Center for Open Access in Science ▪ https://www.centerprode.com/ojit.html
Open Journal for Information Technology, 2025, *8*(2), 21-52.
ISSN (Online) 2620-0627 ▪ https://doi.org/10.32591/coas.ojit.0802.01021m

Reduction of Concept Drift Effect on Mobile Network Fraud Detection Using Deep Learning Markov Transition Encoding Dynamic LSTM Approach

James Mundia, Evans Kirimi Miriti, Stephen Mburu, Andrew Mwaura Kahonge & Christopher Kipchumba Chepken

University Of Nairobi, Faculty of Science, Nairobi, KENYA
Department of Computing and Informatics

Received: 22 July 2025 ▪ Revised: 11 October 2025 ▪ Accepted: 5 November 2025

Abstract

Background: Mobile network infrastructure has exponentially advanced in the past 20 years, resulting in tremendous evolution of the capabilities/services that the network can deliver. With these advancements, fraud activities have in equal measure metamorphosed into a complex web involving and spanning multiple touchpoints. Although different machine learning approaches have been developed and adopted, mobile network fraud remains dynamic and continually changes in form and means with time, resulting in degraded efficiency of these ML models. The phenomenon resulting in this degradation of efficiency with time is known as concept drift, which can be described as changes in the conditional distribution of the target variable, which is also referred to as output given the input features or the inputs, while the distribution of the inputs may still stay unchanged. This study aimed to analyze the effect of concept drift in relation to mobile network fraud detection, propose and develop an approach that reduces the concept drift effect when classifying and detecting mobile network fraud. **Method**: A quantitative research methodology was adopted using experiments as the core method. The dataset used for the study was an extract of DataStream from a Kenyan telco company and contains about 400,000 events or records generated by a process that randomly selected 65000 events for each month (Jan 2023 to June 2023) from the complete dataset. A deep learning model known as MTED-LSTM was developed and trained, then evaluated using the 6-month data stream. The other four commonly used approaches, i.e., Random Forest, Naïve Bias, Support Vector Machine, and Logistic Regression, were also developed and evaluated using the same data stream. Eight evaluation criteria were used to evaluate the effectiveness of the models. The data stream was used to evaluate whether the models were affected by the drift and how good they were in handling the concept drift effect. **Results**: Using each evaluation criterion, the models were tested through each month's data stream without retraining the model. The result indicated a gradual reduction of the effectiveness of the models with time, attributed to the concept drift effect. MTED-LSTM model showed to be more effective through the six-month data stream and least affected by concept drift as compared to other models developed in the study. **Conclusion**: The MTED-LSTM model effectively managed the effects of concept drift better due to its ability to store the previous model state on the time series DataStream, which is also referred to as memory. This is evidenced by the results, where all the performance indicators show that the model is least affected by the drift with time. It is worth noting that between the 4th and 5th months, all the other models drastically reduce their effectiveness, but the MTED-LSTM model recorded improved effectiveness during this time frame, resulting in a stable and smooth performance.

Keywords: mobile network, fraud, concept drift, long short-term memory, deep learning.

Correspondence: James Mundia, University Of Nairobi, Faculty of Science, Department of Computing and Informatics, Nairobi, KENYA.

1. Introduction

Mobile network infrastructure has exponentially advanced in the past 20 years, resulting in tremendous evolution of the capabilities/services that the network can deliver. The infrastructure has metamorphosed from a basic (GSM) Global System for Mobile Communications platform used to exchange SMS (short message service) and make calls from one handset to another to a complex integrated infrastructure carrying and hosting multiple services. I.e., Mobile network is currently acting as the core media of the E-commerce platform, Mobile Money platform with the likes of Mpesa (Kenya), E-Taxi platforms, E-delivery platforms, etc. (Yang, 2018). With this advancement, the data generation dynamics, Velocity, and variety have exponentially increased. With these developments, fraud activities have also metamorphosed, affecting multiple touch points on the Mobile network infrastructure and compounded by complex and huge data sets involved. In addition, fraudsters' capabilities have also advanced, resulting in new fraud patterns that are ever evolving, and the need for an advanced approach to address these challenges is inevitable to reduce the exposure for both customers and service providers (Evgeny et al., 2018).

