A Novel Change Detection Approach on Content Delivery Network (CDN) using Data Streaming Analytics

1st Indra Awal Priyanto, 2nd Wolosz Andras, 3rd Kadar Zoltan, 4rd Baba Adam
Faculty of Informatics
Eotvos Lorand University

***Abstract*—Detecting changes in a data streaming especially with Content Delivery Network (CDN) data is an important area of research with many applications. In this paper, we present a novel method for the detection and estimation of change. In addition to providing statistical guarantees on the reliability of detected changes, our method also provides meaningful descriptions and quantification of these changes. Our approach assumes that the points in the stream are independently generated, but otherwise makes no assumptions on the nature of the generating distribution. Thus our techniques work for both continuous and discrete data. In an experimental study we demonstrate the power of our techniques.**

Keywords: Change Detection, Data Streaming, Data Streaming Analytics, CDN

I. Introduction

In many applications, data is not static but arrives in data streams, especially when we work with Content Delivery Network (CDN) data in Big Data format. Besides the algorithmic difference between processing data streaming and static data, there is another significant difference. For CDN static datasets, it is reasonable to assume that the data was generated by a fixed process, for example, the data is a sample from a static distribution. But a CDN data stream has necessarily a temporal dimension, and the underlying process that generates the data stream can change over time [17, 1, 23]. The quantification and detection of such change in CDN environment is one of the fundamental challenges in data stream settings.

Change has far-reaching impact on any data processing algorithm. For example, when constructing data stream mining models [17, 1], data that arrived before a change can bias the models towards characteristics that no longer hold. If we process queries over data streams, we may want to give separate answers for each time interval where the underlying data distribution is stable. Most existing work has concentrated on algorithms that adapt to changing distributions either by discarding old data or giving it less weight [17]. However, to the best of our knowledge, previous work does not contain a formal definition of change and thus existing algorithms cannot specify precisely when and how the underlying distribution changes.

In this paper we make a first step towards formalizing the detection and quantification of change in a static CDN data and then going through with CDN data streaming. We assume that the data points are generated sequentially and independently by some underlying probability distribution. Our goal is to detect when this distribution changes, and to quantify and describe this change.

It is unrealistic to allow data stream processing algorithms enough memory capacity to store the full history of the stream. It is make sense that we perform statistical analysis using CDN static data before going through to streaming data to get better performance model. Therefore we base our change detection algorithm on a two-window paradigm. The algorithm compares the data in some "reference window" to the data in a current window. Both widows contain a fixed number of successive data points. The current window slides forward with each incoming data point, and the reference window is updated whenever a change is detected. We analyze this paradigm and develop algorithms (or tests) that can be supported by proven guarantees on their sensitivity to change, their robustness against raising false alarms, and their running time. Furthermore, we aim to obtain not only reliable change detection, but also a comprehensible description of the nature of the detected change.

A. Applications

A change detection test with the above properties has many interesting applications:

Quality of Service. User access on CDN providers such as Netflix, Spotify, Youtube, etc, as one of the widely access on a "day-to-day" basis of the internet access. Over the past decades, the demand for Content access through CDN provider has surged as many internet companies strive to fulfil user demand of content and provide sophisticated service management. However, this immense growth often make sudden changes and overloads the networks and caused network blackout. It caused significant financial loss when users disable to accessing content. This inflicts a severe loss to the users and further affects the 'Quality of Experience' of users. This could be avoidable by taking proper measures of this type of sudden change in the network which indeed gain the trust and builds the confidence of the users over the CDN provider.

Data Mining. Organizations base their policies and critical decisions on the analyses of the vast volumes of data, or "Big

Data". These CDN datasets in particular fall within the big data paradigm. The five Vs – Volume, Velocity, Variety, Veracity, and Value [11] are the primary characteristics of big data. There are several big data processing tools, such as Hadoop and Spark, which enable the organizations to collect and manage huge data. Recently, Spark, has gained popularity owing to its unified data analytics engine which can handle massive datasets. Spark offers a variety of abstractions, including data frames (DF) and resilient distributed data frames (RDD), which are immutable, support persistence, are cacheable, and are partitioned into multiple data partitions thereby supporting parallel and distributed operations. Additionally, Spark offers extensions for various programming languages like Scala, Python, Java, and R. Further, Spark extends its support to SQL with SparkSQL, developing scalable machine learning algorithms with SparkMLlib, and streaming analytics with Discretized streams (DStreams) as the abstraction. DStream, is a collection of RDDs collected over time that facilitates batch stream processing. Spark can easily integrate with other ingestion tools such as Apache Kafka, Kinesis, or TCP sockets and other big data processing tools such as Hadoop, Hive, etc. making it to manage the continuous stream of data.

