol{\sigma}(\{\boldsymbol{t_x} \in \boldsymbol{t_x} | \boldsymbol{t_x} < \in\})$$

$$\boldsymbol{t_a} = \{\boldsymbol{t_x} \in \boldsymbol{t_x} | \boldsymbol{t_x} > \in\}$$

S_{seq} = continuous sequences of $t_a \in t_a$

The values for ϵ can be obtained by evaluating $m \in m$, where m represents a sequence of integers with positive values that correspond to the number of standard deviations exceeding $\sigma(t_x)$. The values of m are variable depending upon the specific setting; our experimental findings indicate that a range of precipitation between two to ten yields favorable outcomes. The values less than the lower limit value of m may result in excessive false positives. In a simpler sense, the objective is to choose

a threshold value that, when applied to the data, would result in the maximum percentage decline in both the mean and standard deviation of the smoothed errors (t_x).

In a simpler sense, the objective is to choose a threshold value that, when applied to the data, would result in the maximum percentage decline in both the mean and standard deviation of the smoothed errors (t_x). The function additionally includes costs for the presence of a higher number of anomalous events ($|\boldsymbol{t_a}|$) and sequences ($|S_{seq}|$) in order to mitigate excessively peak value behavior. The normalized value for the largest smoothed error in each sequence of event value is determined based on its deviation from the selected threshold.

5.3 Experimental Analysis

5.3.1 Experimental Setup

Deep learning models and data stream frameworks both are used in the implementation of the proposed event detection system. To design an offline model component with the best and least MAE value, several experimental studies are carried out on Tools and the technologies outlined in Table 5.4. The performance of the proposed framework is evaluated in terms of Mean Absolute Error (MAE). The proposed method configures both RNN and LSTM with different constant hyper parameters as described in Table 5.3.

Table 5.3: Hyper-parameter tuning of LSTM

Hyper parameters	Value
Hidden Units	200
Dropout layer	0.2
Gradient Threshold	1
Initial Learn Rate	0.000001
Learn Rate Drop Factor	0.2
Learn Rate Drop Period	125
Mini Batch Size	64
Epochs	100

Table 5.4: Details of the components and tools of the proposed approach

Component(s)	Tools/Technologies used
Data Gathering	PubNub API
Data Ingestion	Apache Kafka (v:3.5.0)
Online Model Training	Python, Keras, SparkML
Online Data Pre-processing	Apache Spark, Scala
Machine for Spark and Kafka setup	Ubuntu (VM)

5.3.2 Results and Discussion

Observations from RNN Model

RNN model is experimented upon with up to 3 hidden layers. By using 1, 2, and 3 as the number of hidden layers, MAE values are computed and compared. It can be vividly seen from Table 5.5 having MAE values as 0.9315, 0.9899, and 1.2313 that the RNN model with a single hidden layer yields the lowest MAE in comparison to the model with three hidden layers, which yields the highest value and it can be said

that the model with just 1 hidden layer proves to be much better than the model with more hidden layers. Additionally, it proves to be more space efficient.

Observations from LSTM Model

With the proposed structure of LSTM in offline phase of the approach, it achieved the least value of MAE with 3 hidden layers. This work examines the training loss and validation loss in addition to the MAE values after each epoch. This is typically done to evaluate how well the model performs following each iteration. The learning curve displays a good fit. Both the training loss and validation loss curves drop to a stable value, with only a slight difference between them. As seen in Figure 5.11, the model is demonstrated to be optimal and also prevents overfitting and underfitting. Table 5.6 shows performance of LSTM model w.r.t count of hidden neurons and hidden layers. Further, the bar chart in Figure 5.10 depicts the comparison of MAE using both the models RNN and LSTM. Figure 5.11 shows best fit deep learning model w.r.t Training and Validation Loss. Figure 5.12 visualizes the real-time anomalous events identified based on threshold values at a particular time instance.

Table 5.5: Performance of RNN Model

Hidden layers	Count of Hidden Neurons	MAE value
1	[256]	0.9315
2	[70,160]	0.9899
3	[80,150,220]	1.2313

Table 5.6: Performance of LSTM Model

Hidden layers	Count of Hidden Neurons	MAE value
1	[64]	0.9106
2	[64, 128]	0.8711
3	[192,150, 64]	0.7806

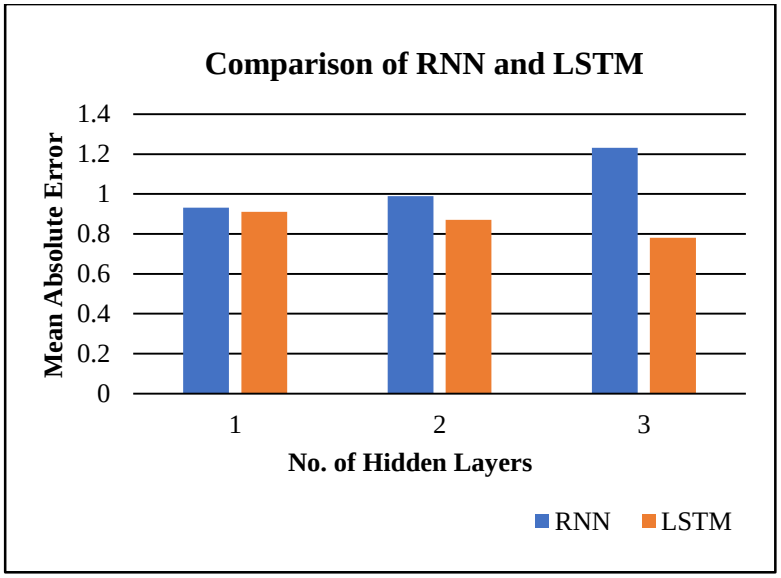


Figure 5.10: Comparison of RNN and LSTM based on MAE value

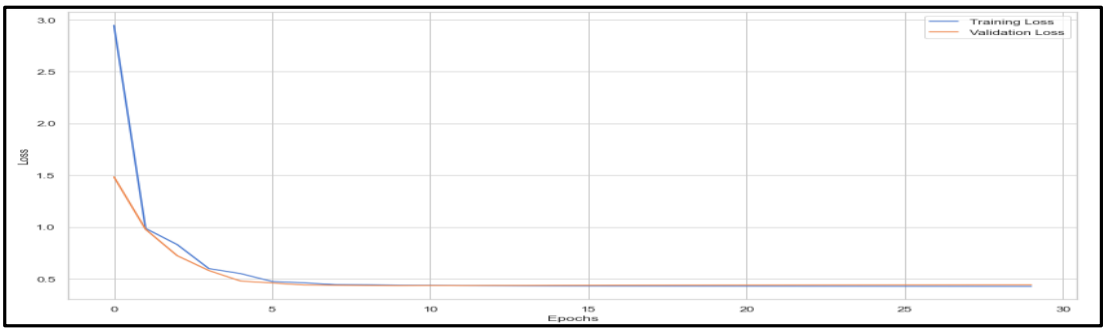


Figure 5.11: Evaluation of best fit deep learning model w.r.t Training and Validation Loss

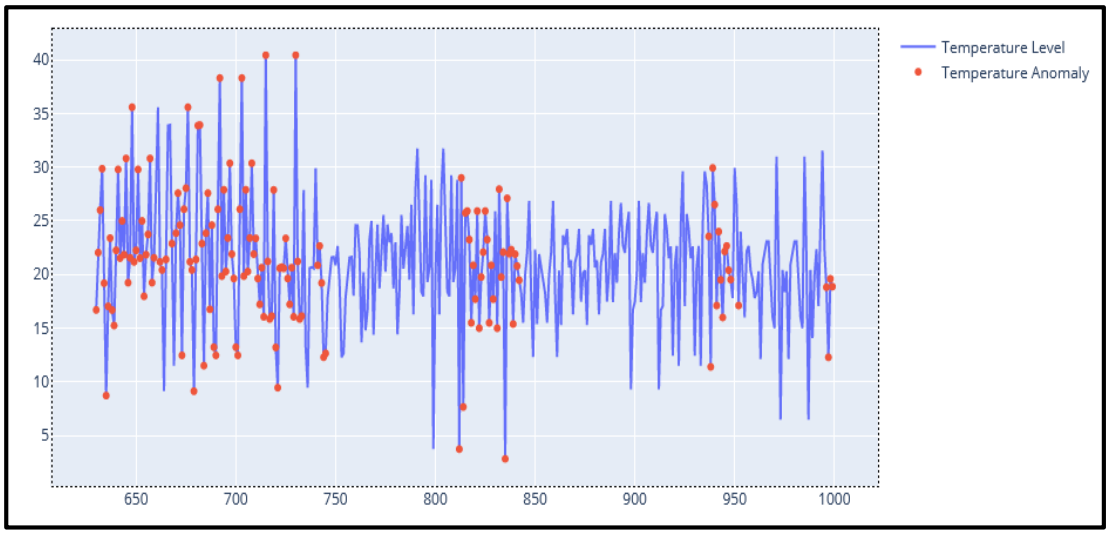


Figure 5.12: Real-Time Events detected based on threshold value at a particular instance of time

5.3.3 Comparison of proposed approach with state-of-the-art approaches

The offline-online approach proposed in this thesis is validated for scalability on continuous data streams from PubNub Weather streaming data having size greater than the default block size of Spark Sliding Window.

The default block size in Spark is generally 64 MB. However, the data block size used in this research is 128 MB. It is perceived that the proposed approach is scalable to this huge fire-hose of data. However, as the data size increases, by using multiple worker nodes the proposed approach takes significantly less amount of time and is also proven space efficient by introducing in-memory processing Spark framework.

Comparison with Baseline approaches

A similar approach proposed by Liu *et al.* [169] for smart energy management system utilizes lambda system and prediction based method for detecting anomalous consumption patterns in real-time. However, they used PARX model which is a periodic auto-regression with exogenous variables and is used for short-term residential electricity consumption prediction. In contrast, the prediction model used in our research is RNN and LSTM unit of RNN that offers better time-efficiency and minimizes latency in streaming data. Moreover, it also works well with SparkML library of Apache Spark.

Another similar approach was proposed by Khan et al. [170] for intrusion detection system utilizing hybrid approach, SparkML and Conv-LSTM. The anomaly

detection module, which is built on Spark ML, is used in the first stage. The Conv-LSTM network-based module for detection of level of compromise due to intrusion is implemented in the second stage which addresses both global and local implicit concern signals. This approach is treated as the baseline approach because it also solves the similar issue in real-time and uses 2-component approach. However, this approach may be inefficient to deal with streaming scenarios in comparison to our proposed approach as it lacks incremental re-training of model along with online processing of evolving streaming data. Table 5.7 entails the feature-based comparison of the proposed technique with some state-of-the-art techniques. However, this is the first approach that uses PubNub sensor data for real-time detection of critical events from weather data streams.

Table 5.7: Feature-based comparison of proposed technique with state-of-the-art techniques

Author	Streaming Data	Early Stopping Mechanism	Use of Signal Patterns within a stream	Contextual Layer	Considering both Offline and Online Event Detection
Xiaohan L. et al.[171]	✗	✗	✓	✓	✗
Uma M. et al.[172]	✓	✗	✗	✗	✗
Veeravalli et al.[173]	✓	✗	✓	✗	✓
Hager et al.[156]	✓	✗	✗	✗	✓
Su Peng et al.[174]	✗	✗	✓	✓	✗
Arora et al. (Proposed technique)	✓	✓	✓	✓	✓

5.4 Chapter Summary

In this chapter, a novel two-component architecture comprising of an offline and an online component to identify events in real-time is discussed. When implemented individually, batch and data stream processing techniques each have their own advantages and disadvantages. However, a significant merit of the proposed event detection approach is to adequately adapt to real-time streaming data and detect an unusual value or critical event in near real-time. Apache Spark, Apache Kafka, and deep neural networks are leveraged in combination for the implementation of the proposed system. The offline component is designed to determine the best deep learning model that produces the least MAE value. Further, RNN and LSTM, the two deep learning models are trained on the PubNub weather streaming dataset, are analyzed in this component.

The continuous data flow is ingested and then transmitted to the Kafka queue in the later component, or online model, where the stream processing engine (Apache Spark) transforms and processes it before passing it to LSTM to quickly detect events. The effectiveness and real-time event detection in the proposed approach is determined by a comprehensive comparison with state-of-the-art techniques. It makes a significant contribution to the development of the critical weather data warning system. LSTM requires four linear layers (MLP layer) per cell to operate, and these layers require a significant amount of memory bandwidth. As a result, our technique is not attractive to hardware designers, and it becomes difficult to expand the number of computational units because each unit needs memory to function.

However, LSTM in combination with in-memory streaming framework. Hence, Spark proves to be time as well as memory efficient.

In the next chapter, the problem of concept drift is addressed that occurs during the training of models on streaming data.

Chapter 6

DETECTING CONCEPT DRIFTING EVENTS IN DATA STREAMS

This chapter discusses dynamic ensemble based techniques leveraging selective transfer models for the detection of concept drifts in data streams. The proposed technique introduces a selective parameter for transferring the effective models in the ensemble to minimize the training time and avoid negative transfer challenges. The proposed approach obtains high prediction performance when tested using varied data streams. In Section 6.1, the background and preliminaries are discussed. Then, the proposed selective transfer for dynamic ensembles approach is discussed in Section 6.2. Further, the working of the proposed approach is elaborated in sub sections of 6.2. The performance evaluation and comparative analysis of an approach is given in Section 6.3. Standard parameters are used to evaluate the performance of the proposed approach.

6.1 Background and Preliminaries

There has been a significant inclination towards online learning in the field of predictive analytics due to recent developments in the area of data stream mining. The domains in which machine learning models are implemented are constantly changing, which presents a significant problem for stream analytics in real-world applications. As a consequence, the tools developed to analyse such data will

eventually become irrelevant or will not be able to adapt to changing patterns of data. In these cases, there is an utter requirement to design techniques for continuous learning that can adjust to new information streams and evolving concepts. The change in the underlying distribution of the changing patterns of data streams is known as concept drift. This change may occur suddenly, gradually or may be recurrent. The challenge of implementing a model for data streams arises when there are huge shifts in patterns of data. The target variable's features shift over time, making it more difficult for a machine learning model to make accurate predictions. In simple words, there are differences in features of the data that a model has been trained on from the data that is targeted. Hence, detecting concept drifts and adapting to those to develop online learning models effectively is need of the hour.

Mathematically, a concept drift may be defined according to Bayesian decision theory [175] that states that the prior probabilities of the classes p(y) and class conditional probability density functions p(X|y) for classes y ∈ 1,2,3,...m where m is the number of classes set for classification job characterize a classification model. Classes are assigned using the posterior probabilities of the classes, with y represented as:

$$\boldsymbol{p(y|X)} = \frac{\boldsymbol{p(y)p(X|y)}}{\boldsymbol{p(X)}} \tag{6.1}$$

At time intervals t_o and t_1, concept drift is detected if there is a change in the joint distribution of the input variables X and the target class variable y.

$$\exists \boldsymbol{X}: \boldsymbol{P_{t0}(X, y)} \neq \boldsymbol{P_{t1}(X, y)} \tag{6.2}$$

P_{to} and P_{t1} above represent the joint distribution at times t_o and t_1, respectively. If we use the aforementioned definition of concept drift, then variations in the data may result from shifts.