Multiple approaches have been developed over time to address the fraud issue in the mobile network, which include Knowledge-based systems that rely on manually predefined rules and rule sets to differentiate between fraud and non-fraud activity. Rule-based methods are effective, but difficult to manage. A lot of work is required to define rules for every imaginable fraud case. Another drawback is that rule-based fraud detection systems need to be revised and updated frequently to cover newly discovered kinds of fraud (Apapan et al., 2018).

Machine learning approaches have recently been adopted to address the fraud issue. The commonly used ML approaches used in mobile network fraud include Support Vector Machine, Naïve-Bayesian, Random Forest, and Artificial Neural Networks (Kim et al., 2015; Sallehuddin et al., 2015; Saravanan et al., 2014). The major challenge that cuts across all the ML approaches is the reduced prediction efficiency with time due to Concept drift. In a non-stationary and dynamically changing mobile network environment, the change of data distribution over time results in a phenomenon known as concept drift. Concept drift, in summary, can be described as changes in the conditional distribution of the target variable, which is also referred to as output given the input features or the inputs, while the distribution of the inputs may still stay unchanged (Gama et al., 2013).

In this regard, we propose a dynamic deep learning approach to address the issue of concept drift when using automated ML tools to detect mobile network fraud. Deep learning, in definition, is a class of machine learning approach, where multiple layers of information-processing are stacked in hierarchical architectures for pattern classification and for representation or feature learning (Yoshua, 2015; Georgios et al., 2019; Ajay et al., 2019; Samira et al., 2018; Xizhao et al., 2020).

2. Background

2.1 *Machine Learning approaches*

Machine learning approaches, both supervised and unsupervised, have been used widely in detecting mobile network fraud and below are the commonly implemented approaches.

Support Vector Machine: SVM application in fraud detection was proposed and developed by Vapnik (1995) and is built on the basis of the theory of statistical learning, where learning is achieved through computation.

Naïve-Bayesian: Naïve Bayesian technique classifies instances by utilizing either the categorical or numerical nature of the records. By using the features or attributes that describe a record, the probability value is calculated for each individual record. Since the records are either numeric or can be represented in numerical form, the continuous values formula can be deployed in the Naïve-Bayesian technique (Saravanan et al., 2014).

Artificial Neural Networks: Artificial Neural Networks (ANN) is a representation of a simulated model of a biological neural network in the form of a non-linear learning algorithm (Ajith, 2005). Through the recursive adjustments of bias levels and weights, ANNs learn from the environment and continuously improve their accuracy.

Random forest: Random Forest is an ensemble algorithm built from multiple decision trees. The output of individual trees is combined to boost the model's generalization ability (Lakshmi et al., 2018). Decision trees, which work on splitting the nodes on all available variables to arrive at a split that results in the most homogeneous sub-nodes, are the backbone of RF (Gerard et al., 2015). Random has widely been used in fraud detection, especially financial fraud, but is also currently used in mobile network fraud detection due to its ease of implementation and general stability (Fidelis et al., 2024; Niveditha et al., 2019).

2.2 *Concept drift*

In a non-stationary and dynamically changing mobile network environment, the change of data distribution over time results in a phenomenon known as concept drift. Concept drift, in summary, can be described as changes in the conditional distribution of the target variable, which is also referred to as output given the input features or the inputs, while the distribution of the inputs may still stay unchanged (Gama et al., 2013). A case of example of a concept drift is a preference change of a user following news online in a continuous stream. In this case, news stream documents distribution remains unchanged, but the conditional distribution of the user's preference changes. In mobile network fraud, this is characterized by unpredictable fraudulent events that do not follow any particular pattern. i.e., a transaction (call, Money transfer, or SMS) with the same predictors or attributes can either be fraud or not fraud under different scenarios. In this case, the prediction output changes while the inputs remain the same (Mundia et al., 2024).

Majority of the techniques previously used to detect mobile network fraud suffer from the concept drift as a result of constant evolution in the fraud patterns and hence the need for an adaptive learning approach to update the model in a real time during its operation to counter the effects of concept drift (Geoffrey et al., 2016, Zhaohui et al., 2022; Sallehuddin et al., 2015).