B. Statistical Requirements

Among the types of sudden changes discussed earlier, change detection has captivated our interest. CDN data history can be examined to detect sudden change within the network. We determined that the following scenarios might assist us in identifying sudden change: (i) network blackout, (ii) unusual user behaviour history, (iii) user access initiated from an unusual location, and (iv) content access involving huge amounts of request, which could increase the probability of sudden change. It is quite difficult to carry out such analysis in real-time within limited time interval. Therefore, there is a need to automate this process by employing various machine learning techniques. Detecting sudden change detection is indeed a binary classification task, where sudden change-point being are regarded as a positive class and non-sudden change-point as a negative class.

C. An Informal Motivation of our Approach

Owing to the significance of identifying change detection under the big data paradigm. Through this study, we addressed the sudden change detection in the big data paradigm using a scalable machine learning algorithms and efficient solution methodology in both static and streaming contexts. To achieve this, we conducted the study as follows:

- Firstly, in the static context, we developed machine learning models on the historic CDN data collected from open source. To create scalable ML models, we used Spark MLlib.
- Secondly, in the streaming context, we used DStreams to capture data at regular intervals and employed a sliding window based change detection approach.

In the current study, we addressed the following challenges in both static and streaming contexts as follows: (i) handling imbalance in the CDN datasets, (ii) developing scalable ML models to meet the requirements of the big data paradigm, (iii) in the streaming context, the latency should be very less and time-time to analysis need to be provided without compromising on the performance, (iv) the streaming methodology should support a large number of data streams, and (v) automating the entire change detection process under big data paradigm.

D. Our Contributions

In the current study, we analyzed an CDN open source data for change detection. As discussed earlier, this change detection problem has a significant impact on daily living has motivated us to work towards providing and effective and efficient solution. In this study, we performed change detection using binary classification tasks. To accomplish this, we employed several machine learning techniques: NB, LR, SVM, DT, RF, GBT, and MLP. The collected CDN data is highly unbalanced in nature. This data imbalance is handled by employing the following techniques such as SMOTE and its variants (SMOTE-ENN, SMOTE-Tomek), and ADASYN. Further, we employed the GAN variants (i.e., V-GAN and W-GAN) too owing to its recent advancements and achievements.

The major contributions of the proposed work are as follows:

- Proposed the resilient change detection framework under Spark.
- Explored and compared ML algorithms for change detection in the static context.
- Investigated the use of GAN variants (V-GAN and W-GAN) for handling data imabalance in change detection.
- Proposed sliding window approach based change detection using data streaming analytics thereby handling the latency and time-to-time decision.
- Handled the imbalance by employing various data balancing techniques.

II. Related Work

This section will cover pertinent research that has been used to detect sudden change-point in CDN provider related to change detection, Quality of Experience, etc. in both static and streaming contexts respectively.

Data mining algorithms play a quintessential role in detecting sudden change-point [1]. These consist of LR, MLP, DT, etc. When cutting-edge methods, such as semi-supervised and unsupervised techniques, are used to mine the data that arrives in a stream, it may also be discovered in almost real-time [2,3]. Online classification, online clustering, outlier detection, etc., are some of the most sophisticated approaches. A reasonable amount of research is reported concerning with change detection [1,4-5], however, change detection in CDN data is seldom the researched topic. To the best of our knowledge, one research article appeared dealing with CDN change detection, where a specialized language is proposed for proactive sudden change-point management in CDN data streams [3]. Depending, on how a user accessing content – at a network, a device, or internet region, there are three alternative options. Other sudden change-point can be perpetrated, and

as the resultant data varies from structured to semi-structured to unstructured, so does the method they can be detected. Wei et al. [15] observed various characteristics and challenges involved in the sudden change-point detection. The development of a scalable system for sudden change-point detection utilizing distributed settings was pioneered by Chan et al. [6]. The authors demonstrated the significance of considering scalability into account when developing the solution.

Chen et al. [7] proposed a binary support vector machine, a hybrid of SVM and genetic algorithm (GA). Here, the optimal mix of support vectors is chosen using a genetic algorithm. In order to estimate the distribution of the input data, self-organizing map (SOM) is also included. In order to fulfil real-time analytics, Dorronsoro et al. [8] proposed an online system for change detection based on the neural classifier. To detect sudden change, the authors proposed a non-linear variation of fisher's discriminant analysis. This strategy could separate a significant change activities from legitimate traffic. A real-time change detection system was presented by Quah et al. [9] that uses the SOM to identify sudden change-point and user behaviour online. Sanchez et al. [10] applied association rule mining to find the pattern of anomalous behaviour that would assist them to identify a sudden change-point. On the CDN data, the authors showed how well the suggested technique worked.

