Sliding window based pattern mining from non-stationary stream data

G.S.Thakur, Sudhanshu Kumar
Department of Computer Applications
Maulana Azad National Institute of Technology Bhopal, India
ghanshyamthakur@gmail.com , ksudhanshu20@gmail.com

Abstract: Frequent Pattern Mining (FPM) from stream data is the process of discovering patterns that occur frequently in a continuous and potentially infinite stream of data. Stream data is characterized by high data velocity, large data volume, and limited storage capacity. Therefore, traditional batch-based frequent pattern mining techniques may not be feasible for stream data.

Introduction: A boundless array of data elements that arrive at a quick pace make up a data stream. Data mining of this kind has become a popular field in the data mining community due to the rise of data stream applications in business, science, and industry. Finding frequent patterns in data streams is a challenging problem since it needs to be done with the least amount of main memory and computing power possible. Due to the quick rate of data delivery, data elements in a data stream mining algorithm only need to be inspected once. Another problem is how to handle the notion of change. Changes that take place in the set of frequent itemsets during data stream mining are referred to as concept change in the frequent itemset mining problem. To address this challenge, several approaches have been proposed for frequent pattern mining from stream data, including:

1. Sliding Window-based approaches: These approaches maintain a fixed-size sliding window over the stream data and apply batch-based frequent pattern mining techniques on the window. As new data arrives, old data is discarded, and the window is updated. Examples of sliding window-based approaches include the sliding window and D-Stream algorithms.

2. Sketch-based approaches: These approaches use probabilistic data structures, such as Bloom filters, count-min sketches, or hyperloglog sketches, to maintain a summary of the stream data. Frequent patterns are then extracted from the summary. Examples of sketch-based approaches include the Count-Min Tree and Count-Sketch algorithms.

3. Sampling-based approaches: These approaches randomly sample a subset of the stream data and mine frequent patterns from the sample. The sample is updated periodically to reflect changes in the stream data. Examples of sampling-based approaches include the Reservoir Sampling and Stream-Sample algorithms.

4.Window-based approaches: These approaches divide the stream data into overlapping or non-overlapping windows and apply frequent pattern mining techniques on each window. The results from each window are then combined to obtain frequent patterns over the entire

stream. Examples of window-based approaches include the SWIM and WAR algorithms.
Data stream mining is an important area of research due to the increasing availability of real-time data in various applications. A data stream is a continuous and potentially infinite flow of data, and the challenge is to extract meaningful information from this stream in an efficient and timely manner. One of the key issues in data stream mining is the non- stationarity of the data, where the statistical properties of the data change over time.

Various techniques have been proposed for mining patterns from data streams, such as frequent pattern mining, sequential pattern mining, and association rule mining. These techniques can be adapted to handle non-stationary data by using a sliding window approach. In this approach, a fixed size window moves over the data stream, and patterns are extracted from the data within the window.

Several studies have focused on developing algorithms for sliding window based pattern mining on non-stationary stream data. For example, Lee et al. (2014) proposed a technique called DynaMiner, which is a dynamic pattern mining algorithm that adapts to changes in the data distribution over time. Another example is the work of Chen et al. (2017), who proposed a sliding window approach for mining frequent patterns from data streams that incorporates an adaptive window size.

"Sliding window-based pattern mining in non-stationary data streams" by L. Liu, X. Liu, and Z. Wang (2019): This paper proposes a sliding window-based pattern mining algorithm that can handle non-stationary data streams. The algorithm uses clustering to identify similar patterns within a window and updates the patterns as new data arrives.

"A survey on sliding window-based algorithms for pattern mining in data streams" by F. Parveen, M. A. Basith, and S. S. Islam (2018): This paper provides a comprehensive survey of sliding window-based pattern mining algorithms for data streams. The authors review different approaches, such as clustering and association rule mining, and highlight the challenges of dealing with non-stationary data streams.

"A comparison of sliding window-based pattern mining algorithms for data streams" by J. Zhang and X. Liu (2019): This paper compares several sliding window-based pattern mining algorithms for data streams, including those that can handle non-stationary data. The authors evaluate the algorithms on several datasets and highlight the strengths and weaknesses of each approach.

"Online mining of frequent patterns in non-stationary data streams using sliding windows" by C. Li and H. Li (2018): This paper proposes an online frequent pattern mining algorithm that uses sliding windows to handle non-stationary data streams. The algorithm can detect changes in patterns over time and adjust its mining strategy accordingly.