Data streams can be divided into two types, static and dynamic. In static data streams, data appears in a fixed probability distribution, whereas with dynamic data streams, the probability distribution or the statistical characteristic of data changes over time in unforeseen ways (Zhaohui et al., 2022). A data stream D defined as $D= \{d_1, d_2, ..., d_i, d_{i+1}, ...\}$, $d_i= \{x_i, y_i = \{0, 1, 2, ...m\}\}$, with x_i being the eigenvalue of the sample space, y_i being the label of the sample and m represents the different categories tag value. Let's assume that a data chunk S_t at a time t obeys a distribution $F_t(X,Y)$, then if $F_t(\mathrm{X},\mathrm{Y}) \neq F_{t+1}(\mathrm{X},\mathrm{Y})$, then this is an indication that concept drift occurs at $t+1$.

In relation to supervised learning, concept drift can be defined as the change in the relationship between the target variable and the input variables that occurs over time (JOAO et al., 2014). The change of data distribution characteristics over time may be unpredictable, taking and assuming different forms, i.e., incremental, sudden, recurring, gradual, blip, and noise (Geoffrey et al., 2016; Zhaohui et al., 2022).

Bayesian Decision Theory describes a classification problem by the prior probabilities of the classes *p(y)* and the class conditional probability density functions *p(X|y)* for all classes *y* = 1,..., c, where c is the number of classes (Duda et al. 2001) This means that when classifying an instance, the decision is arrived with regards to the posterior probabilities of the classes, which for class *y* can be represented as

$$p(y|X) = \frac{p(y)p(X|y)}{p(X)} \qquad (1)$$

Where: $p(X) = \sum_{y=1}^{c} \quad p(y)p(X|y)$

So, we can formally define concept drift between time point t_0 and time point t_1 as

$$\exists X: p_{t0}(X, y) \neq p_{t1}(X, y) \qquad (2)$$

where p_{to} represents the joint distribution at time t_o between target variable *y* and the set of input variables *X*, and p_{t1} represents the same at time t_1 (Gao et al., 2007; Kelly et al., 1999). In other words, the class conditional probabilities *p(X|y)* may change, the prior probabilities of classes *p(y)* may change, resulting in a change in the posterior probabilities of classes *p(y|X),* which ultimately affect the prediction. The *figure 1* illustrates the 2 types of drifts (JOAO et al., 2014) which can be described as:

i. Real concept drift, which indicates a change in *p(y|X)*. this change can occur with or without change in *p(X)* (Salganicoff, 1997; Gao et al., 2007).

ii. Virtual drift occurs if the distribution of the incoming data changes, i.e., *p(X),* without affecting *p(y|X)* (Tsymbal, 2004; Delany et al., 2005). In the literature this kind of drift has been interpreted differently i.e. Widmer (1993) defines virtual drift as a drift that happens as a result of incomplete data representation as opposed to an actual concept drift, Tsymbal (2004) describes virtual drift as a change in the decision boundary as a result of change in data distribution. Other descriptions of the Virtual drift are features change (Gao et al., 2007) and sampling shift (Salganicoff, 1997).