Wei et al. [15] used the traffic vector to identify and store user behaviour to handle this imbalance issue of CDN datasets. The authors proposed a contrast miner, that efficiently extracts critical information by mining the contrast patterns and identifying the unlawful behaviour. Due to the considerable increase in online activities, several users are flooding the most often popular content. There are a number of efforts proposed to identify such flooding in the literature [31]. Arya et al. [16] applied a spiking neural network is inspired by neuroscience, and applied it to several classification problems. Ravisankar et al. [27] proposed a change detection framework to identify the critical change-point ratios obtained by analyzing the financial statements. With this, it would be possible to assess whether the company is carrying out too much bandwith or content. The performance of the models is frequently hampered by the presence of missing values in the CDN data. Fraquad et al. [29] proposed a modified active learning framework, which addresses churn prediction and insurance change detection, uses the SVM to extract the if-then rules to predict sudden change.

Arya and Sastry [49] proposed a change detection based on predictive modelling and named it deep ensemble learning for detecting sudden change-point in real-time data stream. The authors tried to address the overfitting issues which is one of the major concern of the previous studies. They employed ensemble of deep learning network to identify sudden change-point. Mittal and Tyagi [48] discussed the pros and cons of various computational techniques for real time change-point detection. Abakiram et al. [50] proposed a real time model for change-point detection based on deep learning. The authors employed deep auto encoder to classify the sudden change-point. Carcillo et al. [47] proposed a scalable framework for change detection using spark. The authors integrated the ingestion tools such as Kafka, Cassandra into their framework and simulated the sudden change coming in the form of data streams. The employed three different mechanisms to identify the sudden change-point.

III. Methodology

This section starts with the proposed methodology in the static context and then continues with a discussion on the streaming context.

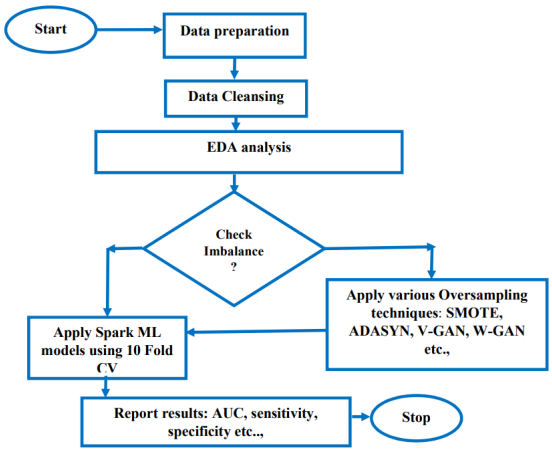

Fig. 1. Flowchart of the proposed methodology in the static environment

A. Binary Classification Methodology Under Spark

The proposed methodology is depicted in Fig. 1, and the block diagram is depicted in Fig. 2, and can be used in a static environment, where the model is trained on historical data. The process starts with the data collection step. In the current work, we collected CDN data from open source. The collected dataset contains user request that occurred in three different modes viz., (i) user bandwith, (ii) at the point internet network, and (iii) internet network latency. It includes critical private information related to the users. To maintain the data integrity and privacy of the users, we masked the feature names. Since, 'garbage data in results in garbage analysis out' is very suitable in machine learning. Therefore, to increase the richness of the data, we include the following data cleansing steps: (i) removing duplicate transactions because the same nformation is possessed twice, which indeed does not add any value, (ii) removing incomplete data within a dataset which includes corrupted features either due to human error or some other, and (iii) filtering out the features having almost null values. As we know, the missing data need to be dealt with through imputation techniques, but if the feature has more than 90% null values, these imputation techniques will fail miserably.

Hence, we dropped off the features having more null values. Thereof, the data can be analyzed to obtain vital insights and check assumptions with the help of generated statistical summary. All the features are normalized. Now, the dataset is observed to be of imbalance type. It is handled by applying various balancing techniques such as ADASYN, SMOTE,

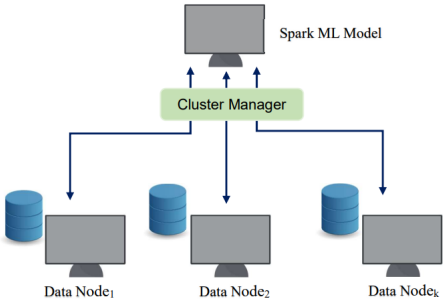

Fig. 2. Block diagram for Change Detection in static context

etc. The dataset is partitioned into training and test sets in the ratio 80:20. While doing so, stratified random sampling is adopted, where the same proportion of the both positive and negative classes is maintained in both the training and test sets. All of these balancing techniques are discussed in detail in the subsequent sections. On a note, while performing oversampling, the test dataset has to be kept intact. Once, the above operations are performed, the dataset is now stored in RDD.