6.1.1 Key Issues due to Concept Drift in Streaming Data

- The classifier performance degrades with time due to concept drifts and gradually, it becomes outdated.
- It is challenging to monitor the evolving concept.
- It is difficult for the classifier to adjust to recent instances under non-stationary data distribution.

6.1.2 Types of Concept Drift

The distribution of data streams is dynamic as they are continuous in nature. The data trends may change with time, and their distribution may differ. Such changes in the data distribution may be broadly categorized into two types on the basis of change in decision boundary a) Real Concept Drift and b) Virtual Concept Drift and can be illustrated in Figure 6.2. The categorization of drifts can also be done on the basis of the speed of concept change that is by dividing them into five major types, namely i) Abrupt (Sudden) ii) Gradual iii) Incremental iv) Recurring and v) Blip. A blip can also be termed as an outlier since it denotes a fast or sudden change (or rare event) in a concept. However, it can be ignored from the categorization of drifts as it falls under abrupt category. Further, the drifts can also be categorized into two types on the basis of magnitude that is Local Concept Drift and Global Concept Drift. The overview of categorization of concept drift is depicted in Figure 6.1.

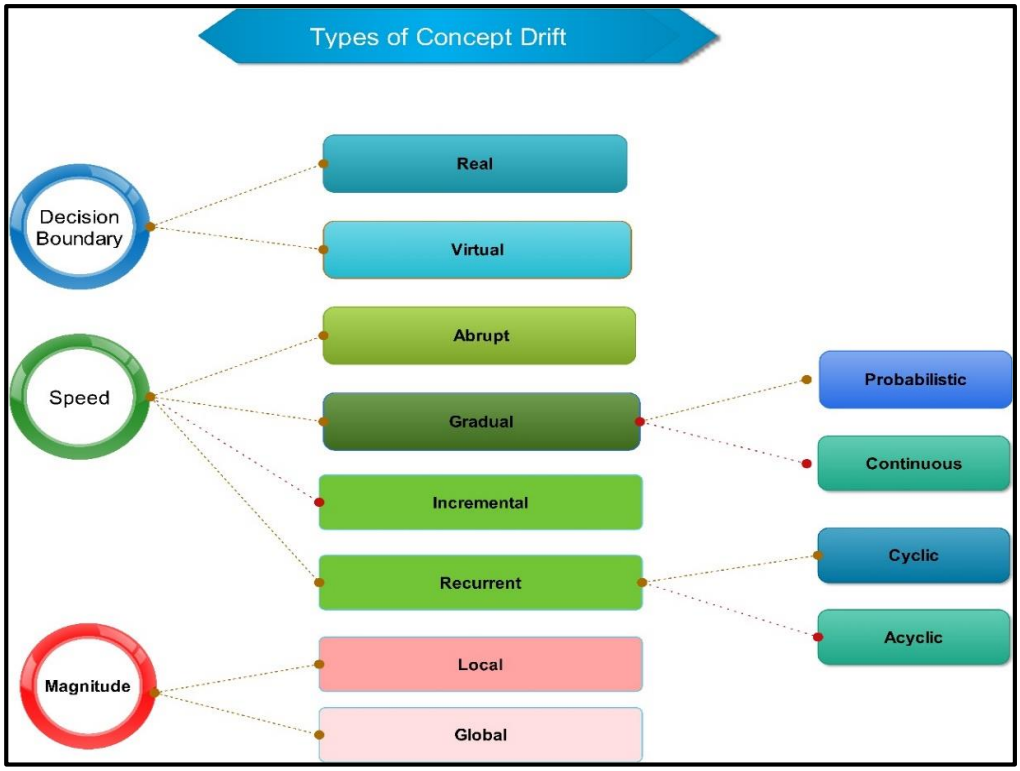
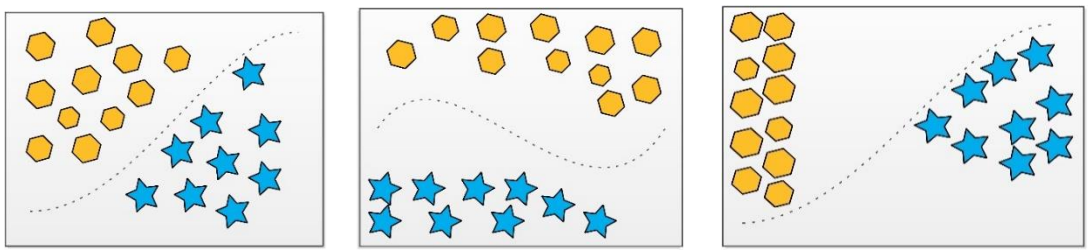


Figure 6.1: Types of Concept Drifts

There is currently no comprehensive knowledge of the variations in the effects of these variants of drift on classifier. Consequently, current streaming algorithms consider virtual drifts in the similar way as real drifts [176].



a) Original Data b) Real Drift c) Virtual Drift

Figure 6.2: Different Forms of Concept Drift in Streaming Data

Types of Concept Drifts on the basis of the speed

The number of epochs essential for a new concept to replace an old one is known as drift duration, also known as drifting time or drift breadth [177] and speed is inversely proportional to drifting time [178]. It signifies that a faster speed corresponds to smaller time steps. Concept Drifts can be classified as abrupt, gradual and recurrent, depending on their speed.

- *Abrupt (Sudden) Drift*: It happens when a novel concept takes the place of an outdated one. As the novel concept rapidly replaces the outdated one, this drift variant immediately degrades the model performance.
- *Gradual Drift*: This type of the drift takes place when the drift duration is prolonged. Since it generates an interval of complexity in steady states, this type of drift is more difficult to identify. Gradual Drift is classified into two categories that is Probabilistic and Continuous gradual drift.
- *Recurrent Drift:* A new concept or a recurring concept that has occurred at previous time steps of the data stream may develop again due to concept drift. For instance, On Black Friday Sale, people who shop moderately throughout the year display interesting patterns where the weekend mobility varies from that on working days. This is another classification of drifts that aims to elaborate previously occurred drifts that may occur again in the future or have trends to occur after a specific season. Recurrent drifts can have two types of behaviors that is cyclic and acyclic. The appearance of recurring concepts might be sudden or gradual, and they can affect the instance space locally or

globally. During the transition phase, several different characteristics of drift might be detected.

- *Incremental Drift:* It is also known as continuous drift, and entails a long drift duration. In this case, the progression from concept P to concept P' is gradual. The time steps taken by an evolving concept to replace an outdated concept entirely is the speed, or duration, of concept drift.

6.1.3 State-of-the-art Concept Drift Detection Techniques

There are a variety of ways to keep track of concept drift, as listed below:

a) The probability distribution of data is checked to detect concept drift because it may alter with time due to the factors such as distribution, noise, outliers, sensor delay, etc.

b) It is possible to check the occurrence of concept drift. It is essentially done by keeping track of the similarities and differences between various sample features or qualities.

c) The classifier's accuracy suffers as a result of concept drift. Consequently, it might be one of the measurements used to detect drift in a stream.

d) To determine the presence of concept drift, one of the input attributes that can be used is the timestamp. The timestamp function can be applied to a single sample or a group of samples [179].

There are mainly three types of techniques that have been developed to address concept drift:

1. Adaptive Base Learners
2. Training set modifying learners
3. Ensemble

This research focuses on Ensemble based techniques, therefore, for simplicity, the techniques that are studied for the development of proposed approach are all based on ensemble techniques which are discussed in section 2.4.1 of this thesis.

6.2 Proposed Approach

This section describes the approach that is proposed to solve the challenges of real-time detection of concept drifts in data streams using selective dynamic ensembles and transfer learning concepts. The architectural diagram of proposed approach selective ensemble classifier using transfer learning (SETL) is shown in Figure 6.3. Both real and synthetic data streams are used for training and validation of the proposed approach. All the datasets contain different types of concept drifts so that the models can be trained efficiently. An algorithm for the proposed model is implemented using Python Programming Language on Jupyter Notebook. The experiments are conducted on a machine having a configuration of 8GB RAM and a CPU with Intel Core i7 processor at 3.4 GHz and is presented in terms of CPU time.

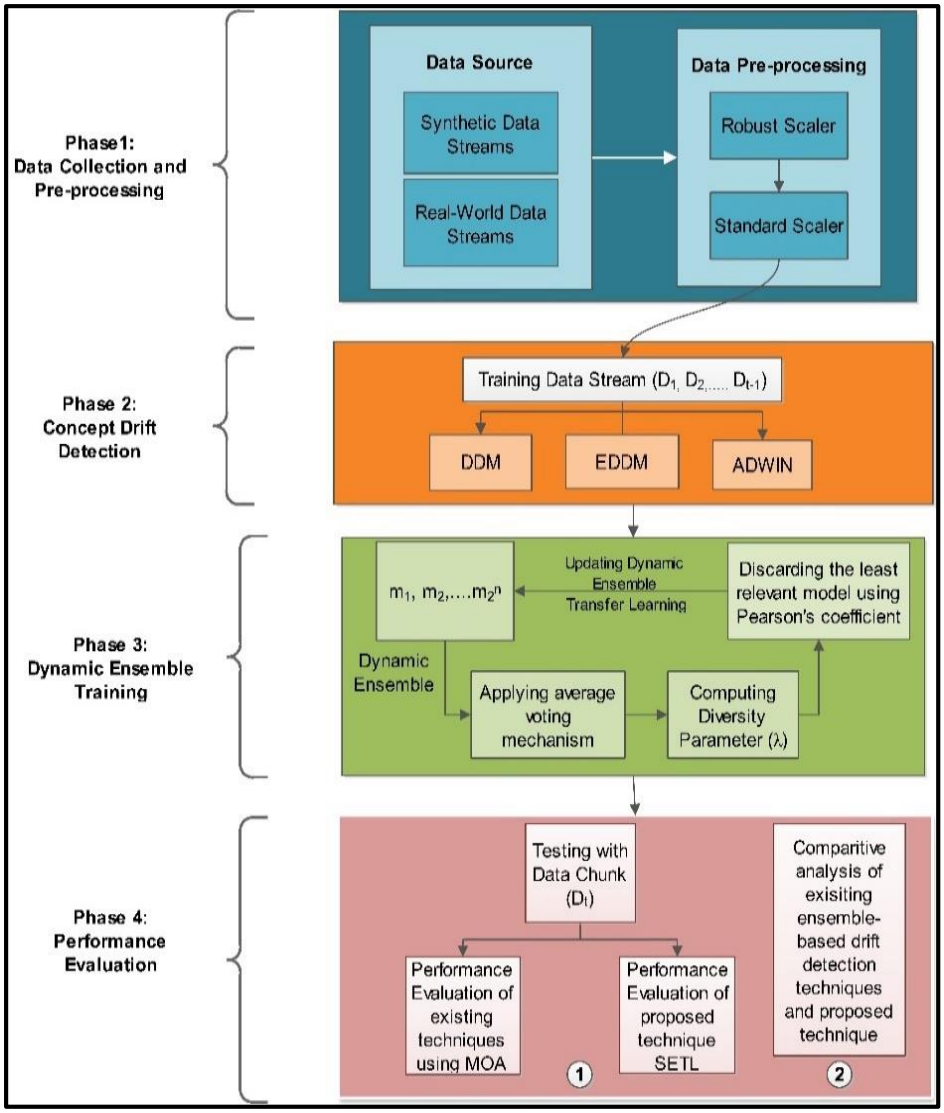


Figure 6.3: Architecture of the proposed approach- SETL

6.2.1 Phase 1: Data Collection and pre-processing

In this primary step, it is highly unlikely to acquire enough real-world streaming data for research, particularly one that has significant concept drifts. Moreover, the performance of a concept drift detection model cannot be exhaustively validated using real-world data sets alone. There is an utter requirement of artificial datasets

that includes diverse concept drifts that is sudden (instant/abrupt) or gradual (slow) to evaluate these models accurately. In this chapter, the experiments are carried out utilizing ten datasets that are available to the public. Amongst those, there are five artificial and five real-world datasets that can be downloaded from several github repositories.

To improve the quality of the data, some basic pre-processing operations such as addressing empty and superfluous values, encoding categorical and the nominal properties, and removing outliers and noisy data are carried out. The robust scaler is used to tackle missing values and eliminate outliers and noise. It is necessary to get rid of noisy data, thus the records that do not have a timestamp are eliminated. Robust Scaler employs the interquartile range for pre-processing the data by eliminating median and scales it as per the interquartile range [180]. The formula for calculating the interquartile range may be found in Eq. 6.3, and the interquartile range itself falls somewhere between the first and third quartiles of each dataset.

$$R=\frac{y_i-Q_1(y)}{Q_3(y)-Q_1(y)} \tag{6.3}$$

6.2.2 Phase 2 and 3: Concept Drift Detection and Training Dynamic Ensembles

For concept drift detection in data stream settings, the proposed approach, SETL uses transfer learning to build dynamic ensemble classifiers. SETL's transfer learning procedure is shown in Figure 6.4. The modules implemented in SETL include concept drift detection, training of existing ensemble models, evaluation of the existing ensemble model considering multiple performance metrics, diversity

calculation, elimination of irrelevant models, and transfer of the best-performing model to the dynamic ensemble used to test the new data chunk. The MOA framework is used for testing and comparison by integrating the proposed dynamic ensemble classifier in it.

Existing state-of-the-art concept drift detectors namely DDM, EDDM, and ADWIN are used to identify the drift in the data chunks of synthetic data streams. Further, Hoeffding Tree classifier is used as a base model with existing ensemble models like AWE and AUE2. The procedure for building up dynamic ensembles is discussed in detail in Algorithm (6.2) and in later subsections.