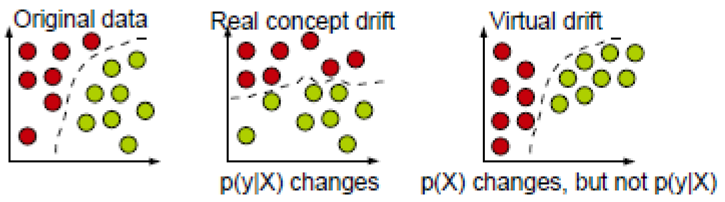


Figure 1. 2 Types of Drifts

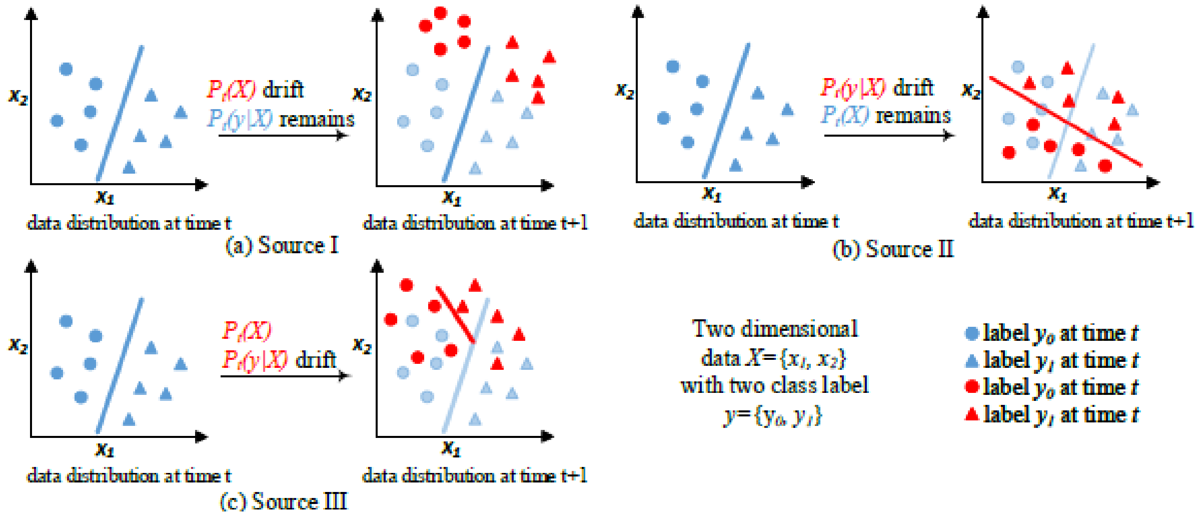
From the above illustration, Real concept drift affects or changes the class boundary, making the previous decision model absolute in classifying the new instances.

Concept drifts (both real and virtual) can be summarized to originate from the following 3 sources (Jie Lu et al., 2020):

i. *Source I*: $P_t(X) \neq P_{t+1}(X)$ while $P_t(y|X) = P_{t+1}(y|X)$, In this case, there is a change in data distribution *Pt(X)* while *Pt(y|X)* remains the same. Since a drift or change $P_t(X)$ does not change the decision boundary, this source of drift results in a virtual drift as illustrated in Figure 2(a).

ii. *Source II*: $P_t(y|X) \neq P_{t+1}(y|X)$ while $P_t(X) = P_{t+1}(X)$. This drift will result in a change of decision boundary as the expectation of *y* given *X* changes with time. The outcome of this source of drift is a decreased accuracy in classification or the real drift as illustrated in Figure 2(b).

iii. *Source III:* mixture of Source I and Source II, namely $P_t(X) \neq P_{t+1}(X)$ and $P_t(y|X) \neq P_{t+1}(y|X)$. The source of this drift is a change of both $P_t(y|X)$ and $P_t(X)$, which will result in a change in the decision boundary, hence, the outcome will be a real drift, as illustrated in Figure 2(c).



$P_t(X)$ drift
$P_t(y|X)$ remains
data distribution at time t
data distribution at time t+1
(a) Source I
$P_t(y|X)$ drift
$P_t(X)$ remains
(b) Source II
$P_t(X)$
$P_t(y|X)$ drift
(c) Source III
Two dimensional data $X=\{x_1, x_2\}$ with two class label $y=\{y_0, y_1\}$
● label y_0 at time t
▲ label y_1 at time t
● label y_0 at time t
▲ label y_1 at time t

Figure 2. Drift Sources

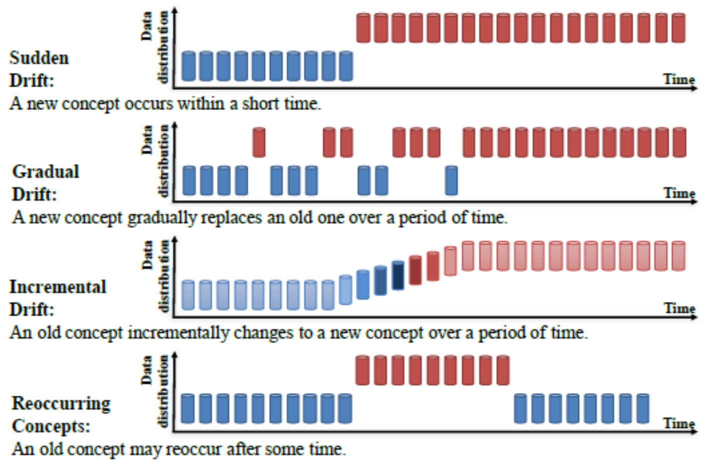
Drift can manifest in 4 different forms (Jie Lu et al., 2020; Zhaohui et al., 2022) as illustrated in Figure 3 and described as:

i. *Sudden drift:* This happens when a new concept occurs with a very short time, resulting in a sudden change of class boundaries in the case of a real concept drift.