Now, the modelling phase begins, where we build the machine learning model such as NB, LR etc. All of these models are built under the spark environment, and we utilized Spark MLlib. This is achieved by constructing a pipeline which consists of a group of stages. Each stage utilizes the data produced by the previous stage. The pipelines in Spark follow Directed Acyclic Graph (DAG), where the stages are specified in topological order. We used feature indexers available in Spark MLlib, to handle the features and then transform the entire data point into a single list. This is required for the Spark MLlib models. Then the model is trained on the training dataset and then tested on the test dataset. The above process is common for the rest of the spark models. While building the models, we used the grid search hyper-parameter tuning technique. On a note, we utilized 10-Fold CV technique and compared the performance thereof. All the metrics area under the receiver operator characteristic curve (AUC), sensitivity and specificity scores are presented and studied effectively.

```
1: Collect the dataset
2: Initialize ws ← window size
3: Initialize sl ← sliding interval
4: Create an Unbounded Table ← ϕ
5: Start StreamingContext(Time)
6: while IncomingStreams == True do
7:       UnboundedTable ← Append the IncomingStream
8:       If Window == Full then
9:             TrainData ← Window[:ws-1]
10:            TestData ← Window[ws]
11:            Train the Model on TrainData
12:            Validate the Model on TestData
13:            AUC ← Compute AUC score on TestData
14:            Sensitivity ← Compute Sensitivity on TestData
15:            Specificity ← Compute Specificity on TestData
16:            Report AUC, Sensitivity and Sensitivity
17:      end if
18:    Slide window by according to sl across UnboundedTable
19: end while
20: Stop StreamingContext
```

Fig. 3. Algorithm Sliding Window approach for Change Detection under streaming context

B. Sliding Window for Data Streaming

The methodology discussed above is very much suitable in the static environment. However, when it comes to processing the stream of data that needs to be handled with the specified interval of time demands a new mechanism. To meet these challenges and demands, we proposed a sliding window inspired streaming approach for CDN data change detection. The algorithm is presented in figure 3, and the block diagram is depicted in Fig. 4.

Spark provides DStream, an array of RDDs where each RDD comprises the value information collected at a certain interval of time. All the operations which are to be handled by DStream are processed in a streaming context. Hence, one needs to initialize the streaming context at the beginning of the process and need to close it once the operations are finished. All the variables, models, broadcasting variables etc., are confined to this streaming context. On a note, Spark provides the batch stream processing. Hence, the operations are performed on the data which is collected in the form of batches. By using the ingestion tools, streaming data is processed into the HDFS cluster. In our context, we didn't use any ingestion tool. However, we simulated the data ingestion by using QStreams which is provided by Spark. We created an unbounded data frame, where the streams of data are coming in multiple batches.

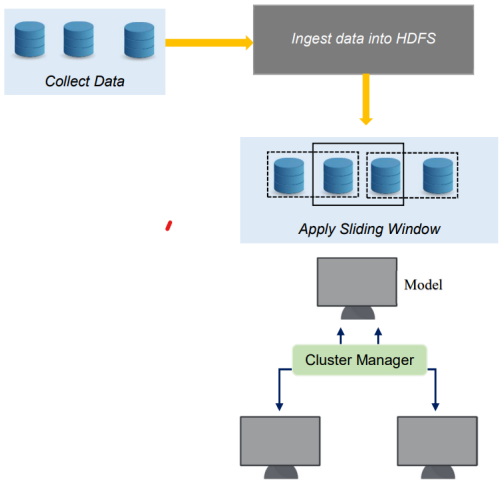

Fig. 4. Block diagram for Change Detection in streaming context

In our approach, we have utilized the sliding window approach. The sliding window approach is designed as follows: transactions data streams are collected in the form of batches with the specified time interval. The data collected thus far collected over a particular interval of time is sitting in a single RDD. The number of RDDs collected are dependent on the window size. Too much window size increases the latency. Let the sliding window length be ws, and then the training is done on the first 'ws-1' batches and testing on the latter batch of the sliding window. If the required number of batches has not yet been collected, it will wait until it is full. As discussed earlier, we know that a large window size demands more previous data to be processed. It further increases the latency too. By considering these criteria, we fixed the window size as 2. By

doing so, we are restricting ourselves to the most recent data. Hence, the underlying machine learning model is trained on the first RDD of the window and then tested on the second RDD of the window. We kept the stride, the moving speed of the window to be 1. Hence, now the second RDD of the second window becomes the first RDD of the next window. This process is continued until the new data is available and ingested into the cluster.

C. Resilient Distributed Dataset (RDD)

Spark offers the abstraction known as Resilient Distributed Dataset (RDD). It is a collection of dataset partitions that are dispersed throughout the cluster and is immutable. This makes use of Spark to carry out the simultaneous execution. In the event of a failure, it has fault tolerance features that are helpful for retrieving the data. It primarily supports (i) Transformations and (ii) Actions, which are two distinct procedures. Transformations are the set of operations which are used to transform the dataset from one form to other. On each element of RDD, these operations are carried out. Actions are aggregated operations, and hence the result is collected back to the driver. Lazy evaluation is supported by Spark RDD. In order to apply the changes whenever necessary, it first remembers them.