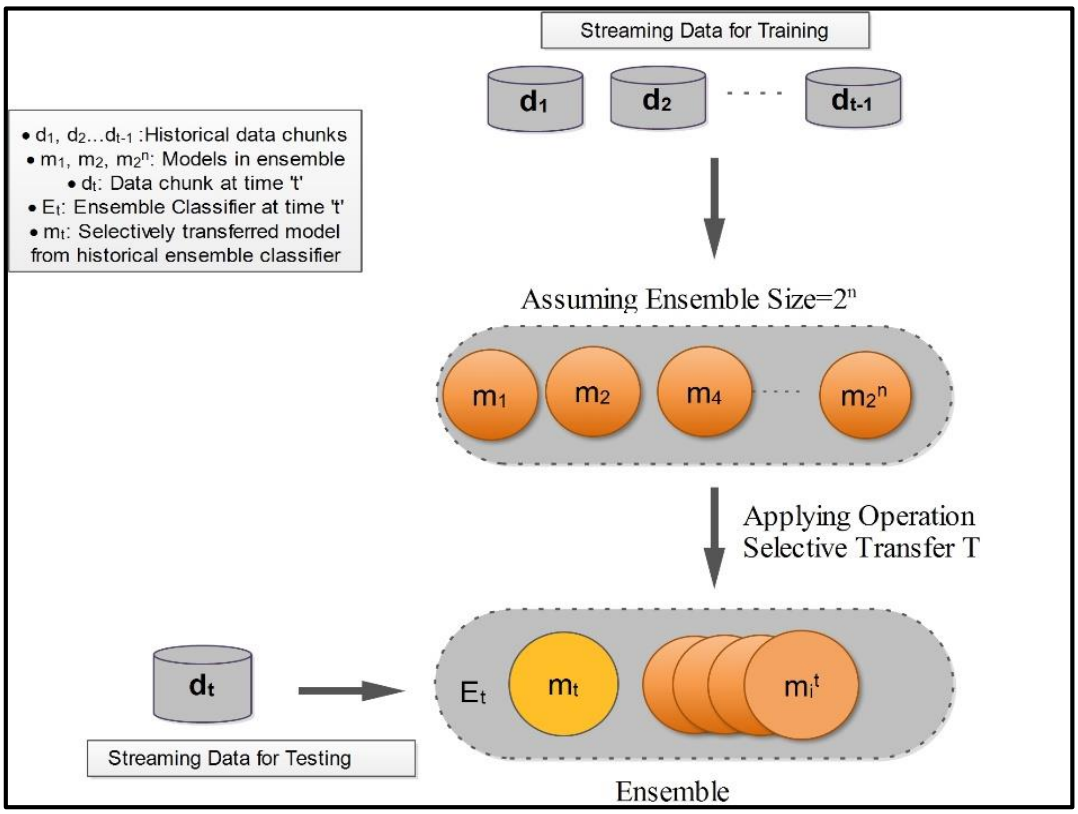


Figure 6.4: The Transfer Learning Process of SETL

Constructing Dynamic Ensembles

The appropriate models are chosen in advance to add them into an ensemble. The main task to construct an ensemble is to calculate the weights and votes of an existing classifiers. Online bagging and online boosting are two common ways of building an ensemble. However, the proposed approach mainly focuses on the online bagging technique and it leverages Hoeffding Tree as a base classifier. The predictions for each instance 'x' are integrated using a function 'f' after an initial ensemble of N learners from L_1 to L_N have been provided. Where $L_i(x)$ is L_i's prediction for x, the ensemble predicts $f(L_1(x)...L_N(x))$. Each model's vote (L_i) carries a certain amount of weight (w_i). The existing ensembles that are trained using datasets considered are mentioned below. These ensemble models are used in the state-of-the-art research and have proven excellent results. Therefore, these are used for the initial training of the drift detectors and hence, are dynamically modified as the concept drifts arrive.

a) Accuracy-Weighted Ensembles (AWE)

This approach implemented by [181] as a way to use non-stream learning models to mine streaming data. To achieve this, the stream of chunks is processed and a new classifier is created for each new data chunk. Every time a new data chunk is added, the model has to begin the process from scratch to analyze it. Eq. 6.4 is used to compute the weight "w", "Acc" is the weight of accuracy and "λ" is the diversity measure of each classifier C.

$$\boldsymbol{w = \beta * Acc + (1 - \beta) * \lambda} \tag{6.4}$$

b) Online Bagging

Another concept that has been considered for constructing dynamic ensembles is Online Bagging which proves to be quite efficient in tuning the diversity of the ensemble. Diversity Parameter (λ) can be easily re-configured to increase or decrease the diversity of the ensemble. Online Bagging is simpler as compared to Online Boosting as it involves only one operation that is the modification of λ. It can be said that the only source of variation apart from the λ is the variation in dataset. When trained with identical distributions of training samples, the basic model learning techniques of online bagging and batch bagging produce equivalent hypotheses.

Θ^a_b Defined as a vector of length N, where the ith element denotes the sheer number of times the ith initial training sample has been included in the bootstrap training set of the a^{th} base model when batch bagging is being used. The batch bagging algorithm samples with replacement by running N trials, with each trial producing one of the N training examples, all of which have an equal probability of being drawn $\frac{1}{N}$. Therefore, m where each training sample has an equal chance of succeeding $\frac{1}{N}$. Definitely the internet bagging equivalent of Θ^a_b.

Each training instance is selected for online bagging a certain number of times in accordance with a Poisson(1) distribution. The total number of instances drawn has a Poisson(N) distribution because there are N training examples and N successful trials. Sampling is recasted in the online bagging algorithm as executing N'~Poisson(N) trials where each h trial produces one of the N training examples, all of which h have an equal probability $\frac{1}{N}$ of being drawn. This is done because each h example has an equal likelihood of being drawn. Hence,

$$\Theta_o \sim \sum_{t=0}^{N} \boldsymbol{P(Poisson(N) = t) Multinomial(t, \frac{1}{N})}$$

c) ADWIN Bagging

This ensemble method proposed by Oza and Russell [109] is used as drift detector and an estimator for the weights in ADWIN Bagging, which is again based on online bagging method. ADWIN keeps a window of recently seen data that can be any length. The window has a maximum length that is statistically consistent with the statement that "the average value inside the window has not changed". ADWIN is based on the idea of a moving window, which changes its size in response to changes in the data stream. It tracks the performance of a certain number (like the error rate, mean, or standard deviation) over time and correlates it to a threshold that can be changed on a regular basis. When a change is found, the window is partitioned into two, and the existing section is pruned. ADWIN Bagging initially offers the training to the base ensemble models by leveraging a concept drift detector to check whether the underperforming models in the ensemble are to be discarded. The mechanism of ADWIN Bagging is illustrated in Algorithm 6.1.

A Poisson distribution has a mean of and a variation of λ. If λ= 1, it is observed that 35% of the instances are zero, 36% are one, while 29% are greater than one. However, considering Poisson(1) weight, it is easy to avoid 35% of the cases and replicate 26% of them. If λ= 6, it can be observed that 0.35% of the integers are 0, 42 % are less than 6, 19 percent are six, and 39 percent are more than six. By setting λ> 1 for Poisson(), the input space of the classifiers in the ensemble is updated and the weights become more diverse. However, the optimum value for λ may be distinct for every single type of dataset.

Algorithm 6.1 ADWIN Bagging [182]
1: Initialise base models h_n for all n Є {1,2,….. N} 2: for all training data instances do 3: for n= 1,2,3……..N do 4: Set *w* = *Poisson*(1) 5: Update h_n with the latest instance having weight *w* 6: Dynamic Output 7: return hypothesis: $h_{f\ in}(x) = \arg\max_{y \in Y} \sum_{t=1}^{T} \mathbf{I}(\boldsymbol{h_t}(\mathbf{x}) = \mathbf{y})$

Discarding the least relevant model

An ensemble model must have at least one base classifier that remains unchanged. A negative transfer operation may occur if the model fails to behave effectively in the desired domain or doesn't operate the same way as it did in the original domain. This negative transfer process doesn't tell us anything useful because the model that was trained on a certain segment of data might not be right for the current segment. SETL first trains on the latest chunk of data and then uses Pearson's coefficient (Eq. 6.5) to compute the correlation between the retained historical model and the newly trained model. This is done to ensure that the ensemble accurately classifies the drifts as well as minimizes the cost of transferring less effective models. If the value of $\boldsymbol{P}_{c_x,c_y}$ is approximately equal to 0, it means that the preserved model is less relevant to the new data distribution.

$$\boldsymbol{P}_{c_x,c_y} = \frac{\sum(p_i-\overline{p})(q_i-\overline{q})}{\sqrt{\sum(p_i-\overline{p})^2 \sum(q_i-\overline{q})^2}} \tag{6.5}$$

Here c_x, c_y denoted the two classifiers, $\boldsymbol{P}_{c_x,c_y}$ ranges between -1 and 1.

The experiments are also performed with Adaptive Size Hoeffding Tree (OzaBagASHT) [183], which is already integrated in the MOA tool. In ASHT, each hoeffding tree classifier in the ensemble has a certain value that shows the maximum

number of internal nodes it can reach. If a node is split in a tree and its size is bigger than the associated value, then tree is reset to a leaf.

In this work, an ensemble of N Hoeffding Tree classifiers with sizes that are powers of 2 (2^1, 2^2, 2^3,..., 2^m) are taken into account. The reason behind choosing size having power of 2 is because small sized trees can adjust to changes more quickly, while bigger trees, which are built with a lot of data, work better when there is little or no change. Trees with a depth of ‘s’ will be reset nearly twice as often as trees with a depth of '2s'. This leads to a set of different restart speeds for a group of these kinds of trees. It is important to note that restarts happen every time, even in static datasets, but this won't change how well an ensemble predicts and may even help with drift recovery.

6.2.3 Phase 4: Implementation of Transfer Learning Concepts

Knowledge from a historical data can be valuable while learning in the context of slow or gradual concept drifts. When there is a gradual concept drift, data instances that follow the previous data distribution are still useful for a while until the drift is over. Furthermore, if there is relatively small concept drift (if the past and new concepts are similar in some aspects), the patterns from the old concept may be useful in learning the new concept. This reinforces the theory that Transfer Learning approaches may be relevant when the source and target data is identical. The complexity to update an historical model with recently acquired information relies on the machine learning model being used. SETL's preset learner is a Hoeffding tree. A knowledge transfer method is used to make transformation to a Hoeffding tree that has already been trained. Every area of a feature space has a class

identity that belongs to a leaf node of the tree. The proposed approach utilizes transfer learning concepts to update the transfer the parameters of the base Hoeffding Tree to the dynamic ensemble created each time when new data chunk arrives and accommodate current data chunk while preserving the structural integrity. The transfer learning task can be achieved by the following mentioned steps:

Step 1*: Retaining the knowledge learned in the base Hoeffding tree*

Update the class labels of each node so that they are aligned with the instances situated within the leaf nodes of the new data chunk D_t. It is essential to have the same structure of the modified Hoeffding tree to that of original one in the initial step.

Step 2: *To satisfy the requirement of correctly identifying data in chunk* D_t

If the termination condition is not met at a leaf node, training continues on a child tree.

Step *3: Reducing the cost of Transfer/ Negative Transfer*

If the archived historical model does not accommodate the incoming data, the transfer operation is futile. However, if the model has been transferred, it may result in overfitting. A novel parameter known as transfer gain is computed to identify the model that is not transfer-effective. The process entails acquiring the tree's leaf nodes using data chunk D_t. By computing the transfer gain function given in Eq. 6.6, the number of leaf nodes is computed to validate if the transfer has occurred. The expected transfer gain is denoted by Tgain. LC represents the number of labels, SP represents the number of leaves that continue splitting, and N represents the total number of leaves.

$$T_{gain}=(\boldsymbol{LC}+\boldsymbol{SP})/\boldsymbol{N} \quad (6.6)$$

Transfer Gain parameter assists in avoiding the excess operation of less transfer-effective model computed in Eq 6.6.

Algorithm 6.2. SETL

Input: $d_1, d_2, ..., d_t$: chunk of data stream, M_t: the group of retained historical models at time t, C: Cardinality of historical models that have been preserved
Result: E_t : the ensemble model at time t
1: for set of data fragments d_t do
2: m_t : new component classifier trained by d_t
3: for m_i in M_t do
4: if m_i is a model that is less relevant then
5: if m_i is old model added in time (t-1) then
6: $m_i^t \leftarrow m_i^{t-1}$
7: else
8: $m_i^t \leftarrow m_i$
9: else if m_i is less transfer effective model then
10: $m_i^t \leftarrow m_i$
11: else
12: $m_i^t \leftarrow$ transfer model m_i with d_t
13: $\lambda_i^t \leftarrow$ compute model m_i^t
14: end for
15: if $|M_{t-1}| < C$ then
15: $C_t \leftarrow C_{t-1} \cup m_t$
16: else
17: $m_w \leftarrow$ the least relevant model in Mt
18: replace m_w with m_t in M_t
19: $\boldsymbol{E_t} = (\sum_i \boldsymbol{w_i^t f_i^t} + \boldsymbol{f_t})/(\sum_i \boldsymbol{w_i^t} + \boldsymbol{1})$
20: end for

6.3 Experimental Evaluation

6.3.1 Streaming Datasets with Concept Drifts

The datasets have been collected from streaming sources with timestamps. In this research, both real-world and synthetic datasets have been considered for experimental evaluation of the proposed approach. The real-world datasets alone are

not enough to evaluate the performance of the proposed approach since they may or may not have concept drifts. Therefore, synthetic datasets are generated with both abrupt and gradual drift types so that SETL is evaluated in all scenarios.

a) Synthetic datasets

For a concept drift detection technique to be effective, the dataset must have unique characteristics like:

- have different kinds of drifts so that the proposed framework can be tested in a wide range of settings.
- include both noisy and noiseless samples.
- must be entirely unique.

The synthetic data streams generated using stream generators from MOA tool are mentioned below. All the stream generators can easily be used by importing from skmultiflow.data. Some unique characteristics of the synthetic datastreams are mentioned in Table 6.1.

i) SEA: This generator is a data stream implementation with sudden concept drift. Only two of the three number attributes, which range from 0 to 10, are significant to the classification job. In this work, a training set of 100,000 instances and a test set of 50,000 instances is generated by SEA generator using the MOA tool. There is a noise of around 10% in both the sets.

ii) *LED*: This generator has a specific setup that yields 24 binary features, 17 of which are meaningless. In this work, this generator is used along with the concept drift feature that introduces drifts after a specified number of

instances for experimental use. It is a noisy database with a noise range fluctuating between 5-30%.

iii) *Random RBF*: This generator presents a stream of radial basis functions. A fresh sample is made by randomly selecting one of the centroids, accounting for their weights, and offsetting the attributes in a random direction from the centroid's centre. It has 100,000 instances with 10 features and 2 classes.

iv) *Waveform*: Based on a random differentiation of various base waveforms, it generates instances with 21 numeric features and 3 classes.

v) *Hyperplane*: This generator is used with parameters: Instances: 100,000, Dimensions:2 delta:0.001

Table 6.1: Summary of features of Library generated synthetic datasets*

Feature	Abrupt	Gradual	Noise	DataStream Length
SEA	Y	N	Y	1,00,000
LED	Y	Y	Y	10,000
RandomRBF	Y	N	N	1,00,000
Waveform	Y	Y	N	10,000
Hyperplane	Y	N	Y	1,00,000

**Source: Extracted using skmultiflow-library of ML*

b) Real-world (open source) data sets

In addition to the synthetic datasets, five real-world datasets are used for experiments in this work. The features of the datasets are summarized in Table 6.2.

i. *CoverType*: This dataset includes 54 attributes that define different types of forest cover. It contains 581,012 instances, each of which describes one of

seven forest cover types for 30 × 30 meter cells, as acquired from the US Forest Service's Resource Information System (RIS).

ii. *ELEC (Electricity Market data)*: ELEC is obtained from the Electricity Market in New South Wales, Australia. It is about power prices fluctuating in response to market demand and supply. From May 1996 through December 1998, this data collection contains 45,312 instances and one day having 48 instances.

iii. *PokerHand*: The problem of identifying the hand in a Poker game is denoted in this dataset. It contains 1,025,010 instances representing all feasible poker hands, with each instance depicting a hand made up of five cards drawn from a standard 52-card deck. Two characteristics are represented by each card in your hand (suit and rank). Each hand is described by ten characteristics.

iv. *Weather*: The National Oceanic and Atmospheric Administration of the United States has accumulated weather data from over 9000 weather stations worldwide [184]. Various parameters (temperature, pressure, wind speed, and so on), as well as indicators for precipitation and other weather-related occurrences, are included in daily measurements.

v. *Phishing*: This dataset is downloaded from the UCI Machine Learning Repository and contains 31 features with binary or ternary values. It has 11,055 records, each with 31 unique attributes.