ii. *Gradual drift*: This happens when a new concept gradually replaces the old concept over a period of time.

iii. *Incremental Drift:* This happens when the old concept incrementally evolves to a new concept over time. Similar to the gradual drift, the transition to the new concept takes a bit of time.

iv. *Reoccurring Drift:* In this case, a new concept occurs either suddenly or gradually, but later the old concept reappears after some time.



Sudden Drift:
A new concept occurs within a short time.
Gradual Drift:
A new concept gradually replaces an old one over a period of time.
Incremental Drift:
An old concept incrementally changes to a new concept over a period of time.
Reoccurring Concepts:
An old concept may reoccur after some time.
Data distribution
Time

Figure 3. Drift manifestations

2.3 *Objectives*

The main objective of this research is to evaluate the effectiveness of deep learning technique in the reduction of concept drift effect on classification accuracy in mobile network fraud detection.

The specific objectives are to:

1. Determine the effect of concept drift in mobile network fraud detection with respect to prediction accuracy.
2. Design and develop a deep learning model that maintains prediction accuracy in mobile fraud detection in the face of concept drift.
3. Compare the performance of the deep learning model with other techniques in mobile fraud detection where concept drift exists in the data.

2.4 *Research questions*

Below are the research questions that will be trying to answer in the research

1. How does concept drift affect the prediction accuracy on models used in mobile network fraud detection?
2. How can concept drift in mobile network fraud data be detected?
3. How can the deep learning approach be used in maintaining prediction accuracy in the face of concept drift when detecting mobile network fraud?
4. What will be the performance of a Deep Learning driven model as compared to other methods including anomaly detection approach previously used in detecting mobile network fraud with concept drifted data?

2.5 *Previous work*

In this section, we review different approaches and algorithms that have previously been used to address the issue of concept drift. Drift detection can be defined as the mechanisms and techniques that quantify and characterize concept drift by identifying the points of change of time intervals for the change (Basseville et al., 1994)

We can categorize methods or algorithms used to handle concept drift into three major categories, namely:

Error rate-based drift detection: The primary goal of these algorithms is to track the change in online classification error of the base classifier. Once the error crosses a particular threshold or is statistically significant, an alarm is triggered that will initiate a retraining or recalibration of the base model. Examples of these types of algorithms are Drift Detection Method (DDM) developed by Gama et al. (2004). This algorithm works by initially defining a time window by which the drift is measured. When new data is available, it is evaluated to detect whether, within the predefined time window, the overall online error rate has significantly increased. If the error rate reaches a warning level, DDM will start building a new leaner model, training it with the new data instances, but still using the old model to perform prediction. Once the new learner or model is trained and the error rate crosses the critical point, the algorithm switches to the new model for prediction. One of the major drawbacks of this approach was the cost of building and training a new model was very high especially if implemented in highly evolving environments like fraud detection. Also, because the old model is abandoned once the new model is implemented, the old information which can be useful in feature predictions is erased with the old model. To address these shortcomings, memory structure can be embedded into the algorithm to store useful information that be used by the newly built model. In addition, as opposed to creating an entirely new model, incremental training can be done on the old model resulting to reduced costs. Other similar implementations that have been developed are Local Drift Detection (LLDD) developed by Gama et al. (2004). This methodology focused on local drifts as opposed to the wholistic drifts which performed well in cases of remote or localized drifts. Baena-Garcia et al (2006) came up with another detection approach known as Early Drift Detection Method (EDDM) with the aim of detecting, training and switching to a new model before the drift affected the predictions significantly but it had the same drawbacks as DDM approach. S. Xu et al (2017) developed Heoffding's inequality-based Drift Detection Method (HDDM) which was based on Hoeffding's bounds with moving averages (A-test) This model was less costly as it performed increment training on the original model as opposed to creation of an entirely new model.