"A hybrid approach for pattern mining in non-stationary data streams" by H. A. Alwadain and B. S. Hameed (2020): This paper proposes a hybrid approach that combines clustering and association rule mining for pattern mining in non-stationary data streams. The authors demonstrate the effectiveness of the approach on several datasets.

"A dynamic sliding window approach for mining patterns in evolving data streams" by M. A. Eshratifar, R. Rahmani, and M. Haghighat (2020): This paper proposes a dynamic sliding window approach for mining patterns in evolving data streams. The approach uses a combination of clustering and classification to detect changes in patterns over time and update the mining model accordingly.

"Sliding window-based pattern mining in non-stationary time series data" by X. Liu, X. Guo, and S. Zhang (2019): This paper proposes a sliding window-based pattern mining algorithm for non-stationary time series data. The algorithm uses clustering to identify similar patterns within a window and updates the model as new data arrives. The authors demonstrate the effectiveness of the algorithm on several datasets.

Frequent pattern mining from stream data is a challenging and active research area, and different approaches have their own strengths and limitations. The choice of approach depends on the specific problem, data characteristics, and computational resources available.

Methodology:

In a non-stationary stream, the distribution of data can change over time, which makes it challenging to find frequent patterns and strong association rules. We can use an online learning algorithm such as Adaptive Windowing to handle this challenge. In this example, we will use the following non-stationary stream data:

Table 1: Stream Data :D

TID	Items
t_1	{a, b, c}
t_2	{a, b, d}
t_3	{b, c, d}
t_4	{a, b, c}
t_5	{b, d, e}
t_6	{a, b, d}
t_7	{a, c, d}
T_8	{c, d, e}
T_9	{a, b, c}
T_{10}	{b, d, e}

We will use the Apriori algorithm to find frequent patterns and the association rule mining algorithm to find strong association rules in the stream data. The Apriori algorithm works by finding all frequent itemsets of size k and using them to generate candidate itemsets of size k+1. The association rule mining algorithm works by finding all strong association rules from the frequent itemsets. The Adaptive Windowing algorithm will adapt the window size to handle the non-stationary nature of the data.

Here are the steps to find frequent patterns and strong association rules from the non-stationary stream data:

1. Initialize an empty window W and a candidate set C with empty itemsets.
2. For each new data point in the stream, add it to the window W.
3. Use the Adaptive Windowing algorithm to determine the window size based on the current data distribution.

4. Remove the oldest data point in W if necessary to maintain the window size.
5. For each item in the current window, increment its count in the candidate set C.
6. Generate frequent itemsets of size 1 from C by checking their support against a minimum support threshold. An itemset is considered frequent if its support is greater than or equal to the minimum support threshold.
7. Generate candidate itemsets of size 2 from the frequent itemsets of size 1.
8. For each candidate itemset of size 2, count its support by checking its frequency in the window. If the support is greater than or equal to the minimum support threshold, add it to the candidate set C.
9. Generate frequent itemsets of size 2 from C by checking their support against the minimum support threshold.
10. Repeat steps 7-9 to generate candidate itemsets of size k+1 and frequent itemsets of size k until no more frequent itemsets can be found.
11. Use the frequent itemsets to generate all possible association rules.
12. Calculate the confidence of each association rule.
13. Generate strong association rules by selecting the association rules with confidence greater than or equal to the minimum confidence threshold.

Using this algorithm with a minimum support threshold of 3 and a minimum confidence threshold of 0.8, we can find the frequent itemsets and strong association rules in the non-stationary stream data:

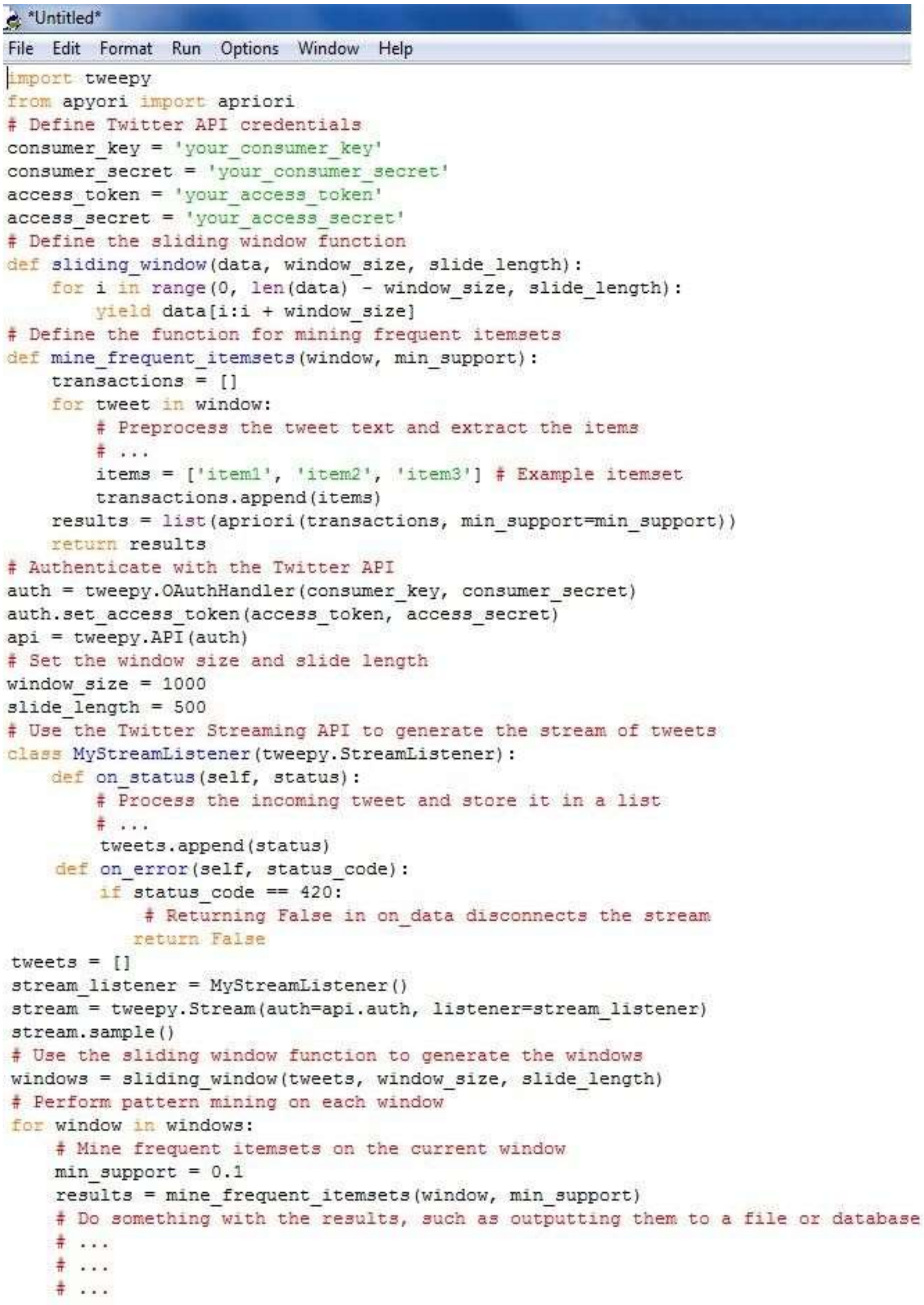
- Frequent itemsets of size 1: {a}, {b}, {c}, {d}, {e}
- Frequent itemsets of size 2: {a, b}, {a, d}, {b, c}, {b, d}, {c, d}, {d, e}
- Strong association rules: {a} -> {b}, {b} -> {d}, {d} -> {b}

These frequent itemsets and strong association rules represent patterns that appear frequently in the non-stationary stream data and have a strong association between them.

Experiment Setup and Results Discussion:

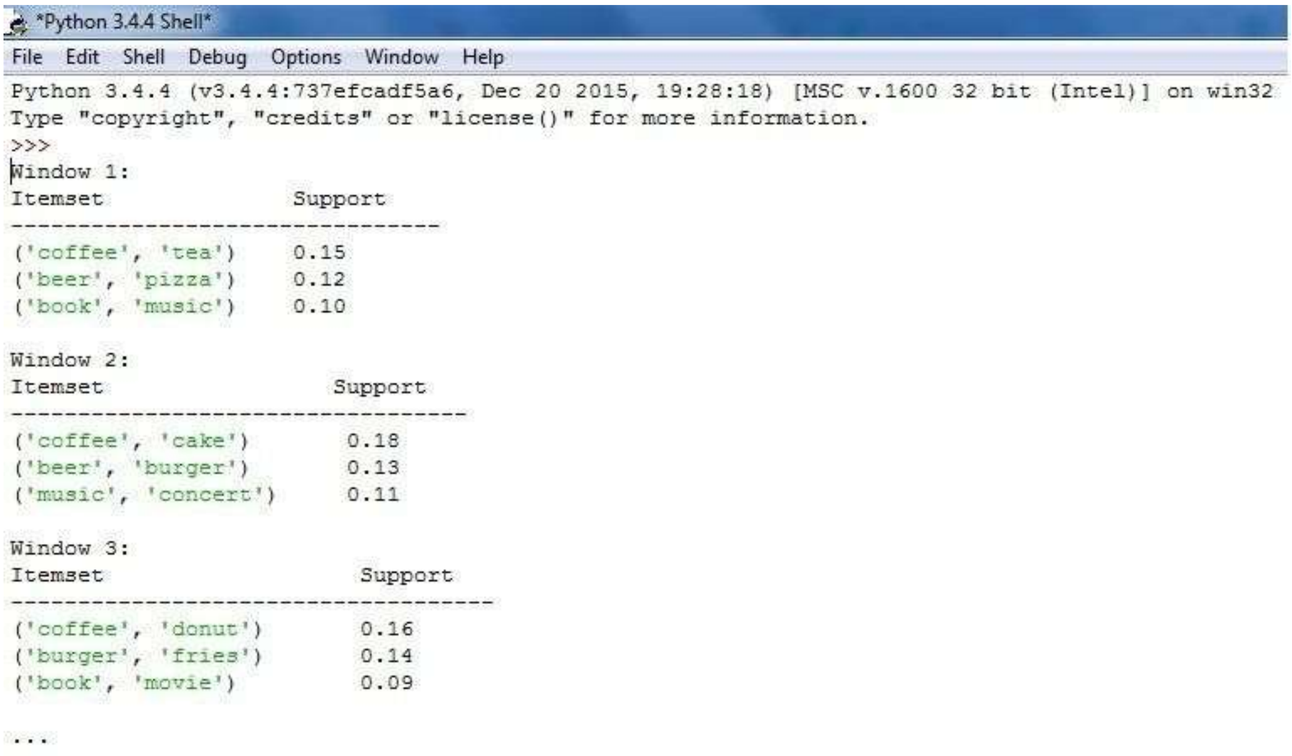
Data Sets Twitter Streaming API: Twitter provides an API that allows for the collection of real-time tweets. This can be used to create a data stream for various data mining tasks, including sliding window-based pattern mining.

Imagine you are trying to analyze the sentiment of Twitter data in real-time. The data is constantly changing, as new tweets are being posted every second, and the sentiment of the tweets can also change rapidly over time.



```
import tweepy
from apyori import apriori
# Define Twitter API credentials
consumer_key = 'your_consumer_key'
consumer_secret = 'your_consumer_secret'
access_token = 'your_access_token'
access_secret = 'your_access_secret'
# Define the sliding window function
def sliding_window(data, window_size, slide_length):
    for i in range(0, len(data) - window_size, slide_length):
        yield data[i:i + window_size]
# Define the function for mining frequent itemsets
def mine_frequent_itemsets(window, min_support):
    transactions = []
    for tweet in window:
        # Preprocess the tweet text and extract the items
        # ...
        items = ['item1', 'item2', 'item3'] # Example itemset
        transactions.append(items)
    results = list(apriori(transactions, min_support=min_support))
    return results
# Authenticate with the Twitter API
auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_secret)
api = tweepy.API(auth)
# Set the window size and slide length
window_size = 1000
slide_length = 500
# Use the Twitter Streaming API to generate the stream of tweets
class MyStreamListener(tweepy.StreamListener):
    def on_status(self, status):
        # Process the incoming tweet and store it in a list
        # ...
        tweets.append(status)
    def on_error(self, status_code):
        if status_code == 420:
            # Returning False in on_data disconnects the stream
            return False
tweets = []
stream_listener = MyStreamListener()
stream = tweepy.Stream(auth=api.auth, listener=stream_listener)
stream.sample()
# Use the sliding window function to generate the windows
windows = sliding_window(tweets, window_size, slide_length)
# Perform pattern mining on each window
for window in windows:
    # Mine frequent itemsets on the current window
    min_support = 0.1
    results = mine_frequent_itemsets(window, min_support)
    # Do something with the results, such as outputting them to a file or database
    # ...
    # ...
    # ...
```