Table 6.2: Summary of features of real-world datasets used in this work

Feature / Dataset	Abrupt	Gradual	Recurrent	Imbalance	High Dimensionality
Covertype	Y	N	N	N	Y
ELEC	Y	N	Y	N	Y

PokerHand	N	Y	N	N	N
Weather	N	Y	Y	Y	N
Phishing	Y	N	N	N	Y

6.3.2 Working of SETL

Assume that the data chunks D_1... D_t arrive in sequential order. In SETL, the Hoeffding Tree functions as the base learner. As soon as the first segment of data comes in, the base learner 'm1' is trained on it and is preserved in an entity. An additional model is constructed from scratch and the archived model is retrained on the newly arrived data chunk D_t. An ensemble model, including both the preserved and scratch models, is constructed at time step t. The model's retention or replacement is determined using a weighted majority vote process. In the context of incremental learning in ensembles, it yields satisfactory results.

Each preserved model's weight is calculated independently using Eq. (6.7) in a weighted majority approach. Eq. (6.8) is used to calculate the weight of each currently trained model.

$$\boldsymbol{W_r^t = \frac{1}{E_r^t + E_i^t + \epsilon}} \tag{6.7}$$

$$\boldsymbol{W_t = \frac{1}{E_r^t + \epsilon}} \tag{6.8}$$

the prediction error of the maintained tree m_i on data chunk D_t is given by E_i^t, and the mean square error of a random model is given by E_r^t so that the numerator is not zero. The posterior probability in the tree model is based on the proportion of that

class in that particular leaf node. Once the archive of historical trees is full, it determines whether a newly trained tree will take the place of a preserved tree in the archive.

6.3.3 Experiments with Synthetic Data Streams: Abrupt concept drift

A thorough experimental analysis is performed, and the proposed approach is is evaluated in comparison with state-of-the-art ensemble methods (AES [33], AUE2 [4], AWE [38], KUE [34], ARF[35], Melanie [25], HE-CDTL [27], DTEL [24], and ACDC [26]). These existing techniques were chosen because they demonstrated robust ensemble approaches for drift detection in real-time data. To further verify the effectiveness of selecting ensemble methods to address the issue of drift detection in real and synthetic datasets, SETL is further compared with existing drift detectors (Table 6.3 and Table 6.5) that employ approaches other than ensemble-based. The MOA framework provides an implementation for some of these existing methods, therefore it is used to evaluate how well they perform on the 10 chosen datasets. The experiment uses the Hoeffding Tree (HT) classifier as the base classifier for SETL across all ensembles. Currently, the experiments are configured in MOA with the underlined parameters:

- The instance limit is set to : 100,000.
- The method of evaluation chosen is : Prequential evaluation

Since the static streams are based on real-world data, they are subject to their own instance restriction. They strictly adhere to a maximum number of active instances. The experiments are run in MOA utilising real-world datasets such as CoverType,

Pokerhand, Electricity, Weather, and Phishing. In terms of accuracy, SETL consistently surpasses competing algorithms.

The data stream generators explored pertaining to the HT classifier with concept drift are LED, RandomRBF, Waveform, and Hyperplane. Table 6.4 presents the results of extensive experiments conducted with both real and synthetic data streams, demonstrating that SETL achieved higher accuracy when compared to existing drift detectors taking into consideration both types of datasets.

The experiments for SETL (proposed classifier) are conducted in two modes, namely Abrupt Concept Drift Detector and Gradual Concept Drift Detector, on synthetic data streams (SEA, Random RBF, LED, Waveform, Hyperplane) and real-world datasets, respectively. The plots in Figure 6.5 demonstrates that utilising the proposed dynamic classifier with various synthetic data streams, accuracy and kappa measurements produce outstanding results. Despite an increase in the number of streaming instances, the results achieved remained consistent. Thus, the proposed approach does not depend on the quantity of instances present in synthetic data streams.

Table 6.3: Performance comparison of proposed approach with state-of-the-art approaches: EDDM, DDM, ADWIN, STEPD on real-world datasets

Approach	Results	Covertype	PokerHand	Electricity	Weather	Phishing
EDDM [185]	Accuracy (%)	85.95	49.88	84.91	50.00	92.78
	Kappa (%)	77.38	-0.02	69.08	-25.64	85.31
	Kappa-T (%)	-184.50	11.52	-2.86	0	85.15
	Time (ms)	10.58	0.27	0.62	0	0.17
DDM [186]	Accuracy (%)	89.87	50.40	86.60	50.00	92.75
	Kappa (%)	78.71	0.26	73.04	-25.64	85.25
	Kappa-T (%)	-161.26	15.36	5.63	0	85.07
	Time (ms)	7.39	0.38	1.39	0	0.17
ADWIN [187]	Accuracy (%)	87.81	49.70	84.08	69.5	93.40
	Kappa (%)	78.11	-0.17	66.94	-34.04	86.58

	Kappa-T (%)	-126.40	11.20	-8.55	-28.57	86.41
	Time (ms)	384.19	10.50	25.36	0.05	6.39
STEPD [188]	Accuracy (%)	84.26	49.88	81.78	50.00	92.44
	Kappa (%)	72.16	-0.01	62.04	-25.64	84.62
	Kappa-T (%)	-164.31	11.52	-24.21	0	84.44
	Time (ms)	6.28	0.23	0.39	0.02	0.17
Proposed (SETL)	Accuracy (%)	88.97	76.11	87.74	72.14	93.48
	Kappa (%)	79.11	0.37	77.59	-34.57	86.73
	Kappa-T (%)	-124.56	16.45	-	0	86.25
	Time (ms)	5.32	0.01	0.23	0	0.10

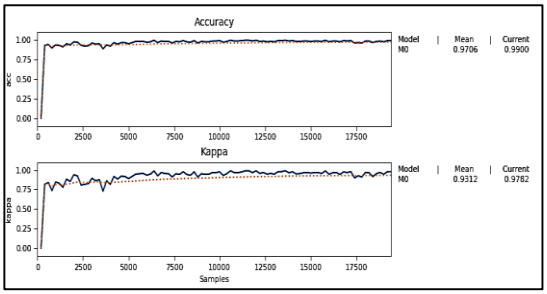
Table 6.4: Accuracy (in %) comparison of proposed approach with state-of-the-art ensemble techniques for concept drift detection under non-stationary environment

Dataset	Ensemble Technique								
	AES [183]	AUE2 [103]	DTEL [189]	AWE [190]	KUE[191]	ARF[94]	Melanie [192]	HE-CDTL[193]	Proposed(SETL)
Covertype	79.37	83.83	91.80	77.27	84.01	80.52	88.37	76.45	90.97
Poker	49.66	49.67	51.20	31.45	49.32	50.82	44.66	47.50	66.11
ELEC	78.87	78.04	75.05	63.02	78.94	79.62	75.37	73.57	81.74
Weather	73.33	73.34	74.32	68.87	74.56	77.33	72.00	77.83	77.91
Phishing	92.78	93.20	96.21	93.40	92.44	93.75	91.87	93.00	96.74
SEA	87.66	86.23	85.28	88.71	82.13	88.93	93.50	83.33	91.76
LED	74.01	73.98	72.59	70.52	72.57	74.33	70.10	71.90	76.63
RandomRBF	94.12	94.65	94.50	80.19	94.47	96.33	93.12	89.55	97.06
Waveform	83.86	83.36	83.35	83.31	84.20	79.74	82.35	80.30	84.20
Hyperplane	89.09	91.88	85.77	65.08	92.08	80.74	87.30	89.00	92.74

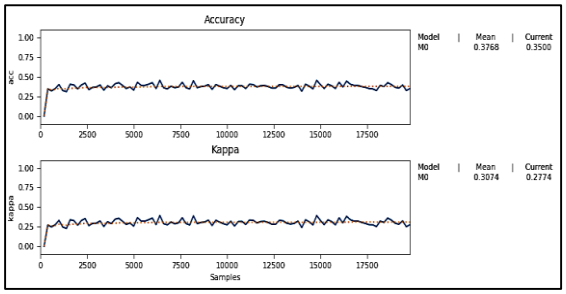
Table 6.5: Comparison of proposed approach with state-of-the-art approaches: EDDM, DDM, ADWIN, STEPD on synthetic datasets: Abrupt Concept Drift

Approach	Metric	SEA	LED	Random RBF	Waveform	Hyperplane
EDDM [185]	Accuracy (%)	87.66	74.01	95.02(1h 11m)	80.94	88.65
	Kappa (%)	72.42	71.12	90.04	71.42	77.30
	Kappa-T (%)	73.14	71.11	90.03	71.24	77.36
	Precision (%)	87.48	74.51	95.01	81.13	88.63
	Recall (%)	85.28	74.00	95.02	80.97	88.64
	F-1 Score (%)	86.36	74.25	95.01	81.05	88.64
DDM [186]	Accuracy (%)	88.34	73.98	85.39	81.62	88.75
	Kappa (%)	73.99	71.09	70.78	72.44	77.51
	Kappa-T (%)	74.62	71.09	70.64	72.27	77.57
	Precision (%)	88.16	74.03	85.38	81.77	88.74

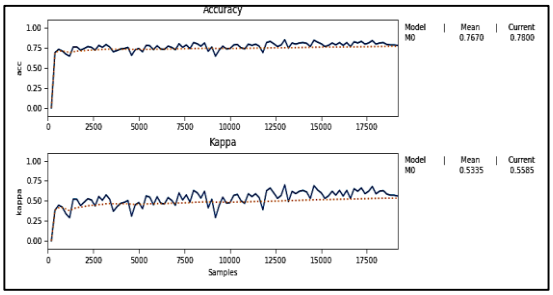
	Recall (%)	86.11	73.98	85.39	81.64	88.75
	F-1 Score (%)	87.13	74.00	85.39	81.70	88.75
ADWIN [187]	Accuracy (%)	88.34	32.59	85.39	37.59	51.31
	Kappa (%)	73.99	25.10	70.78	6.15	2.81
	Kappa-T (%)	74.62	25.12	70.64	5.83	2.89
	Precision (%)	88.16	32.58	85.38	62.50	69.51
	Recall (%)	86.11	32.59	85.38	37.42	51.40
	F-1 Score (%)	87.13	32.59	85.39	50.86	59.10
STEPD [188]	Accuracy (%)	86.72	70.52	77.66	80.31	89.97
	Kappa (%)	70.10	67.24	55.31	70.49	79.94
	Kappa-T (%)	71.01	67.25	55.09	70.29	80.00
	Precision (%)	86.75	70.48	77.67	83.20	89.97
	Recall (%)	83.90	70.52	77.65	80.38	89.97
	F-1 Score (%)	85.30	70.50	77.66	81.77	89.96
Proposed (SETL)	Accuracy (%)	97.06	76.63	89.75	84.20	90.20
	Kappa (%)	73.98	72.25	88.03	82.44	80.94
	Kappa-T (%)	74.01	73.10	85.01	82.27	81.33
	Precision (%)	96.00	76.00	86.02	83.77	89.09
	Recall (%)	99.00	73.75	85.01	83.64	89.17
	F-1 Score (%)	98.00	76.59	85.01	83.70	90.96



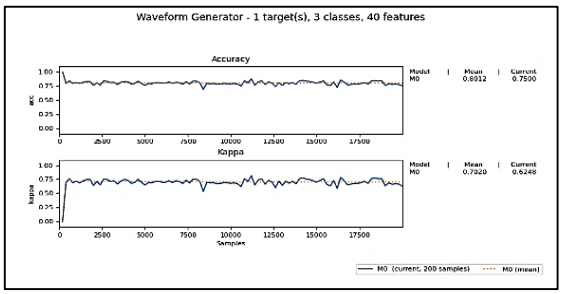
a) SEA Data Stream



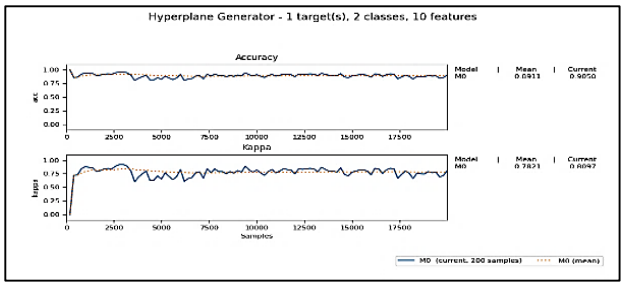
c) LED Data Stream



b) RandomRBF Data Stream

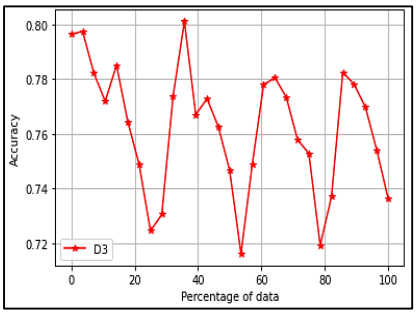


d) Waveform Data Stream

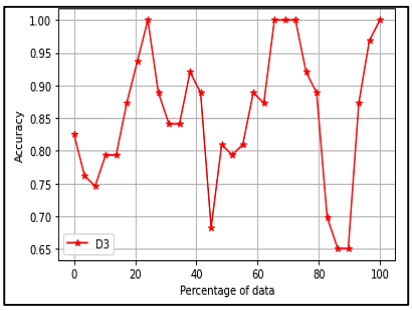


e) Hyperplane Data Stream

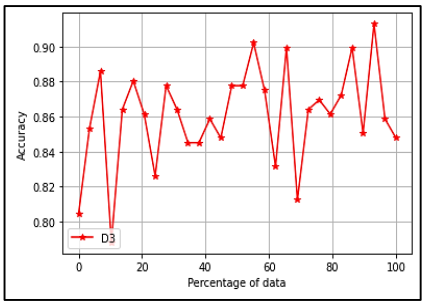
Figure 6.5: Experimental results for average accuracy and Kappa-measure with synthetic data streams