Data Distribution-based Drift Detection: The goal of this category algorithm is to quantify, by the use of a distance metric, the dissimilarity or the significant difference between the distributions of the historical and the new data streams. With this goal, this algorithm addresses the root source of the drift. If there is statistically enough evidence that there is a dissimilarity between the distribution of the historical and the new data streams, the system initiates a model upgrade process which will include retraining and recalibration of the model once the dissimilarity reaches a critical threshold a switch to the new model is committed (N. Lu et al,2014). Examples of algorithms that fall under this category are Statistical Change Detection for multidimensional data (SCD) by X. Song et al (2007) which continually monitored the data distributions changes of multi-dimensional data streams for significant changes then invoking a model retrain once a significant change is detected. N. Lu et al. (2016) developed Competence Model-based drift detection (CM) methodology, J. Shao et al. (2014) developed prototype-based classification model for evolving data streams called SyncStream and PCA-based change detection framework (PCA-CD) by S. Xu et al. (2017). The goal of these algorithms was to detect the change in data distribution in the data stream. The drawback of these algorithms was the assumption that drift was only caused by data distribution changes which is not usually the case in real life data stream where noise, and conditional distribution of outcome can also be sources of drifts.

Multiple Hypothesis Test Drift Detection: This set of algorithms performs more than one hypothesis testing, either to detect data distribution change or error rate change in the system. The goal of these algorithms is to detect any significant change in the drift of either the distribution or the accuracy. The multiple hypothesis testing can either be done in parallel or hierarchical (C. Alippi et al., 2008). Though these algorithms can be effective in detecting and acting on drifts, they are very complex to implement as different thresholds from the multiple hypothesis can easy trigger model switch instability (Y. Zhang et al,2017) examples of these some of the algorithm under this category are Rate drift detection (LFR) developed by W. Heng et al. (2015) which tracks and maintains the drift or change in True Negative rate (TN), True positive rate (TP), False Negative rate (FN) and False Positive rate(FP) in an online manner, Hierarchical Linear Four Rate (HLFR) by S. Yu et al. (2017) used multiple hierarchies to track both the error rate changes and data distribution changes resulting to a combined metric that tracked the drift. Two-Stage Multivariate Shift-Detection based on EWMA(TSMSD-EWMA) by H. Raza et al. (2015) used 2 stages one to track the error rate change and the other tracking the data distribution change resulting to a compound drift tracking metric. Another drawback of these algorithms is that they can be very unstable in highly attributed data stream like mobile network data streams.

According Jie Lu et al. (2020) any drift detection algorithm must answer one or more of the following 3 questions:

i. *When did the drift occur?:* Drift detector should be able to identify the timestamp t when the drift occurred, i.e.

$$\exists t: p_t(X, y) \neq p_{t+1}(X, y) \quad (3)$$

In this case, time *t* represents the time drift occurred.

ii. *How is the drift?*: This is the measure of severity of the drift at time *t* i.e.

$$S = f(p_t(X, y); p_{t+1}(X, y)) \quad (4)$$

where *f* is a function that measures the difference between two data distributions, *t* is the timestamp when the concept drift happened, and S is a non-negative value representing the severity of the drift.

iii. *Where did the drift occur?* This question is answered by identifying the region where concept drift occurred, which can be identified as the regions of conflict between the previous concept and the new concept. This can be defined as the data feature space X where $P_t(X; y)$ and $P_{t+1}(X; y)$ are significantly different.

3. Markov Transition Encoded Dynamic LSTM Framework

In section we discuss the proposed MTED-LSTM framework, a framework that continuously and dynamically learns from incoming data streams while maintaining the attributes relationships during the encoding phase. MTED-LSTM continuously monitors the Mobile network input stream and adapts to the new evolving fraud patterns.

The framework comprises 6 phases or layers as illustrated in Figure 4 and Figure 5.