D. Discretized Stream

Fig. 5 shows Discretized Stream (DStream) another abstraction made available by Spark Streaming. It is obvious that there is a constant flow of information in a streaming environement. As a result, DStream contains an ongoing collection of RDDs. Here each RDD consists of the information gathered during a specific period of time. These DStream RDDs allow for the application of all operations, including Transformations and Actions. DStream is an immutable distributed stream that enables parallel operations, just as RDDs. The operations which are applied over DStream is inherently applied over these RDDs.

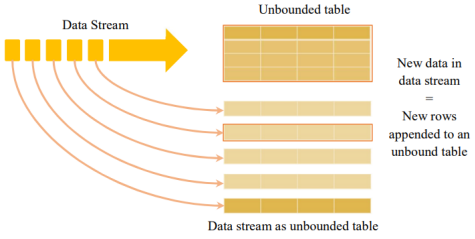

Fig. 5. DStreams

E. Queue Streams Window Operations

In the real world, the dataset is collected over a certain time frame. Such collected streams are processed in batches using Spark. As a result, the underlying queue of RDDs is considered as a batch of RDDs. As a result, all operations are performed on the batch of RDDs. To support this, Spark has an API called Queue Stream that allows users to create DStreams from a collection of RDDs. It is useful in generating an input stream from a queue of RDDs amassed over time. Such generated streams may be handled sequentially or simultaneously.

Additionally, Spark also provides the leverage of applying window operations, which is useful to apply transformations over a sliding window of data. All of the RDDs included in the window are merged, and the corresponding operations are carried out over it, as the window glides over a DStream. On a note, all these window operations are applied based on two important factors which are as follows:

- window length: which defines the duration of the window;
- sliding interval: the interval at which the window operation is performed.

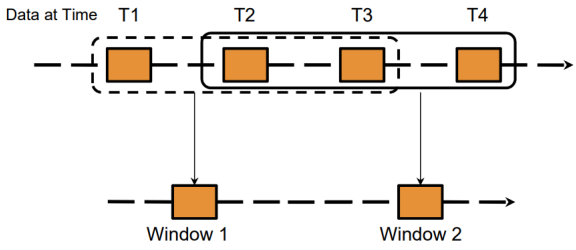

Fig. 6. Sliding Window Operation on Dstreams (window length size = 3, sliding interval=1)

F. Evaluation Measure

We considered AUC for measuring the robustness of the employed classifier. In addition to AUC, we also considered sensitivity and specificity as the other measures for the binary classification. **AUC**

It is proven to be a robust measure while handling imbalanced datasets and is an average of specificity and sensitivity. The mathematical representation of AUC is given in Eq. (1).

$$AUC = \frac{(Sensitivity + Specificity)}{2} \tag{1}$$

Sensitivity It is also well known as True Positive Rate (TPR). It is the ratio of the positive samples that are truly predicted to be positive to all the positive samples. The mathematical notation is given in Eq. (2).

$$Sensitivity = \frac{TP}{TP + FN} \tag{2}$$

where TP is a true positive, and FN is a false negative.
Specificity

$$Specificity = \frac{TN}{TN + FP} \tag{3}$$

where TN is a true negative, and FP is a false positive.

IV. Result

As discussed in Section 3, all the experiments followed the same identical data cleansing, and data pre-processing technique. We observed that the CDN dataset collected from

open source is highly imbalanced. Hence, we incorporated various balancing techniques viz., SMOTE, SMOTE-ENN, V-GAN, etc. It is to be noted that all the experiments are performed in 10-Fold Cross Validation (CV). In the imbalanced classification datasets, AUC is proven to be a robust measure which is quite evident in previous case studies such as credit card fraud detection. Hence, while choosing the best model, AUC is given more preference. The corresponding results are presented in Table fig 7. Further, we also discussed sensitivity and specificity scores in Table 4 and Table 5 respectively.

Out of all ML models, MLP turned out to be the best model in terms of AUC (refer to Table fig 7) after balancing the dataset with the V-GAN balancing technique. The second and third place is secured by GBT and RF respectively. It is interesting to note that both RF and GBT obtained similar AUC. The same is also reflected in sensitivity and specificity scores. As a decision maker, while choosing the best model the complexity of the model should also need to be considered along with the numerical AUC. In such a case, RF turned out to be the best model because of its less number of learnable parameters than MLP and low complexity than GBT. Further, it is observed to be having a very minor difference (AUC of 0.2%) in terms of mean AUC when compared to the other two best models. Overall, all the tree-based models performed relatively well when compared to other ML models. In terms of balancing techniques, V-GAN clearly outperformed the other balancing techniques in obtaining the best AUC.