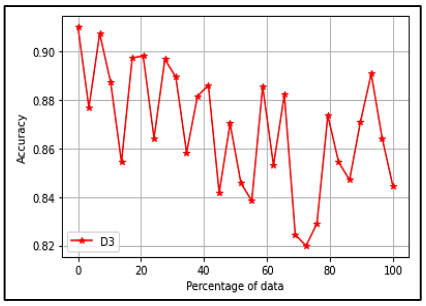
a) CoverType



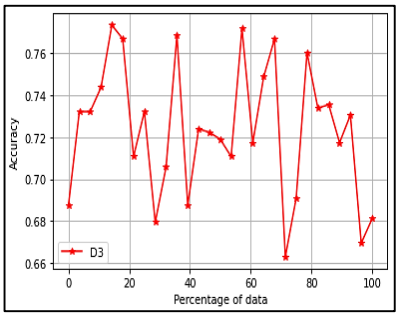
b) PokerHand



c) Weather

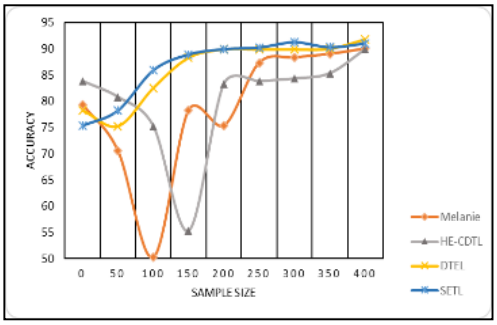


d) Phishing

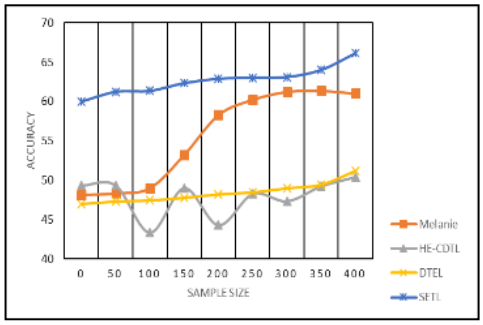


e) ELEC

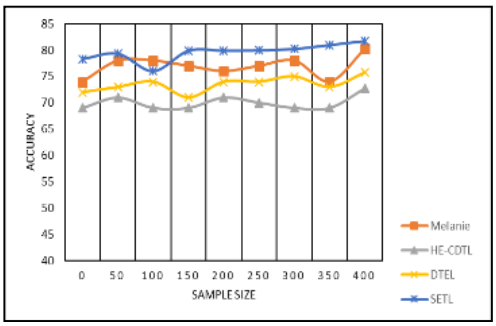
Figure 6.6: Experimental results for real data streams



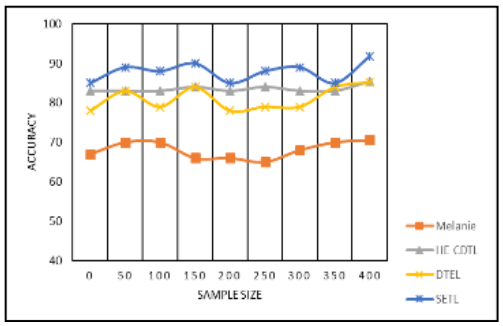
a) Covertype



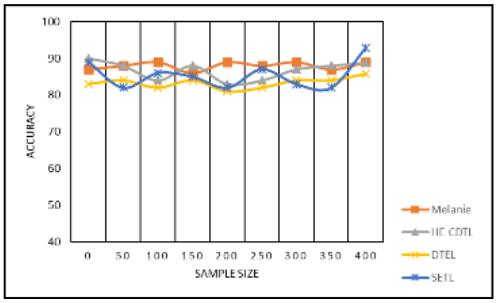
b) Pokerhand



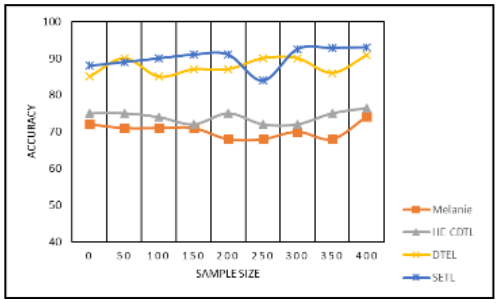
c) Electricity



d) SEA



e) Hyperplane



f) Rotating Concepts

Figure 6.7: Accuracy comparison of SETL with existing techniques on both real and synthetic datasets

6.3.4 Experiments with Real-World data: Gradual Concept Drift

On the contrary, experiments in the Gradual Concept Drift Detector configuration of SETL are conducted with real-world datasets that are often used in the

existing research. These datasets are taken into account while evaluating the proposed method since they exhibit gradual concept drift.

The performance metric: accuracy results are demonstrated in Figure 6.6. It is evident that the accuracy changes as the percentage of data increases. It is caused due to the data's slowly drifting patterns. Thus, it can be concluded that the number of instances in real-world data streams is a key factor in determining accuracy.

Furthermore, various experiments are performed on both synthetic and real data sets comprising both abrupt and gradual drifts for accuracy comparison of the proposed technique with certain contemporary techniques. From the results shown in Figure 6.7, it is clear that SETL achieves higher accuracy than some of the other techniques on some specific datasets. However, for few other datasets, the outcome is remarkably comparable.

6.4 Chapter Summary

This chapter comprises of a novel approach proposed to address concept drifts in streaming data with a unique selective transfer learning technique for dynamic ensembles (SETL). The approach offers drift adaptation to sudden as well as gradual drifts. To prevent the same set of models from being used repeatedly in a dynamic ensemble, it additionally carries a diversity parameter. Experiments on both synthetic and real-world datasets demonstrate that the proposed approach yields a reliable model selection in the form of an ensemble for enhanced classification and precision. Additionally, by curbing the transfer of less pertinent models, SETL's selective technique avoids negative transfer and overfitting problems. SETL outperforms

conventional machine learning algorithms in terms of evaluation measures including accuracy, kappa, kappa-t, F1-score, and recall.

SETL performs better and uses resources effectively when compared to other contemporary ensembles. From the experimental evaluations in parametric conditions, it can be deduced that SETL not only minimizes the transfer runtime but also limits the overhead of model transfer.

In the future, the focus of study will be on the automatic setting of statistical parameters for different types of drifts and to effectively manage recurring concept drifts

Chapter 7

CONCLUSION AND FUTURE DIRECTIONS

This chapter concludes the thesis by summarizing the crucial findings, research contributions,

outcomes and the scope of future work in the domain of event detection in streaming data. The research contributions and its outcomes are discussed in Section 7.1 and future directions in this area are presented in Section 7.2.

7.1 Main Contribution

This research has made a significant contribution in the domain of streaming data analytics. It can be said that transformation and event detection techniques implemented in this research will prove to be an effective way for business analysts and decision-makers draw useful insights in real-time when the data is continuously flowing. It may also help the machine learning developers to update their models according to the changing distributions of datasets. An in-memory processing based Apache Spark framework is used for processing of data streams. Usually, the streaming data comes in different formats and requires transformation before it can be fed to processing frameworks or can be used to train machine learning models. Further, the data can be well-analyzed only if there are some critical events or abnormal values in the data to be detected in real-time. Hence, an efficient approach for transformation and analysis of data streams is proposed in Chapter 4. Real-life application considered for implementing this approach is based on classification of tweets into rumor and non-rumors during pandemic times. Rumorous tweets are

detected and classified further into 3 categories. The proposed approach uses customized stateful transformations to transform the streaming data and based on these transformations existing VADER classifier is improvised to classify the rumors into further categories. The proposed approach performs better than similar existing approaches for rumor detection in terms of classification accuracy and time efficiency.

Further, in order to detect critical events in data streams 2-component-offline-online approach utilizing deep learning models along with streaming frameworks is proposed. To prove the effectiveness of the approach, real-world data streams from PubNub sensors are used firstly to train RNN and LSTM model in offline component. Secondly, in the online component, stream processing frameworks named Kafka and Spark Streaming are utilized along with the best trained model from offline component to detect events in real-time. This approach tried to enhance the accuracy and latency of the existing approaches those either detect the events in offline or online mode. However, the proposed combination of offline and online component has proven to be effective as it utilizes the time taken to re-train the offline model for every instance of data in a separate module and does not interferes with the processing time of online component. The proposed approach is validated using a dataset of ambient weather sensors, ingested using PubNub API and transformed using stateful approaches. The proposed approach is scalable and is able to detect events from other similar datasets efficiently. The performance of the proposed algorithm is compared with the existing algorithms.

Finally, the challenge of concept drifts that generally occur in models that are trained on streaming data is addressed. A concept drift occurs, when the statistical distribution

of the underlying data changes with time and the model is not able to adapt to these dangling patterns as it was trained on different trends. To overcome this challenge, a novel selective ensemble transfer learning approach (SETL) to detect and adapt to concept drifts in streaming data is proposed and validated on several real-world and synthetic data streams. The proposed approach effectively minimizes the negative transfer operation by introducing a new transfer gain operation and then selectively transferring only those models to the next dynamic ensemble which have more weights referring to a weighted majority voting scheme. To evaluate the performance of the proposed approach, several metrics such as Accuracy, F1-score, Kappa measure, Precision and Recall are computed and compared with similar existing state-of-the-art approaches.

It is assumed that this work will raise the research interest in combining data engineering and data-intensive technologies to improve event detection in data streams.

7.2 Future Directions

The research described in this thesis has a number of promising further directions for future research. The work might be explored for various applications that have streaming datasets like credit card fraud detection, financial market analysis, network analysis, public opinion analysis etc. for deducing interesting patterns. Further, there are various areas in different domains that need to be touched for detection of unusual events so that one can contribute towards the betterment of the society.

Fake News Detection

Similar to rumor detection, this application uses machine learning algorithms and transformation techniques to identify and combat the spread of incorrect information on social media platforms.

Event History Analysis

Events must first be detected in order to be analyzed, hence combining "Event History Analysis" with techniques for event detection is a significant problem that could be solved in the future.

Zero-Shot Learning

An interesting unexplored area in streaming data is Zero-Shot-Learning which means learning to identify the concept by just having a description of it. This would be extremely relevant to the streams with significant concept drift or streams with concept evolution.

Concept Drift Adaptation

Research on drift detection should not only concentrate on precisely defining the time of drift occurrence, but need to include some information about drift intensity and areas. These details might be used to improve adaptability of concept drift.

Preventive maintenance

Analyzing machinery and equipment data, streaming analytics can make predictive and preventive maintenance feasible. Organizations can proactively anticipate equipment defects, plan preventative maintenance, limit downtime, and conserve operational costs by identifying trends and abnormalities.

References

[1] L. M. Ang and K. P. Seng, "Big Sensor Data Applications in Urban Environments," *Big Data Research*, vol. 4, pp. 1–12, 2016, doi: 10.1016/j.bdr.2015.12.003.

[2] M. Dias de Assunção, A. da Silva Veith, and R. Buyya, "Distributed data stream processing and edge computing: A survey on resource elasticity and future directions," *Journal of Network and Computer Applications*, vol. 103, pp. 1–17, 2018, doi: 10.1016/j.jnca.2017.12.001.

[3] N. Tantalaki, S. Souravlas, and M. Roumeliotis, "A review on big data real-time stream processing and its scheduling techniques," *International Journal of Parallel, Emergent and Distributed Systems*, vol. 35, no. 5. pp. 571–601, 2020. doi: 10.1080/17445760.2019.1585848.

[4] E. Lughofer and P. Angelov, "Handling drifts and shifts in on-line data streams with evolving fuzzy systems," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 2057–2068, 2011, doi: 10.1016/j.asoc.2010.07.003.

[5] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, pp. 1–37, Apr. 2014, doi: 10.1145/2523813.

[6] S. Ramírez-Gallego, B. Krawczyk, S. García, M. Woźniak, and F. Herrera, "A survey on data preprocessing for data stream mining: Current status and future directions," *Neurocomputing*, vol. 239, pp. 39–57, 2017, doi: 10.1016/j.neucom.2017.01.078.

[7] L. Y. Wei and W. C. Peng, "An incremental algorithm for clustering spatial data streams: Exploring temporal locality," *Knowledge and Information Systems*, vol. 37, no. 2, pp. 453–483, 2013, doi: 10.1007/s10115-013-0636-8.

[8] B. Zhou, J. Li, J. Wu, S. Guo, Y. Gu, and Z. Li, "Machine-learning-based online distributed denial-of-service attack detection using spark streaming," in *IEEE International Conference on Communications*, 2018, pp. 1–6. doi: 10.1109/ICC.2018.8422327.

[9] A. C. Onal, O. Berat Sezer, M. Ozbayoglu, and E. Dogdu, “Weather data analysis and sensor fault detection using an extended IoT framework with semantics, big data, and machine learning,” in *Proceedings - 2017 IEEE International Conference on Big Data, Big Data 2017*, 2017, pp. 2037–2046. doi: 10.1109/BigData.2017.8258150.

[10] M. Helu, T. Sprock, D. Hartenstine, R. Venketesh, and W. Sobel, “Scalable data pipeline architecture to support the industrial internet of things,” *CIRP Annals*, vol. 69, no. 1, pp. 385–388, 2020, doi: 10.1016/j.cirp.2020.04.006.

[11] S. Salloum, R. Dautov, X. Chen, P. X. Peng, and J. Z. Huang, “Big data analytics on Apache Spark,” *International Journal of Data Science and Analytics*, vol. 1, no. 3–4, pp. 145–164, 2016, doi: 10.1007/s41060-016-0027-9.

[12] Q. Zhou, Y. Simmhan, and V. Prasanna, “Knowledge-infused and consistent Complex Event Processing over real-time and persistent streams,” *Future Generation Computer Systems*, vol. 76, pp. 391–406, 2017, doi: 10.1016/j.future.2016.10.030.

[13] A. Goswami and A. Kumar, “A survey of event detection techniques in online social networks,” *Social Network Analysis and Mining*, vol. 6, no. 1, pp. 1–25, 2016, doi: 10.1007/s13278-016-0414-1.

[14] “Real-time data streaming turns live data into insights.” https://www.pubnub.com/demos/real-time-data-streaming/

[15] N. Li, F. Chang, and C. Liu, “Human-related anomalous event detection via spatial-temporal graph convolutional autoencoder with embedded long short-term memory network,” *Neurocomputing*, vol. 490, pp. 482–494, 2022, doi: 10.1016/j.neucom.2021.12.023.

[16] G. F. Dar, T. R. Padi, S. Rekha, and Q. F. Dar, “Stochastic Modeling for the Analysis and Forecasting of Stock Market Trend using Hidden Markov Model,” *Asian Journal of Probability and Statistics*, vol. 18, no. 1, pp. 43–56, 2022, doi: 10.9734/ajpas/2022/v18i130436.

[17] A. Bakumenko and A. Elragal, “Detecting Anomalies in Financial Data Using Machine Learning Algorithms,” *Systems*, vol. 10, no. 5, p. 130, 2021.

[18] H. Tabbaa, S. Ifzarne, and I. Hafidi, “An Online Ensemble Learning Model for

Detecting Attacks in Wireless Sensor Networks," pp. 1–15, 2022, [Online]. Available: http://arxiv.org/abs/2204.13814

[19] J. Wang, R. Zhu, and S. Liu, "A Differentially Private Unscented Kalman Filter for Streaming Data in IoT," *IEEE Access*, vol. 6, pp. 6487–6495, 2018, doi: 10.1109/ACCESS.2018.2797159.

[20] M. A. P. Chamikara, P. Bertok, D. Liu, S. Camtepe, and I. Khalil, "An efficient and scalable privacy preserving algorithm for big data and data streams," *Computers and Security*, vol. 87, p. 101570, 2019, doi: 10.1016/j.cose.2019.101570.

[21] M. Zhang, J. Guo, X. Li, and R. Jin, "Data-driven anomaly detection approach for time-series streaming data," *Sensors (Switzerland)*, vol. 20, no. 19, pp. 1–17, 2020, doi: 10.3390/s20195646.

[22] C. Fernandez-Basso, A. J. Francisco-Agra, M. J. Martin-Bautista, and M. Dolores Ruiz, "Finding tendencies in streaming data using Big Data frequent itemset mining," *Knowledge-Based Systems*, vol. 163, pp. 666–674, 2019, doi: 10.1016/j.knosys.2018.09.026.