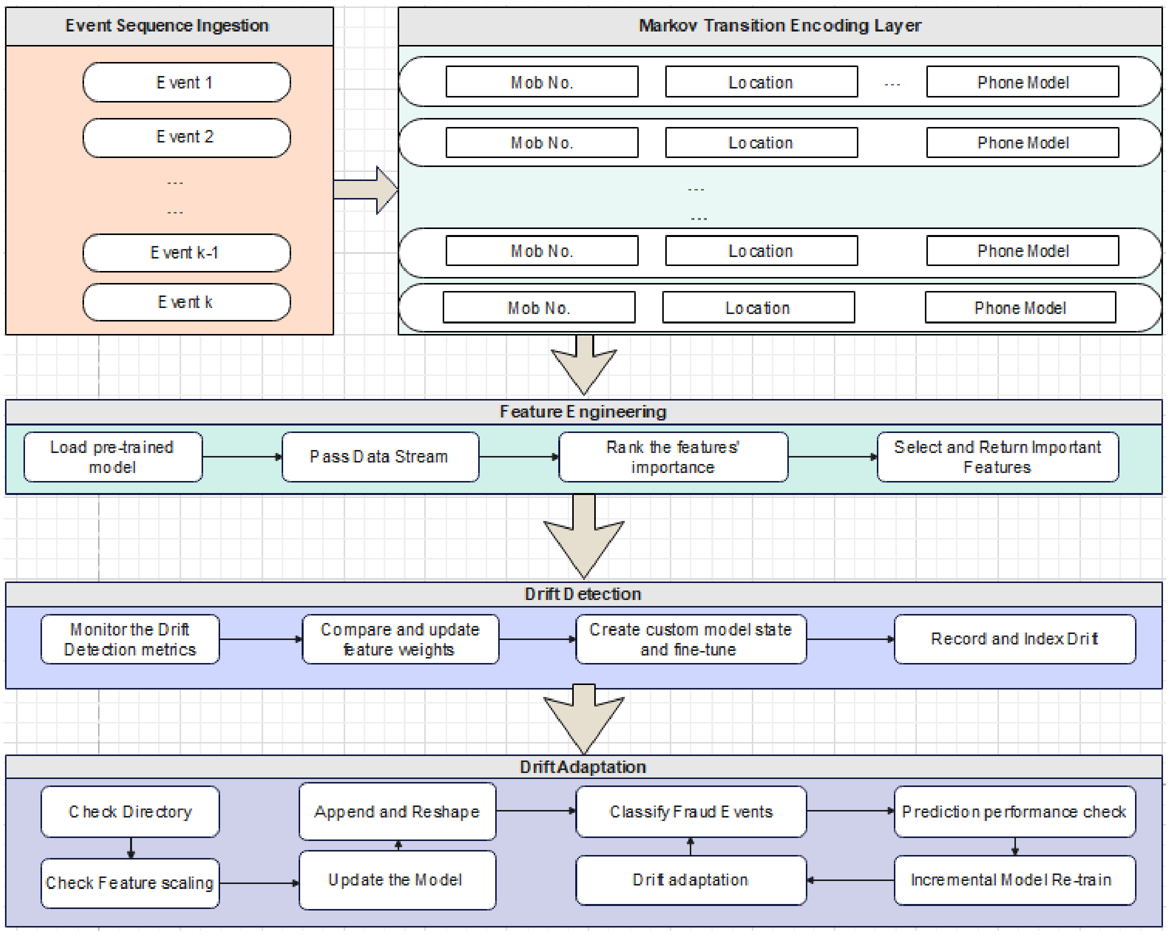


Figure 4. MTED-LSTM Framework

3.1 *Event Sequence Ingestion phase*

This is the first stage of the model where the preprocessed data stream is ingested into the system. One of the main functions in this stage is to make sure that the input data is passed into the next stage in a sequential manner, i.e., the events are arranged sequentially as they occur as this enforces the time series nature of the model. Each record or event is timestamped as per its occurrence. Any incomplete record that might have passed the preprocessing stage is dropped to ensure the stability of the model. Any invalid record that might have incorrect data type is also dropped at this stage.

The rate of ingestion into the model is controlled at this layer; that is, if there is an influx of records in the data stream, the smoothening happens at this layer, where the input into the next stage is throttled to ensure a smooth and consistent training phase, which occurs are the next phase. A subset of the data stream features or attributes that are relevant for the model training are allowed to pass to the next stage of the model to reduce data dimensions flowing to the next stage.

3.2 *Data stream encoding using Markov Transition phase*

The categorical data in the dataset must be encoded before training though one of the issues related to this is the loss of inter-relationships information amount the attributes which can be key ingredient in the training and testing of the model (Ruinan et at., 2018). To preserve the inter-relationships information in the data stream, we propose a framework similar to (Campanharo et al., 2011) used for encoding dynamical transition statistics, but we modify the framework by representing the Markov transition probabilities sequentially to preserve inter-relationships information in the time domain.