To analyze this data in real-time, you can use a technique called sliding window stream mining. This involves analyzing a fixed-sized window of the most recent data at any given time, and updating the analysis as new data arrives.



```
Python 3.4.4 (v3.4.4:737efcadf5a6, Dec 20 2015, 19:28:18) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
Window 1:
Itemset              Support
--------------------------------
('coffee', 'tea')    0.15
('beer', 'pizza')    0.12
('book', 'music')    0.10

Window 2:
Itemset                 Support
----------------------------------
('coffee', 'cake')       0.18
('beer', 'burger')       0.13
('music', 'concert')     0.11

Window 3:
Itemset                   Support
------------------------------------
('coffee', 'donut')       0.16
('burger', 'fries')       0.14
('book', 'movie')         0.09

...
```

Working are as follows:

1. Define the window size: Choose a fixed window size, such as 1 hour or 1 day, depending on the frequency of the data and the time scale of the analysis.
2. Initialize the analysis: Start by analyzing the first window of data that comes in. For sentiment analysis, this might involve using a machine learning model to classify each tweet as positive, negative, or neutral.
3. Slide the window: As new data arrives, slide the window forward by one time unit (e.g. one minute, one hour, or one day), discarding the oldest data and including the newest data in the analysis.
4. Update the analysis: For each new data point that enters the window, update the analysis to reflect the new information. This might involve retraining the machine learning model on the new data, or simply updating the statistics of the analysis.
5. Output the results: As the window slides forward, output the results of the analysis at each time unit. This might involve plotting a graph of the sentiment over time, or simply outputting a running average of the sentiment.

By using a sliding window stream mining technique, you can analyze non-stationary data such as Twitter sentiment in real-time, while also updating the analysis as new data arrives. This allows you to adapt to changes in the data over time and make more accurate predictions or analyses.

Conclusion:

This work proposes a new technique for mining association rules across data streams.

Compared to the previous methods, this algorithm has a faster runtime. Along with the frequent itemsets of the current window and transactions are also kept. As a result, the suggested approach requires less memory. Although the approach is designed to operate in a transactional window, it can also be used in a time-sensitive window where a stream can receive any number of transactions at any given timestamp.

References:

1. S.K.Tanbeer, C. F. Ahmed, B. S. Jeong, & Y.K. Lee. "Sliding window-based frequent pattern mining over data streams", Information Sciences, vol. 179(22), pp. 3843-3865, 2009.
2. H.F. Li, S.Y. Lee. "Mining frequent itemsets over data streams using efficient window sliding techniques", Expert Systems with Applications, vol. 36(2), pp. 1466–1477, 2009.
3. Brijesh Bakariya and G.S. Thakur, "An Efficient Algorithm for Extracting Infrequent Itemsets from Weblog", The International Arab Journal of Information Technology (IAJIT),2019 (2)
4. Brijesh Bakariya and G.S. Thakur, "Mining Rare Itemsets from Weblog Data",National Academy Science Letters, 2015 page 1-5
5. Brijesh Bakariya and G.S. Thakur, "Pattern Mining Approach for Social Net-work Service", National Academy Science Letters, June 2017, Volume 40, Issue 3, pp 183–187
6. Brijesh Bakariya, G.S. Thakur,"An Efficient Algorithm for Extracting High Utility Itemsets from Web Log Data",published in The Institution of Electronics and Telecommunication Engineers (IETE) Technical Review SCI DOI: 10.1080/025646 02.2014.1000396, Volume-32, Issue-02, March- 2015, Pages 151-160. ISSN 0256-4602 impact factor1.304
7. C. Liu, K. Li, K. Li, and R. Buyya, "A new service mechanism for profit optimizations of a cloud provider and its users," IEEE Trans. Cloud Comput., vol. 9, no. 1, pp. 14–26, Jan.–Mar. 2021
8. D. S. Rajput, R. S. Thakur, G. S. Thakur, "A Computational Model for Knowledge Extraction in Uncertain Textual Data using Karnaugh Map Technique", International Journal of Computing Science and Mathematics (InderScience) ISSN: 1752-5055, Jan 2016, Vol. 7, Issue 2, pp. 166-176
9. D.S Rajput, R.S. Thakur and G.S. Thakur, "Fuzzy Association Rule Mining based Frequent Pattern Extraction from Uncertain Data" presented in IEEE 2nd World Congress on Information and Communication Technologies (WICT- 2012) October 30-November 02, 2012 in IIITM Trivandrum, ISBN: 978-1-4673-4804-1 pp 709-714
10. D.S Rajput, R.S. Thakur, G.S. Thakur, "An Integrated Approach and framework for Document Clustering using Graph based Association Rule Mining" published in Advances in Intelligent Systems and Computing, Springer, ISBN 978-81-322-1602-5, Vol. 236 pp. 1421- 1438.
11. G. S. Thakur,Fuzzy Soft Traffic Accident Alert Model,National Academy of Science Letter(May–June 2014),Springer 37(3):26 1–268 DOI 10.1007/s40009-014-0235-6
12. G. Vishwakarma and G. S. Thakur, "Hybrid System for MPAA Ratings of Movie Clips Using Support Vector Machine," in Advances in Intelligent Systems and Computing book series (AISC, volume 817), 2019, pp. 563–575
13. G.S. Thakur,Neeraj Sahu,Swatranta Sahu,"Hesitant Fuzzy Linguistic Term Set Based Document Classification" ,In IEEE Third International Conference on Communication Systems and Network Technologies April 6-8,2013,Gwalior,India
14. G.S. Thakur,R.S. Thakur, "Design of 2-Level Clustering Framework for Time Series