[23] A. Ghosh, A. Chakraborty, D. Chakraborty, M. Saha, and S. Saha, "UltraSense: A non-intrusive approach for human activity identification using heterogeneous ultrasonic sensor grid for smart home environment," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–22, 2019, doi: 10.1007/s12652-019-01260-y.

[24] T. H. Hamam and J. Romberg, "Streaming Solutions for Time-Varying Optimization Problems," *IEEE Transactions on Signal Processing*, vol. 70, no. 1, pp. 3582–3597, 2022, doi: 10.1109/TSP.2022.3188208.

[25] A. Glyn-Davies and M. Girolami, "Anomaly detection in streaming data with gaussian process based stochastic differential equations," *Pattern Recognition Letters*, vol. 153, pp. 254–260, 2022, doi: 10.1016/j.patrec.2021.12.017.

[26] M. Kristan and A. Leonardis, "Online discriminative kernel density estimator with gaussian kernels," *IEEE Transactions on Cybernetics*, vol. 44, no. 3, pp. 355–365, 2014, doi: 10.1109/TCYB.2013.2255983.

[27] R. Fok, A. An, and X. Wang, "Mining Evolving Data Streams with Particle

Filters," *Computational Intelligence*, vol. 33, no. 2, pp. 147–180, 2017, doi: 10.1111/coin.12071.

[28] S. Bagui and P. Stanley, "Mining frequent itemsets from streaming transaction data using genetic algorithms," *Journal of Big Data*, vol. 7, no. 1, pp. 1–20, 2020, doi: 10.1186/s40537-020-00330-9.

[29] S. A. P. Kumar and B. Xu, "Big Data Analytics Framework for Improved Decision Making," 2014. [Online]. Available: www.honeywell.com

[30] P. P. Angelov and X. Zhou, "Evolving fuzzy-rule-based classifiers from data streams," *IEEE Transactions on Fuzzy Systems*, vol. 16, no. 6, pp. 1462–1475, 2008, doi: 10.1109/TFUZZ.2008.925904.

[31] K. Ali, T. Anwaro, I. H. Naqvi, and M. H. Jafry, "Composite event detection and identification for WSNs using General Hebbian Algorithm," in *IEEE International Conference on Communications*, 2015, pp. 6463–6468. doi: 10.1109/ICC.2015.7249354.

[32] J. Proskurnia, R. Mavlyutov, C. Castillo, K. Aberer, and P. Cudré-Mauroux, "Efficient Document Filtering Using Vector Space Topic Expansion and Pattern-Mining," in *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, 2017, pp. 457–466. doi: 10.1145/3132847.3133016.

[33] C. Castillo, M. Mendoza, and B. Poblete, "Information credibility on Twitter," *Proceedings of the 20th International Conference Companion on World Wide Web, WWW 2011*, pp. 675–684, 2011, doi: 10.1145/1963405.1963500.

[34] H. Tabrizchi and M. Kuchaki Rafsanjani, "A survey on security challenges in cloud computing: issues, threats, and solutions," *Journal of Supercomputing*, vol. 76, no. 12, pp. 9493–9532, 2020, doi: 10.1007/s11227-020-03213-1.

[35] N. Oliveira, I. Praça, E. Maia, and O. Sousa, "Intelligent cyber attack detection and classification for network-based intrusion detection systems," *Applied Sciences (Switzerland)*, vol. 11, no. 4, pp. 1–21, 2021, doi: 10.3390/app11041674.

[36] A. D'Alconzo, A. Coluccia, F. Ricciato, and P. Romirer-Maierhofer, "A distribution-based approach to anomaly detection and application to 3G mobile

traffic," in *GLOBECOM - IEEE Global Telecommunications Conference*, 2009, pp. 1–8. doi: 10.1109/GLOCOM.2009.5425651.

[37] S. E. Smaha, "Haystack: An intrusion detection system," vol. 4 th. pp. 37–44, 1988. doi: 10.1109/acsac.1988.113412.

[38] T. F. Lunt, A. Tamaru, F. Gilham, R. Jagannathan, C. Jalali, and P. G. Neumann, "A REAL-TIME INTRUSION-DETECTION EXPERT SYSTEM (IDES)," 1992.

[39] W. Lee and S. J. Stolfo, "Data Mining Approaches for Intrusion Detection," p. 356, 1998, doi: 10.1111/j.1751-1097.1992.tb02170.x.

[40] F. Carcillo, A. Dal Pozzolo, Y. A. Le Borgne, O. Caelen, Y. Mazzer, and G. Bontempi, "SCARFF: A scalable framework for streaming credit card fraud detection with spark," *Information Fusion*, vol. 41, pp. 182–194, 2018, doi: 10.1016/j.inffus.2017.09.005.

[41] F. Carcillo, Y. A. Le Borgne, O. Caelen, Y. Kessaci, F. Oblé, and G. Bontempi, "Combining unsupervised and supervised learning in credit card fraud detection," *Information Sciences*, vol. 557, pp. 317–331, 2021, doi: 10.1016/j.ins.2019.05.042.

[42] A. A. Khine and H. W. Khin, "Credit Card Fraud Detection Using Online Boosting with Extremely Fast Decision Tree," in *2020 IEEE Conference on Computer Applications, ICCA 2020*, 2020, pp. 1–4. doi: 10.1109/ICCA49400.2020.9022815.

[43] A. M. Rahmani, Z. Babaei, and A. Souri, "Event-driven IoT architecture for data analysis of reliable healthcare application using complex event processing," *Cluster Computing*, vol. 24, no. 2, pp. 1347–1360, 2021, doi: 10.1007/s10586-020-03189-w.

[44] L. R. Nair, S. D. Shetty, and S. D. Shetty, "Applying spark based machine learning model on streaming big data for health status prediction," *Computers and Electrical Engineering*, vol. 65, pp. 393–399, 2018, doi: 10.1016/j.compeleceng.2017.03.009.

[45] A. Kanavos, M. Trigka, E. Dritsas, G. Vonitsanos, and P. Mylonas, "A regularization-based big data framework for winter precipitation forecasting on

streaming data," *Electronics (Switzerland)*, vol. 10, no. 16, pp. 1–21, 2021, doi: 10.3390/electronics10161872.

[46] A. I. Maarala, M. Rautiainen, M. Salmi, S. Pirttikangas, and J. Riekki, "Low latency analytics for streaming traffic data with Apache Spark," in *Proceedings - 2015 IEEE International Conference on Big Data, IEEE Big Data 2015*, 2015, pp. 2855–2858. doi: 10.1109/BigData.2015.7364101.

[47] S. Ameer and M. A. Shah, "Exploiting Big Data Analytics for Smart Urban Planning," in *IEEE Vehicular Technology Conference*, 2018, pp. 1–5. doi: 10.1109/VTCFall.2018.8691036.

[48] B. Amen and A. Grigoris, "A Theoretical Study of Anomaly Detection in Big Data Distributed Static and Stream Analytics," in *Proceedings - 20th International Conference on High Performance Computing and Communications, 16th International Conference on Smart City and 4th International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2018*, 2019, pp. 1177–1182. doi: 10.1109/HPCC/SmartCity/DSS.2018.00198.

[49] D. Jin, S. Shi, Y. Zhang, H. Abbas, and T. T. Goh, "A complex event processing framework for an adaptive language learning system," *Future Generation Computer Systems*, vol. 92, pp. 857–867, 2019, doi: 10.1016/j.future.2017.12.032.

[50] S. A. Shah, D. Z. Seker, S. Hameed, and D. Draheim, "The rising role of big data analytics and IoT in disaster management: Recent advances, taxonomy and prospects," *IEEE Access*, vol. 7, pp. 54595–54614, 2019, doi: 10.1109/ACCESS.2019.2913340.

[51] V. Jain, "Perspective analysis of telecommunication fraud detection using data stream analytics and neural network classification based data mining," *International Journal of Information Technology*, vol. 9, no. 3, pp. 303–310, 2017.

[52] D. Sovilj, P. Budnarain, S. Sanner, G. Salmon, and M. Rao, "A comparative evaluation of unsupervised deep architectures for intrusion detection in sequential data streams," *Expert Systems with Applications*, vol. 159, p. 113577, 2020, doi: 10.1016/j.eswa.2020.113577.

[53] A. Kanavos, T. Panagiotakopoulos, G. Vonitsanos, M. Maragoudakis, and Y. Kiouvrekis, “Forecasting Winter Precipitation based on Weather Sensors Data in Apache Spark,” 2021. doi: 10.1109/IISA52424.2021.9555553.

[54] H. M. Gomes, J. P. Barddal, A. F. Enembreck, and A. Bifet, “A survey on ensemble learning for data stream classification,” *ACM Computing Surveys*, vol. 50, no. 2, pp. 1–36, 2017, doi: 10.1145/3054925.

[55] C. A. Medina, M. R. Perez, and L. C. Trujillo, “IoT paradigm into the smart city vision: a survey,” in *Proceedings - 2017 IEEE International Conference on Internet of Things, IEEE Green Computing and Communications, IEEE Cyber, Physical and Social Computing, IEEE Smart Data, iThings-GreenCom-CPSCom-SmartData 2017*, 2018, pp. 695–704. doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2017.109.

[56] M. A. Khan, R. M. Karim, and Y. Kim, “A Two-Stage Big Data Analytics Framework with RealWorld Applications Using Spark Machine Learningand Long Short-Term Memory Network,” *Symmetry*, vol. 10, no. 10, p. 485, 2018, doi: https://www.mdpi.com/2073-8994/10/10/485.

[57] M. Zaharia, T. Das, H. Li, S. Shenker, and I. Stoica, “Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters,” *4th USENIX Workshop on Hot Topics in Cloud Computing, HotCloud 2012*, 2012.

[58] F. Matsebula and E. Mnkandla, “A big data architecture for learning analytics in higher education,” in *2017 IEEE AFRICON: Science, Technology and Innovation for Africa, AFRICON 2017*, 2017, pp. 951–956. doi: 10.1109/AFRCON.2017.8095610.

[59] F. Matsebula and E. Mnkandla, “Information systems innovation adoption in higher education: Big data and analytics,” in *Proceedings - 2016 3rd International Conference on Advances in Computing, Communication and Engineering, ICACCE 2016*, 2017, pp. 326–329. doi: 10.1109/ICACCE.2016.8073769.

[60] M. Hussain Iqbal and T. Rahim Soomro, “Big Data Analysis: Apache Storm Perspective,” *International Journal of Computer Trends and Technology*, vol. 19, no. 1, pp. 9–14, 2015, doi: 10.14445/22312803/ijctt-v19p103.

[61] E. Friedman and K. Tzoumas, *Introduction to Apache Flink: stream processing for real time and beyond*. O'Reilly Media, Inc., 2016.

[62] Z. Zhang, Z. Yang, Y. Han, and C. Di, "View-driven real-time sensor data processing and servitization system," in *ICEIEC 2019 - Proceedings of 2019 IEEE 9th International Conference on Electronics Information and Emergency Communication*, 2019, pp. 689–694. doi: 10.1109/ICEIEC.2019.8784611.

[63] S. A. Noghabi *et al.*, "Samza: Stateful scalable stream processing at linkedin," *Proceedings of the VLDB Endowment*, vol. 10, no. 12, pp. 1634–1645, 2017, doi: 10.14778/3137765.3137770.

[64] S. Singh, R. Garg, and P. K. Mishra, "Performance optimization of MapReduce-based Apriori algorithm on Hadoop cluster," *Computers and Electrical Engineering*, vol. 67, pp. 348–364, 2018, doi: 10.1016/j.compeleceng.2017.10.008.

[65] B. R. Hiraman, M. C. Viresh, and C. K. Abhijeet, "A Study of Apache Kafka in Big Data Stream Processing," in *2018 International Conference on Information, Communication, Engineering and Technology, ICICET 2018*, 2018, pp. 12–14. doi: 10.1109/ICICET.2018.8533771.

[66] M. Patel and M. Bhise, "Raw data processing framework for IoT," in *2019 11th International Conference on Communication Systems & Networks (COMSNETS)*, 2019, pp. 695–699.

[67] A. M. S. Osman, "A novel big data analytics framework for smart cities," *Future Generation Computer Systems*, vol. 91, pp. 620–633, 2019, doi: 10.1016/j.future.2018.06.046.

[68] P. D. Talagala, R. J. Hyndman, K. Smith-Miles, S. Kandanaarachchi, and M. A. Muñoz, "Anomaly Detection in Streaming Nonstationary Temporal Data," *Journal of Computational and Graphical Statistics*, vol. 29, no. 1, pp. 13–27, 2020, doi: 10.1080/10618600.2019.1617160.

[69] M. Hasan, M. A. Orgun, and R. Schwitter, "Real-time event detection from the Twitter data stream using the TwitterNews+ Framework," *Information Processing and Management*, vol. 56, no. 3, pp. 1146–1165, 2019, doi: 10.1016/j.ipm.2018.03.001.

[70] B. Leang, S. Ean, G. A. Ryu, and K. H. Yoo, “Improvement of kafka streaming using partition and multi-threading in big data environment,” *Sensors (Switzerland)*, vol. 19, no. 1, p. 134, 2019, doi: 10.3390/s19010134.

[71] K. P. Murphy, *Machine learning: a probabilistic perspective*. MIT Press, 2012.

[72] T. Hastie, R. Tibshirani, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer.

[73] C. Manning and H. Schutze, *Foundations of statistical natural language processing*. MIT Press, 1999.

[74] D. Jurafsky, *Speech & language processing*. Pearson Education India, 2000.

[75] F. Hogenboom, F. Frasincar, U. Kaymak, and F. De Jong, “An overview of event extraction from text,” in *CEUR Workshop Proceedings*, 2011, pp. 48–57.

[76] P. Goyal, P. Kaushik, P. Gupta, D. Vashisth, S. Agarwal, and N. Goyal, “Multilevel Event Detection, Storyline Generation, and Summarization for Tweet Streams,” *IEEE Transactions on Computational Social Systems*, vol. 7, no. 1, pp. 8–23, 2020, doi: 10.1109/TCSS.2019.2954116.

[77] O. Alonso, J. Strotgen, R. Baeza-Yates, and M. Gertz, “Temporal Information Retrieval: Challenges and Opportunities,” in *Twaw*, 2011, pp. 1–8. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.204.4720&rep=rep1&type=pdf

[78] T. Sakaki, Makoto Okazaki, and Y. Matsuo, “Earthquake shakes twitter users: real-time event detection by social sensors,” in *Proceedings of the 19th international conference on World wide web*, 2010, pp. 851–860.

[79] D. Zhang, D. Gatica-Perez, S. Bengio, and L. McCowan, “Semi-supervised adapted HMMs for unusual event detection,” in *Proceedings - 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR 2005*, 2005, pp. 611–618. doi: 10.1109/CVPR.2005.316.

[80] A. Ahmed, Q. Ho, J. Eisenstein, E. P. Xing, A. J. Smola, and C. H. Teo, “Unified analysis of streaming news,” in *Proceedings of the 20th International Conference on World Wide Web, WWW 2011*, 2011, pp. 267–276. doi: 10.1145/1963405.1963445.