There is another tie-breaker which can be considered while choosing the best model. It is a well-known fact that DT is the explainable model in the tree-based models. RF and GBT are ensemble models. Hence, there are complex models that stand second in terms of explainable aspect. But if one closely observed that the AUC is 2% less than the RF and GBT which is quite not acceptable in critical applications like fraud detection. Hence, despite having lesser interpretability, RF and GBT are ranked over DT. Because of its numerical superiority and less model complexity, RF ranked as the best model.

Model	Balancing Technique						
	Imbalanced	SMOTE	SMOTE-Tomek	SMOTE-ENN	ADASYN	V-GAN	W-GAN
NB	0.721	0.775	0.775	0.777	0.692	0.897	0.798
LR	0.709	0.803	0.803	0.805	0.690	0.893	0.869
SVM	0.807	0.806	0.805	0.807	0.694	0.890	0.853
DT	0.755	0.823	0.823	0.824	0.775	0.953	0.904
RF	0.759	0.837	0.837	0.838	0.791	**0.975**	0.920
GBT	0.759	0.837	0.837	0.838	0.791	0.975	0.920
MLP	0.757	0.837	0.831	0.831	0.721	**0.977**	0.904

Fig. 7. Mean AUC obtained by various ML models under 10-Fold CV

Similar observations were noticed with respect to the sensitivity scores (refer to Table fig 8). As we know that sensitivity is also another critical parameter that artefacts the number of sudden change-point truly identified to be as change-point ones. MLP outperformed the rest of the model by obtaining higher mean sensitivity scores. However, when considering the model complexity and numerical superiority as tie-breakers, RF turned out to be the best model in terms of mean sensitivity.

On the other hand, a similar observation is made on the specificity score (refer to Table fig 9) which artefacts the number of non-sudden change-point truly identified to be as non-sudden change ones. Specificity is also another critical factor because no loyal user playing content should be tagged as a sudden change-point. It is observed that SVM obtained a sensitivity score of 1.0 yet the specificity score is remarkably inferior to tree-based models. After considering tie-breakers, we concluded that RF turned out to be the best model in the static context.

Model	Balancing Technique						
	Imbalanced	SMOTE	SMOTE-Tomek	SMOTE-ENN	ADASYN	V-GAN	W-GAN
NB	0.546	0.779	0.778	0.780	0.699	0.871	0.674
LR	0.540	0.782	0.781	0.783	0.660	0.787	0.745
SVM	0.736	0.775	0.774	0.775	0.632	0.781	0.707
DT	0.607	0.789	0.787	0.789	0.738	0.930	0.824
RF	0.618	0.773	0.773	0.776	0.769	0.961	0.850
GBT	0.618	0.773	0.773	0.776	0.769	0.961	0.850
MLP	0.613	0.781	0.781	0.781	0.675	**0.969**	0.825

Fig. 8. Mean Sensitivity scores obtained by various ML techniques on 10-Fold CV

Model	Balancing Technique						
	Imbalanced	SMOTE	SMOTE-Tomek	SMOTE-ENN	ADASYN	V-GAN	W-GAN
NB	0.895	0.772	0.773	0.774	0.685	0.924	0.922
LR	0.878	0.825	0.825	0.826	0.720	0.999	0.992
SVM	0.877	0.837	0.836	0.838	0.757	**1.0**	0.999
DT	0.902	0.857	0.859	0.859	0.813	0.975	0.991
RF	0.901	0.901	0.902	0.901	0.812	0.989	0.991
GBT	0.901	0.901	0.902	0.901	0.812	0.989	0.991
MLP	0.902	0.882	0.882	0.882	0.767	0.986	0.982

Fig. 9. Mean Specificity scores obtained by various ML techniques on 10-Fold CV

A. *Statistical testing of the results*

The two-tailed t-test analysis is conducted to check whether model A's numerical superiority over model B is purely a coincidence or, indeed, due to its superior nature. Hence, we conducted a pairwise t-test analysis with the best performing model to the rest. The t-test analysis is conducted on the AUC obtained by models over 10 fold cross validation which are reported in Table fig 7. The t-test results are reported in Table fig 10.

The null hypothesis is, H0: both the algorithms are statistically equal.

The alternate hypothesis is, H1: both the algorithms are statistically not equal.

The p-value determines whether to accept or reject the null hypothesis. The significance level is chosen to be 5%, and the degree of freedom is 18 (10+10-2). Hence, the p-value should be less than 5% to reject the null hypothesis. Table 6 infers that when the p-values are less than 0.05; hence the null hypothesis is rejected, and the alternate hypothesis is accepted in the t-test conducted between RF and DT. This concludes that in the static context, RF is more statistically significant than DT in terms of AUC. However, RF and GBT are proven to be statistically equal. However, in terms of complexity RF is less than GBT which makes RF to be considered as the best model.