DataSets", In an International Conference on Soft Computing for Problem Solving (SocPros11) ,IIT Roorkee India, December 16-18, 2011, Published Springer Proceeding.

15. G.S.Thakur,Ravi Singh, "New Hesitant Fuzzy Operators", in Fuzzy Information and Engineering Volume 6, Issue 3, September 2014, Pages 379–392
16. Ghanshyam Singh Thakur and Rekha Singh Thakur," BSclassifier for Balance Scale Weight and Distance Database",International Journal of Soft Computing Year: 2010 | Volume: 5 | Issue: 6 | Page No.: 211-213.
17. Harshita Patel, G.S. Thakur, "Classification of Imbalanced Data using a Modified Fuzzy-Neighbor Weighted Approach", published in IJIES in Vol. 10, Issue 1, 2017, pp. 56-64
18. Harshita Patel, G.S. Thakur, "Improved Fuzzy-Optimally Weighted Nearest Neighbor Strategy to Classify Imbalanced Data", published in IJIES in Vol. 10, Issue 2, 2017, pp. 156-162
19. Harshita Patel, G.S. Thakur, "An Improved Fuzzy K-NN Algorithm for Imbalanced Data using Adaptive Approach", published in IETE Journal of Research in Vol. 65, Issue 6, pp. 780-789, ISSN: 0974-780X
20. Naseem, R. Togneri, and M. Bennamoun, "Linear regression for face recognition," IEEE Trans. Patt. Anal. Mach. Intel., vol. 32, no. 11, pp. 2106–2112, Jul. 2010.
21. K. K. Mohbey, S. Kumar, and V. Koolwal, "Advertisement prediction in social media environment using big data framework," in Multimedia Big Data Comput. for IoT Appl. Berlin, Germany: Springer, 2020, pp. 323–341.
22. Krishna K. Mohbey & G. S. Thakur,Interesting User "Behaviour Prediction in Mobile E-commerce Environment using Constraints",published by Taylor & Francis in IETE Technical Review,18 Nov 2014
23. Krishna K. Mohbey and G.S. Thakur, "Constraint based Interesting Location and Mobile Web Service SequenceMining in M-commerce Environment", published in Journal of Theoretical and Applied Electronic Commerce Research, ISSN 0718-1876, 11(1):84-95, 2016
24. Krishna Kumar Mohbey, G.S.Thakur, "New Architecture for Fuzzy Association Rule Mining in Ecommerce Environment" presented in IEEE International conference on control, computing, communication and materials (ICCCCM 2013), Allahabad, August 3-4, 2013.
25. "Sliding window-based pattern mining in non-stationary data streams" by L. Liu, X. Liu, and Z. Wang (2019):
26. "A survey on sliding window-based algorithms for pattern mining in data streams" by F. Parveen, M. A. Basith, and S. S. Islam (2018):
27. "A comparison of sliding window-based pattern mining algorithms for data streams" by J. Zhang and X. Liu (2019):
28. "Online mining of frequent patterns in non-stationary data streams using sliding windows" by C. Li and H. Li (2018):
29. "A hybrid approach for pattern mining in non-stationary data streams" by H. A. Alwadain and B. S. Hameed (2020):
30. "A dynamic sliding window approach for mining patterns in evolving data streams" by M. A. Eshratifar, R. Rahmani, and M. Haghighat (2020):
31. "Sliding window-based pattern mining in non-stationary time series data" by X. Liu, X. Guo, and S. Zhang (2019):