[81] S. Alam Ansari and A. Zafar, “A fusion of dolphin swarm optimization and improved sine cosine algorithm for automatic detection and classification of objects from surveillance videos,” *Measurement: Journal of the International Measurement Confederation*, vol. 192, no. October 2021, p. 110921, 2022, doi: 10.1016/j.measurement.2022.110921.

[82] M. M. Masud *et al.*, “Addressing concept-evolution in concept-drifting data streams,” in *Proceedings - IEEE International Conference on Data Mining, ICDM*, 2010, pp. 929–934. doi: 10.1109/ICDM.2010.160.

[83] M. Alarfaj *et al.*, “Automatic Anomaly Monitoring in Public Surveillance Areas,” *Intelligent Automation & Soft Computing*, vol. 35, no. 3, pp. 2655–2671, 2023, doi: 10.32604/iasc.2023.027205.

[84] T. Sakaki, M. Okazaki, and Y. Matsuo, “Earthquake shakes Twitter users: Real-time event detection by social sensors,” in *Proceedings of the 19th International Conference on World Wide Web, WWW ’10*, 2010, pp. 851–860. doi: 10.1145/1772690.1772777.

[85] A. Kumar, R. M. Hegde, R. Singh, and B. Raj, “Event detection in short duration audio using Gaussian Mixture Model and Random Forest Classifier,” in *European Signal Processing Conference*, 2013, pp. 1–5.

[86] G. Spanos, K. M. Giannoutakis, K. Votis, and D. Tzovaras, “Combining statistical and machine learning techniques in IoT anomaly detection for smart homes,” in *IEEE International Workshop on Computer Aided Modeling and Design of Communication Links and Networks, CAMAD*, 2019, pp. 5–10. doi: 10.1109/CAMAD.2019.8858490.

[87] R. M. Shukla and S. Sengupta, “Scalable and Robust Outlier Detector using Hierarchical Clustering and Long Short-Term Memory (LSTM) Neural Network for the Internet of Things,” *Internet of Things (Netherlands)*, vol. 9, p. 100167, 2020, doi: 10.1016/j.iot.2020.100167.

[88] E. Alomari, R. Mehmood, and I. Katib, “Road traffic event detection using twitter data, machine learning, and apache spark,” in *Proceedings - 2019 IEEE SmartWorld, Ubiquitous Intelligence and Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Internet of People and*

Smart City Innovation, SmartWorld/UIC/ATC/SCALCOM/IOP/SCI 2019, 2019, pp. 1888–1895. doi: 10.1109/SmartWorld-UIC-ATC-SCALCOM-IOP-SCI.2019.00332.

[89] M. Srivastava, R. Garg, and P. K. Mishra, "Analysis of data extraction and Data cleaning in web usage mining," *ACM International Conference Proceeding Series*, vol. 06-07-Marc, 2015, doi: 10.1145/2743065.2743078.

[90] N. Srivastava *et al.*, "Applications of fungal cellulases in biofuel production: Advances and limitations," *Renewable and Sustainable Energy Reviews*, vol. 82, no. August, pp. 2379–2386, 2018, doi: 10.1016/j.rser.2017.08.074.

[91] M. Walther and M. Kaisser, "Geo-spatial event detection in the twitter stream," in *European conference on information retrieval*, 2013, pp. 356–367.

[92] H. Tian, L. Wang, H. Shen, and A. W.-C. Liew, "Improved Ensemble Classification for Evolving Data Streams," *IEEE Intelligent Systems*, 2020.

[93] R. S. M. de Barros and S. G. T. d. C. Santos, "An overview and comprehensive comparison of ensembles for concept drift," *Information Fusion*, vol. 52, pp. 213–244, 2019, doi: 10.1016/j.inffus.2019.03.006.

[94] H. M. Gomes *et al.*, "Adaptive random forests for evolving data stream classification," *Machine Learning*, vol. 106, no. 9, pp. 1469–1495, 2017, doi: 10.1007/s10994-017-5642-8.

[95] S. Ren, B. Liao, W. Zhu, and K. Li, "Knowledge-maximized ensemble algorithm for different types of concept drift," *Information Sciences*, vol. 430, pp. 261–281, 2018, doi: 10.1016/j.ins.2017.11.046.

[96] P. Zhao, L. W. Cai, and Z. H. Zhou, "Handling concept drift via model reuse," *Machine Learning*, vol. 109, no. 3, pp. 533–568, 2020, doi: 10.1007/s10994-019-05835-w.

[97] M. Gupta, K. Rajnish, and V. Bhattacharjee, "Impact of Parameter Tuning for Optimizing Deep Neural Network Models for Predicting Software Faults," *Scientific Programming*, vol. 2021, 2021, doi: 10.1155/2021/6662932.

[98] P. S. Bishnu and V. Bhattacherjee, "Software fault prediction using quad tree-based K-means clustering algorithm," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 6, pp. 1146–1150, 2012, doi:

10.1109/TKDE.2011.163.

[99] Z. Li, W. Huang, Y. Xiong, S. Ren, and T. Zhu, “Incremental learning imbalanced data streams with concept drift: The dynamic updated ensemble algorithm,” *Knowledge-Based Systems*, vol. 195, p. 105694, 2020, doi: 10.1016/j.knosys.2020.105694.

[100] Y. Lu, Y. M. Cheung, and Y. Yan Tang, “Adaptive chunk-based dynamic weighted majority for imbalanced data streams with concept drift,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 31, no. 8, pp. 2764–2778, 2020, doi: 10.1109/TNNLS.2019.2951814.

[101] W. N. Street and Y. Kim, “A streaming ensemble algorithm (SEA) for large-scale classification,” in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’01*, 2001, pp. 377–382. doi: 10.1145/502512.502568.

[102] H. Wang, W. Fan, P. S. Yu, and J. Han, “Mining concept-drifting data streams using ensemble classifiers,” in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’03*, 2003, pp. 226–235. doi: 10.1145/956750.956778.

[103] D. Brzezinski and J. Stefanowski, “Reacting to different types of concept drift: The accuracy updated ensemble algorithm,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 81–94, 2014, doi: 10.1109/TNNLS.2013.2251352.

[104] N. C. Oza, “Online Bagging and Boosting,” in *2005 IEEE International Conference on Systems, Man and Cybernetics*, 2005, pp. 2340–2345. doi: 10.1109/ICSMC.2005.1571498.

[105] R. Y. T. Hou and Y. W. Leung, “Hybrid parent selection for peer-to-peer multimedia streaming,” in *Proceedings - 2012 8th International Conference on Computing Technology and Information Management, ICCM 2012*, 2012, vol. 2, no. i, pp. 683–687.

[106] J. S. Zhang and Y. W. Leung, “Improved possibilistic C-means clustering algorithms,” *IEEE Transactions on Fuzzy Systems*, vol. 12, no. 2, pp. 209–217, 2004, doi: 10.1109/TFUZZ.2004.825079.

[107] R. Pelossof, M. Jones, I. Vovsha, and C. Rudin, “Online coordinate boosting,” in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops 2009*, 2009, pp. 1354–1361. doi: 10.1109/ICCVW.2009.5457454.

[108] A. J. Wyner, M. Olson, J. Bleich, and D. Mease, “Explaining the success of adaboost and random forests as interpolating classifiers,” *Journal of Machine Learning Research*, vol. 18, no. 1, pp. 1558–1590, 2017.

[109] R. S. M. De Barros, S. G. T. De Carvalho Santos, and P. M. G. Junior, “A Boosting-like Online Learning Ensemble,” in *Proceedings of the International Joint Conference on Neural Networks*, 2016, pp. 1871–1878. doi: 10.1109/IJCNN.2016.7727427.

[110] R. Sebastião, J. Gama, and T. Mendonça, “Fading histograms in detecting distribution and concept changes,” *International Journal of Data Science and Analytics*, vol. 3, no. 3, pp. 183–212, 2017, doi: 10.1007/s41060-017-0043-4.

[111] J. N. Van Rijn, G. Holmes, B. Pfahringer, and J. Vanschoren, “Having a blast: Meta-learning and heterogeneous ensembles for data streams,” *Proceedings - IEEE International Conference on Data Mining, ICDM*, vol. 2016-Janua, pp. 1003–1008, 2016, doi: 10.1109/ICDM.2015.55.

[112] A. Apicella, F. Donnarumma, F. Isgrò, and R. Prevete, “A survey on modern trainable activation functions,” *Neural Networks*, vol. 138, pp. 14–32, 2021, doi: 10.1016/j.neunet.2021.01.026.

[113] C. Manapragada, G. I. Webb, and M. Salehi, “Extremely fast decision tree,” in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2018, pp. 1953–1962. doi: 10.1145/3219819.3220005.

[114] G. Hulten, L. Spencer, and P. Domingos, “Mining time-changing data streams,” in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining - KDD ’01*, 2001, vol. 18, pp. 97–106. doi: 10.1145/502512.502529.

[115] M. Han, X. Zhang, Z. Chen, H. Wu, and M. Li, “Dynamic ensemble selection classification algorithm based on window over imbalanced drift data stream,”

Knowledge and Information Systems, pp. 1–24, 2022, doi: 10.1007/s10115-022-01791-5.

[116] A. Bifet *et al.*, "Extremely fast decision tree mining for evolving data streams," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 1733–1742. doi: 10.1145/3097983.3098139.

[117] S. Yoon, N. Elhadad, and S. Bakken, "A practical approach for content mining of tweets," *American Journal of Preventive Medicine*, vol. 45, no. 1, pp. 122–129, 2013, doi: 10.1016/j.amepre.2013.02.025.

[118] M. O Connor, K. Conboy, and D. Dennehy, "COVID-19 affected remote workers: a temporal analysis of information system development during the pandemic," *Journal of Decision Systems*, vol. 31, no. 3, pp. 207–233, 2022, doi: 10.1080/12460125.2020.1861772.

[119] M. Asgari-Chenaghlu, M. R. Feizi-Derakhshi, L. Farzinvash, M. A. Balafar, and C. Motamed, "Topic Detection and Tracking Techniques on Twitter: A Systematic Review," *Complexity*, 2021, doi: 10.1155/2021/8833084.

[120] F. Yang, X. Yu, Y. Liu, and M. Yang, "Automatic detection of rumor on Sina Weibo," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2012, pp. 1–7. doi: 10.1145/2350190.2350203.

[121] M. S. Al-Rakhami and A. M. Al-Amri, "Lies Kill, Facts Save: Detecting COVID-19 Misinformation in Twitter," *IEEE Access*, vol. 8, pp. 155961–155970, 2020, doi: 10.1109/ACCESS.2020.3019600.

[122] R. Dayani, N. Chhabra, T. Kadian, and R. Kaushal, "Rumor detection in twitter: An analysis in retrospect," in *International Symposium on Advanced Networks and Telecommunication Systems, ANTS*, 2016, vol. 2016-Febru, pp. 6–8. doi: 10.1109/ANTS.2015.7413660.

[123] V. Qazvinian, E. Rosengren, D. R. Radev, and Q. Mei, "Rumor has it: Identifying misinformation in microblogs," in *EMNLP 2011 - Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 2011, pp. 1589–1599.

[124] C. Castillo, M. Mendoza, and B. Poblete, “Information credibility on Twitter,” in *Proceedings of the 20th International Conference Companion on World Wide Web, WWW 2011*, 2011, pp. 675–684. doi: 10.1145/1963405.1963500.

[125] J. C. S. Reis, A. Correia, F. Murai, A. Veloso, F. Benevenuto, and E. Cambria, “Supervised Learning for Fake News Detection,” *IEEE Intelligent Systems*, vol. 34, no. 2, pp. 76–81, 2019, doi: 10.1109/MIS.2019.2899143.

[126] S. Hamidian and M. T. Diab, “Rumor Detection and Classification for Twitter Data,” 2019, [Online]. Available: http://arxiv.org/abs/1912.08926

[127] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, “The WEKA data mining software: An Update,” *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, 2009, doi: 10.1145/1656274.1656278.

[128] A. Gumaei, M. S. Al-Rakhami, M. M. Hassan, V. H. C. De Albuquerque, and D. Camacho, “An Effective Approach for Rumor Detection of Arabic Tweets Using eXtreme Gradient Boosting Method,” *ACM Transactions on Asian and Low-Resource Language Information Processing*, vol. 21, no. 1, pp. 1–16, 2022, doi: 10.1145/3461697.

[129] M. H. ur Rehman, C. S. Liew, A. Abbas, P. P. Jayaraman, T. Y. Wah, and S. U. Khan, “Big Data Reduction Methods: A Survey,” *Data Science and Engineering*, vol. 1, no. 4, pp. 265–284, 2016, doi: 10.1007/s41019-016-0022-0.

[130] Y. Duan, L. Jiang, X. Lin, and X. Chen, “An Improved Content Based Image Retrieval System On Apache Spark,” in *Proceedings of the 2016 International Conference on Mechatronics, Control and Automation Engineering*, 2016, pp. 107–110. doi: 10.2991/mcae-16.2016.26.

[131] J. Roesslein, “Tweepy: Twitter for Python!,” 2020. url: https://github.com/tweepy/tweepy

[132] K. Shu, D. Mahudeswaran, S. Wang, and H. Liu, “Hierarchical propagation networks for fake news detection: Investigation and exploitation,” in *Proceedings of the 14th International AAAI Conference on Web and Social Media, ICWSM 2020*, 2020, pp. 626–637. doi: 10.1609/icwsm.v14i1.7329.

[133] Y. Liu and Y. F. B. Wu, “FNED: A Deep Network for Fake News Early

Detection on Social Media," *ACM Transactions on Information Systems*, vol. 38, no. 3, pp. 1–33, 2020, doi: 10.1145/3386253.

[134] R. Deveaud and F. Boudin, "Effective tweet contextualization with hashtags performance prediction and multi-document summarization," in *CEUR Workshop Proceedings*, 2013, pp. 1245–1255.

[135] E. Hutto, C.J. and Gilbert, "VADER: A Parsimonious Rule-based Model for sentiment analysis of social media text," in *Eighth International AAAI Conference on Weblogs and Social Media*, 2014, pp. 216–225. [Online]. Available: https://www.aaai.org/ocs/index.php/ICWSM/ICWSM14/paper/viewPaper/8109

[136] "MEDISYS." https://medisys.newsbrief.eu/medisys/homeedition/en/home.html

[137] E. Van Der Goot, H. Tanev, and J. P. Linge, "Combining twitter and media reports on public health events in medisys," *WWW 2013 Companion - Proceedings of the 22nd International Conference on World Wide Web*, pp. 703–705, 2013, doi: 10.1145/2487788.2488028.

[138] S. Petrović, M. Osborne, and V. Lavrenko, "Streaming first story detection with application to twitter," in *NAACL HLT 2010 - Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, Proceedings of the Main Conference*, 2010, pp. 181–189.

[139] C. C. Aggarwal and K. Subbian, "Event detection in social streams," in *Proceedings of the 12th SIAM International Conference on Data Mining, SDM 2012*, 2012, pp. 624–635. doi: 10.1137/1.9781611972825.54.

[140] A. J. McMinn and J. M. Jose, "Real-time entity-based event detection for twitter," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2015, vol. 9283, pp. 65–77. doi: 10.1007/978-3-319-24027-5_6.