B. *Streaming Context results*

Now, we will discuss the performance of the models in the streaming context. As we already discussed, the CDN dataset is highly imbalanced. Hence, the values collected over

Model	Parameter	
	t-statistic	p-value
RF vs GBT	0.0	1.0
RF vs DT	33.39	1.21×10^{-17}

Fig. 10. Paired t-test results

a window might not have any sudden change-point. Further, the binary classification models cannot be built when only one single class-related data point are given. Hence, in our approach, one assumption is made in the streaming context with respect to proportion of the negative and positive class transactions collected over a window. Hence, in every data stream, we will include the sudden change-point (positive class samples) with the collected CDN data. Hence, the balanced datasets are only considered for being evaluated in the streaming context. Further, latency is another critical parameter while evaluating the model in the streaming context. Hence, the models should be less complex. Therefore, we considered the simple and tree-based models and compared the performance. It is important to note that all the models are evaluated in a sliding window fashion, and the metrics such as AUC, sensitivity, and specificity are calculated and discussed in this section. After employing the balancing technique, the dataset has now become 10 lakh transactions. Each window comprises 1000 series values thereby making the number of windows as 1000.

Model	Balancing Technique					
	SMOTE	SMOTE-Tomek	SMOTE-ENN	ADASYN	V-GAN	W-GAN
LR	0.811	0.811	0.811	0.691	0.842	0.875
KNN	0.854	0.855	0.855	0.774	0.870	0.896
DT	0.889	0.889	0.889	0.810	0.908	0.909
RF	0.896	0.898	0.898	0.821	**0.910**	0.911

Fig. 11. Mean AUC scores obtained by various ML techniques over sliding windows

Further, here also, we considered AUC as the suitable metric to choose the best model, and the results are presented in Table fig 11. We presented AUC obtained in Fig. 14 to Fig. 19 to illustrate the outlier behaviour of the employed ML models. We further presented AUC obtained over different windows in Fig. 20 to Fig. 23 to give an abstract of the performance of various models. This will indeed give a depiction over the performance of ML model. Overall, RF model performed the best irrespective of the balancing technique. DT stands second on the list. Further, Fig. 18 shows that RF model has very fewer outlier behaviour which makes it more stable while identifying the positive class. However, DT showed higher variability which makes it be second despite having interpretability aspect. The same is observed in the sensitivity and specificity scores. Interestingly, RF not only showed superior performance in mean AUC, sensitivity and specificity scores but also the showed lesser variability which supports its robustness. The same observation is noticed with respect to other balancing techniques. V-GAN is empirically obtained to be an efficient balancing technique in this setup which is followed by SMOTE-ENN.

Model	Balancing Technique					
	SMOTE	SMOTE-Tomek	SMOTE-ENN	ADASYN	V-GAN	W-GAN
LR	0.806	0.806	0.806	0.667	0.848	0.914
KNN	0.860	0.861	0.861	0.746	0.828	0.859
DT	0.895	0.895	0.895	0.779	0.865	0.867
RF	0.899	0.895	0.895	0.793	**0.871**	0.873

Fig. 12. Mean Sensitivity scores obtained by various ML techniques over sliding windows

Model	Balancing Technique					
	SMOTE	SMOTE-Tomek	SMOTE-ENN	ADASYN	V-GAN	W-GAN
LR	0.815	0.816	0.816	0.715	0.836	0.836
KNN	0.849	0.850	0.850	0.801	0.913	0.933
DT	0.882	0.882	0.882	0.842	**0.950**	0.952
RF	0.893	0.892	0.892	0.810	0.948	0.949

Fig. 13. Mean Specificity scores obtained by various ML techniques over sliding windows

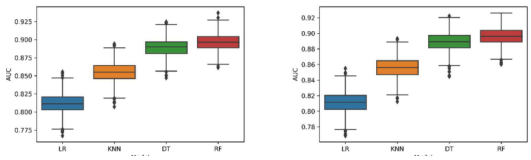

Fig. 14. Left: AUC obtained when SMOTE is employed in the streaming context. Right: AUC obtained when SMOTE-Tomek is employed in the streaming context

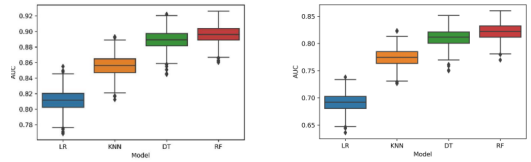

Fig. 15. Left: AUC obtained when SMOTE-ENN is employed in the streaming context. Right: AUC obtained when ADASYN is employed in the streaming context

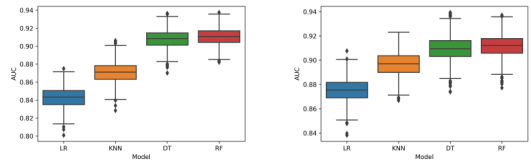

Fig. 16. Left: AUC obtained when V-GAN is employed in the streaming context. Right: AUC obtained when W-GAN is employed in the streaming context