[141] Ø. K. Repp and H. Ramampiaro, "Event Detection in Social Media - Detecting News Events from the Twitter Stream in Real-Time," 2016. [Online]. Available: http://hdl.handle.net/11250/2410729%5Cnhttp://hdl.handle.net/11250/2410729

[142] J. Pang, X. Pu, and C. Li, “A Hybrid Algorithm Incorporating Vector Quantization and One-Class Support Vector Machine for Industrial Anomaly Detection,” *IEEE Transactions on Industrial Informatics*, vol. 18, no. 12, pp. 8786–8796, 2022, doi: 10.1109/TII.2022.3145834.

[143] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, “Estimating the support of a high-dimensional distribution,” *Neural Computation*, vol. 13, no. 7, pp. 1443–1471, 2001, doi: 10.1162/089976601750264965.

[144] P. Machaka, A. Bagula, and F. Nelwamondo, “Using exponentially weighted moving average algorithm to defend against DDoS attacks,” in *2016 Pattern Recognition Association of South Africa and Robotics and Mechatronics International Conference, PRASA-RobMech 2016*, 2017, pp. 1–6. doi: 10.1109/RoboMech.2016.7813157.

[145] M. Pattnaik, “Abnormal event detection in pedestrian pathway using GARCH model and MLP classifier,” *INTERNATIONAL JOURNAL OF ADVANCES IN SIGNAL AND IMAGE SCIENCES*, vol. 5, no. 2, pp. 15–22, 2019.

[146] V. Guralnik and J. Srivastava, “Event detection from time series data,” in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, 1999, pp. 33–42. doi: 10.1145/312129.312190.

[147] V. Ishimtsev, I. Nazarov, A. Bernstein, and E. Burnaev, “Conformal k-NN Anomaly Detector for Univariate Data Streams,” in *Proceedings of Machine Learning Research*, 2017, pp. 1–15. [Online]. Available: http://arxiv.org/abs/1706.03412

[148] G. Manthei and M. Guckert, “Classification of Located Acoustic Emission Events Using Neural Network,” *Journal of Nondestructive Evaluation*, vol. 42, no. 1, pp. 1–12, 2023, doi: 10.1007/s10921-022-00913-x.

[149] C. Bahhar *et al.*, “Wildfire and Smoke Detection Using Staged YOLO Model and Ensemble CNN,” *Electronics (Switzerland)*, vol. 12, no. 1, p. 228, 2023, doi: 10.3390/electronics12010228.

[150] P. V. N. Rajeswari, M. Shashi, T. K. Rao, M. Rajya Lakshmi, and L. V. Kiran, “Effective intrusion detection system using concept drifting data stream and

support vector machine," *Concurrency and Computation: Practice and Experience*, vol. 34, no. 21, pp. 1–17, 2022, doi: 10.1002/cpe.7118.

[151] F. Ucar, O. F. Alcin, B. Dandil, and F. Ata, "Power quality event detection using a fast extreme learning machine," *Energies*, vol. 11, no. 1, pp. 1–14, 2018, doi: 10.3390/en11010145.

[152] C. L. Fernando, T. Yoshii, and T. Tsubota, "Combining the Deep Neural Network with the K-Means for Traffic Accident," *World Academy of Science, Engineering and Technology International Journal of Transport and Vehicle Engineering*, vol. 16, no. 07, p. 2022, 2022.

[153] R. J. Frank, N. Davey, and S. P. Hunt, "Time series prediction and neural networks," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 31, no. 1–3, pp. 91–103, 2001, doi: 10.1023/A:1012074215150.

[154] A. R. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic modeling using deep belief networks," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 1, pp. 14–22, 2012, doi: 10.1109/TASL.2011.2109382.

[155] L. R. Rabiner and B. H. Juang, "An Introduction to Hidden Markov Models," *IEEE ASSP Magazine*, vol. 3, no. 1, IEEE, pp. 4–16, 1986. doi: 10.1109/MASSP.1986.1165342.

[156] H. Saleh, E. M. G. Younis, R. Sahal, and A. A. Ali, "Predicting Systolic Blood Pressure in Real-Time Using Streaming Data and Deep Learning," *Mobile Networks and Applications*, 2020, doi: 10.1007/s11036-020-01645-w.

[157] R. Pascanu, M. Tomas, and oshua Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, 2013, pp. 1310–1318. doi: 10.1007/978-3-319-93145-6_3.

[158] H. Zhang, L. Zhang, and Y. Jiang, "Overfitting and Underfitting Analysis for Deep Learning Based End-to-end Communication Systems," in *2019 11th International Conference on Wireless Communications and Signal Processing, WCSP 2019*, 2019, pp. 1–6. doi: 10.1109/WCSP.2019.8927876.

[159] S. Hochreiter and Jü. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997, doi:

10.17582/journal.pjz/2018.50.6.2199.2207.

[160] F. A. Gers, Jü. Schmidhuber, and F. Cummins, “Learning to forget: Continual prediction with LSTM,” *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000, doi: 10.5860/choice.27-5238.

[161] K. Greff, R. K. Srivastava, J. Koutnik, B. R. Steunebrink, and J. Schmidhuber, “LSTM: A Search Space Odyssey,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2017, doi: 10.1109/TNNLS.2016.2582924.

[162] S. Dawar, V. Sharma, and V. Goyal, “Mining top-k high-utility itemsets from a data stream under sliding window model,” *Applied Intelligence*, vol. 47, no. 4, pp. 1240–1255, 2017, doi: 10.1007/s10489-017-0939-7.

[163] S. Dawar, V. Goyal, and D. Bera, “A hybrid framework for mining high-utility itemsets in a sparse transaction database,” *Applied Intelligence*, vol. 47, no. 3, pp. 809–827, 2017, doi: 10.1007/s10489-017-0932-1.

[164] P. Rawat, K. D. Singh, H. Chaouchi, and J. M. Bonnin, “Wireless sensor networks: A survey on recent developments and potential synergies,” *Journal of Supercomputing*, vol. 68, no. 1, pp. 1–48, 2014, doi: 10.1007/s11227-013-1021-9.

[165] G. Giambene, V. A. Le, T. Bourgeau, and H. Chaouchi, “Iterative Multi-Level Soft Frequency Reuse with Load Balancing for Heterogeneous LTE-A Systems,” *IEEE Transactions on Wireless Communications*, vol. 16, no. 2, pp. 924–938, 2017, doi: 10.1109/TWC.2016.2634536.

[166] D. T. Shipmon, J. M. Gurevitch, P. M. Piselli, and S. T. Edwards, “Detection of anomalous drops with limited features and sparse examples in noisy highly periodic data,” 2017, [Online]. Available: http://arxiv.org/abs/1708.03665

[167] M. Chong, A. Abraham, and M. Paprzycki, “Traffic accident analysis using machine learning paradigms,” *Informatica (Ljubljana)*, vol. 29, no. 1, pp. 89–98, 2005.

[168] C. Savaglio, M. Ganzha, M. Paprzycki, C. Bădică, M. Ivanović, and G. Fortino, “Agent-based Internet of Things: State-of-the-art and research challenges,” *Future Generation Computer Systems*, vol. 102, pp. 1038–1053, 2020, doi:

10.1016/j.future.2019.09.016.

[169] X. Liu and P. S. Nielsen, “Scalable prediction-based online anomaly detection for smart meter data,” *Information Systems*, vol. 77, pp. 34–47, 2018, doi: 10.1016/j.is.2018.05.007.

[170] M. A. Khan, M. R. Karim, and Y. Kim, “A scalable and hybrid intrusion detection system based on the convolutional-LSTM network,” *Symmetry*, vol. 11, no. 4, 2019, doi: 10.3390/sym11040583.

[171] X. Li, S. Wu, and L. Wang, “Blood pressure prediction via recurrent models with contextual layer,” *26th International World Wide Web Conference, WWW 2017*, pp. 685–693, 2017, doi: 10.1145/3038912.3052604.

[172] U. M. Girkar, R. Uchimido, L. wei H. Lehman, P. Szolovits, L. Celi, and W. H. Weng, “Predicting Blood Pressure Response to Fluid Bolus Therapy Using Attention-Based Neural Networks for Clinical Interpretability,” *arXiv*, pp. 1–6, 2018.

[173] B. Veeravalli, C. J. Deepu, and D. Ngo, “Real-Time, Personalized Anomaly Detection in Streaming Data for Wearable Healthcare Devices,” no. October, pp. 403–426, 2017, doi: 10.1007/978-3-319-58280-1_15.

[174] P. Su, X. R. Ding, Y. T. Zhang, J. Liu, F. Miao, and N. Zhao, “Long-term blood pressure prediction with deep recurrent neural networks,” *arXiv*, no. March, pp. 4–7, 2017.

[175] Z. Chen, M. Han, H. Wu, M. Li, and X. Zhang, “A multi-level weighted concept drift detection method,” *Journal of Supercomputing*, vol. 79, no. 5, pp. 5154–5180, 2023, doi: 10.1007/s11227-022-04864-y.

[176] G. Oliveira, L. L. Minku, and A. L. I. Oliveira, “Tackling Virtual and Real Concept Drifts: An Adaptive Gaussian Mixture Model Approach,” *IEEE Transactions on Knowledge and Data Engineering*, 2021, doi: 10.1109/TKDE.2021.3099690.

[177] H. Ghomeshi, M. M. Gaber, and Y. Kovalchuk, “EACD: evolutionary adaptation to concept drifts in data streams,” *Data Mining and Knowledge Discovery*, vol. 33, no. 3, pp. 663–694, 2019, doi: 10.1007/s10618-019-00614-6.

[178] L. L. Minku, A. P. White, and X. Yao, “The impact of diversity on online ensemble learning in the presence of concept drift,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 5, pp. 730–742, 2010, doi: 10.1109/TKDE.2009.156.

[179] C. Te Li, M. K. Shan, S. H. Jheng, and K. C. Chou, “Exploiting concept drift to predict popularity of social multimedia in microblogs,” *Information Sciences*, vol. 339. pp. 310–331, 2016. doi: 10.1016/j.ins.2016.01.009.

[180] S. Kharya *et al.*, “Weighted Bayesian Belief Network: A Computational Intelligence Approach for Predictive Modeling in Clinical Datasets,” *Computational Intelligence and Neuroscience*, vol. 2022, 2022, doi: 10.1155/2022/3813705.

[181] D. Brzezinski, “Mining Data Streams with Concept Drift,” 2010. [Online]. Available: http://www.cs.put.poznan.pl/dbrzezinski/publications/ConceptDrift.pdf

[182] A. Bifet, G. Holmes, B. Pfahringer, and R. Gavaldà, “Improving adaptive bagging methods for evolving data streams,” in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2009, pp. 23–37. doi: 10.1007/978-3-642-05224-8_4.

[183] M. K. Olorunnimbe, H. L. Viktor, and E. Paquet, “Dynamic adaptation of online ensembles for drifting data streams,” *Journal of Intelligent Information Systems*, vol. 50, no. 2, pp. 291–313, 2018, doi: 10.1007/s10844-017-0460-9.

[184] K. Namitha and S. G. Kumar, “Concept drift detection in data stream clustering and its application on weather data,” *International Journal of Agricultural and Environmental Information Systems*, vol. 11, no. 1, pp. 67–85, 2020, doi: 10.4018/IJAEIS.2020010104.

[185] M. Baena-García, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavaldà, and R. Morales-Bueno, “Early Drift Detection Method,” in *4th ECML PKDD International Workshop on Knowledge Discovery from Data Streams*, 2006, vol. 6, pp. 77–86. [Online]. Available: http://www.lsi.upc.edu/~abifet/EDDM.pdf

[186] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, “Learning with Drift

Detection," in *Brazilian Symposium on Artificial Intelligence*, 2004, pp. 286–295. doi: 10.1007/978-3-540-28645-5_29.

[187] A. Bifet, "Adaptive learning and mining for data streams and frequent patterns," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 55–56, 2009, doi: 10.1145/1656274.1656287.

[188] R. F. Mansour, S. Al-Otaibi, A. Al-Rasheed, H. Aljuaid, I. V. Pustokhina, and D. A. Pustokhin, "An optimal big data analytics with concept drift detection on high-dimensional streaming data," *Computers, Materials and Continua*, vol. 68, no. 3, pp. 2843–2858, 2021, doi: 10.32604/cmc.2021.016626.

[189] Y. Sun, K. Tang, Z. Zhu, and X. Yao, "Concept Drift Adaptation by Exploiting Historical Knowledge," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4822–4832, 2018, doi: 10.1109/TNNLS.2017.2775225.

[190] S. Dalhoumi, G. Dray, J. Montmain, G. Derosiere, and S. Perrey, "An adaptive accuracy-weighted ensemble for inter-subjects classification in brain-computer interfacing," in *2015 7th International IEEE/EMBS Conference on Neural Engineering (NER)*, Apr. 2015, pp. 126–129. doi: 10.1109/NER.2015.7146576.

[191] A. Cano and B. Krawczyk, "Kappa Updated Ensemble for drifting data stream mining," *Machine Learning*, vol. 109, no. 1, pp. 175–218, 2020, doi: 10.1007/s10994-019-05840-z.

[192] H. Du, "Transfer Learning for Data Stream Mining in Non-Stationary Environments," 2021.

[193] C. Yang, Y. M. Cheung, J. Ding, and K. C. Tan, "Concept Drift-Tolerant Transfer Learning in Dynamic Environments," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 8, pp. 3857–3871, 2022, doi: 10.1109/TNNLS.2021.3054665.

LIST OF RESEARCH PUBLICATIONS

SCI/SCIE

1. S. Arora, R. Rani and N. Saxena (2021) “An efficient approach for detecting anomalous events in real-time weather datasets”, Concurrency and Computation: Practice and Experience (Wiley), 34(5), e6707. [SCIE Indexed, Impact Factor: 2.0] *(Published)*
2. S. Arora, R. Rani and N. Saxena (2022) “Modified valence aware dictionary for sentiment reasoning classifier for detection and classification of Covid-19 related rumors from social media data streams”, Concurrency and Computation: Practice and Experience (Wiley), 34(21), e7124. [SCIE Indexed, Impact Factor: 2.0] *(Published)*
3. S. Arora, R. Rani and N. Saxena (2023) “SETL: A Transfer Learning based Dynamic Ensemble Classifier for Concept Drift Detection in Streaming Data”, Cluster Computing (Springer), 1-16. [SCIE Indexed, Impact Factor: 4.4] *(Published)*
4. S. Arora, R. Rani and N. Saxena (2022) “A Systematic Review on Detection and Adaptation of Concept Drift in Streaming Data using Machine Learning Techniques”, WIREs (Wiley). [SCIE Indexed, Impact Factor: 8.1] *(Under Review)*

Conferences

1. S. Arora and R. Rani. "A Novel Framework for Distributed Stream Processing and Analysis of Twitter Data." *In proceedings of International Conference on Innovative Computing and Communications: Proceedings of ICICC 2020*, pp. 147-161. Springer.
2. S. Arora, R. Rani and N. Saxena. "Music Stream Analysis for the Prediction of Song Popularity using Machine Learning and Deep Learning Approach." *In proceedings of 2022 3rd International Conference on Computing, Analytics and Networks (ICAN)*, pp. 1-5. IEEE.