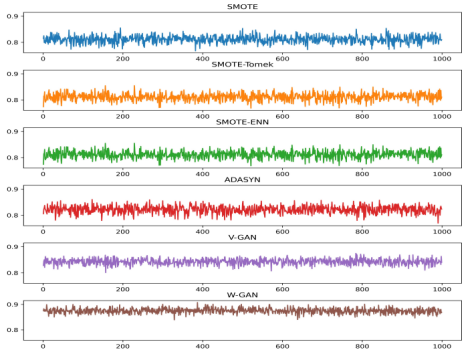

Fig. 17. AUC obtained by LR model over several windows

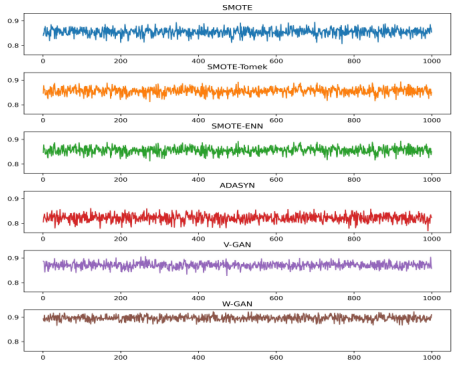

Fig. 18. AUC obtained by KNN model over several windows

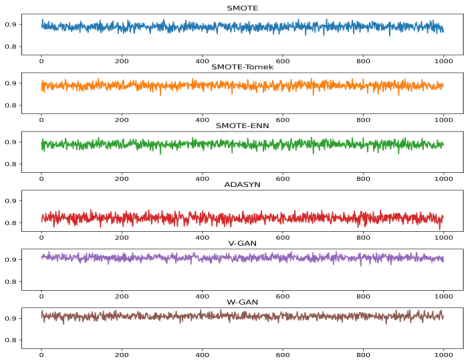

Fig. 19. AUC obtained by DT model over several windows

C. Statistical testing of the results

The Wilcoxon-test is a non-parametric test is performed to check whether the superior performance of model A over model B is purely a coincidence or due to its true superior nature. Hence, we conducted Wilcoxon test analysis on top-2 models by considering AUC obtained over several windows. The Wilcoxon test results are presented in Table fig 21.

The null hypothesis is, H0: both the algorithms are statistically equal.

The alternate hypothesis is, H1: both the algorithms are statistically not equal.

The p-value determines whether to accept or reject the null hypothesis. The significance level is chosen to be 5%. Hence, the p-value should be less than 5% to reject the null hypothesis. Table 10 infers that in the streaming context, RF is more statistically significant than the next-best performing model DT.

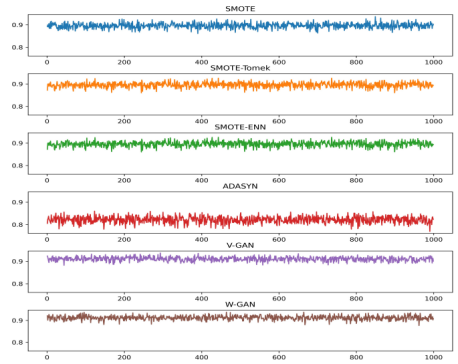

Fig. 20. AUC obtained by RF model over several windows

Model	Parameter	
	Wilcoxon test-statistic	p-value
DT vs RF	204017.0	$5.34x10^{-07}$

Fig. 21. Paired Wilcoxon-test results

V. Conslusion

In the current study, we developed a scalable ML models for the identification of sudden change-point in the CDN dataset collected from open source. We devised two different mechanisms suitable for both streaming and static environments respectively. The collected dataset is observed to be highly imbalanced. Therefore, we incorporated various sampling techniques viz., SMOTE, ADASYN, and GAN. Of all the sampling techniques, it turned out that V-GAN significantly outperformed the rest of the models. Further, we employed various machine learning techniques in the static and streaming contexts. In the static context, RF outperformed the rest of the models by achieving AUC of 0.973 followed by GBT with similar AUC. In the streaming context, RF secured first place. Further, we conducted statistical test which turned out that RF turned out to be statistically significant.

Our proposed sliding window approach helps to mitigate the risk of correctly identifying sudden change-point. However, we observed that there is one certain limitation which is as follows: every window assumes that it has received a certain amount of positive sample information. To mitigate this risk, while training the model, we need to provide some historic positive samples along with the collected data at the same timestamp. Otherwise, any ML technique fails to correctly identify the fraudulent samples. Alternatively, we can employ One Class Classification (OCC) model instead of binary classification models.

References

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.
[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
[4] K. Elissa, “Title of paper if known,” unpublished.
[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[7] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.