Let $n \times 1$ vector F_j represents the distribution at state j, with n being the number of attributes. Let M represents (n by n) Markov transition matrix governing the transformation of F_j into F_k, the distribution at state k, can represented as

$$F_k = M' \cdot F_j \qquad (5)$$

if n equals 3, the Markov matrix can be represented as.

$$M = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad (6)$$

with every element of the matrix, a_{jk}, giving the probability of transition to state k from the initial state j and these elements are hence referred to as Markov transition probabilities. With the assumption that the Markov transition matrix does not change i.e., Markov transition probabilities are unchanged over time, then the distribution after s periods can be archived by executing equation (5) s times to have:

$$F_{t+s} = (M^s)' \, . F_t \qquad (7)$$

When it comes to mobile network data, suppose for every event a_i there is a total number of k categorical features. Each event a_i is first encoded using label encoder method, that is all features are bin and one-hot coded into a binary vector. The indexes of the binary vector are extracted with a maximum numerical value of the length (l) of the binary vectors, denoted as V_i. Every categorical feature f_i is coded with a value of 1 on v_{ij}, where $a \leq j < b$ on the segment $s_i = [a, b)$. With this transition, every vector v_i in V is guaranteed k non-zero numerical values. Then by taking all non-zero numerical values from V_i to V_{i+1} a Markov Transition Field M is built as illustrated in the matrix 2 (XIN et al., 2024; Ruinan et at, 2018). The meaning of a transition probability m_{ij} in M can be interpreted as the probability of the likelihood of transition from f_x to f_y where i ∈ s_x and j ∈ s_y:

$$M = \begin{array}{c} \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} & m_{15} & m_{16} & \cdots & m_{1l} \\ m_{21} & m_{22} & m_{23} & m_{24} & m_{25} & m_{26} & \cdots & m_{2l} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ m_{l1} & m_{l2} & m_{l3} & m_{l4} & m_{l5} & m_{l6} & \cdots & m_{ll} \end{bmatrix} \\ \underbrace{\hphantom{m_{11}\quad m_{12}\quad m_{13}}}_{\substack{s_1 \\ \Downarrow \\ f_1}} \quad \underbrace{\hphantom{m_{14}\quad m_{15}\quad m_{16}}}_{\substack{s_2 \\ \Downarrow \\ f_2}} \; l \quad \cdots \quad \underbrace{\hphantom{m_{1l}}}_{\substack{s_i \\ \Downarrow \\ f_i}} \end{array} \tag{8}$$

Applying this approach while encoding the input data stream ensures the inter-relationships between the attributes is maintained before passing this encoded data stream to the LSTM layer for training.

3.3 *Feature engineering phase*

In this phase the model takes the encoded scaled data and performs feature or attribute engineering. This process starts by taking a window of preprocessed data stream and corresponding target or the dependent variable as input, the window is represented as a matrix where each column stands for an attribute and each row stand for a time step. The window is defined in the equation (9). The algorithm determines the attributes that strongly determine the target variables and removes the features that have a least on no effect on the target variable in the current window. 'TakeKbest' method is utilized to select the most relevant features alongside the target variable, the method is defined by equation (10). We call this process active feature selection and extraction process which enhances the model efficiency and adaptability. This process is archived by assigning weights to the individual attributes or features then dynamically adjust these weights based on their relevance to the target or the dependent variable. By a continuous comparison and update on the weights assigned to each feature, the model selects the most important feature of features influencing the prediction in that window. Using this approach the model can adapt to changing data distribution and hence handle the concept drift caused by the evolution of the underlying data distribution. As the data pattern change with time the importance of a feature in relation to the target variable might also change hence using this dynamic window selection and analysis allow the models to capture the important feature at a given time. If there is a significant difference between extracted features of two concurrent window periods, then this is an indication of a concept drift and a decision on whether to retrain the model or update the initial features need to made. The key factor influencing this decision would be the drift magnitude.

$$W = \{(X_t, y_t) | t \in (t_0, t_0 + n)\} \tag{9}$$

$$X' = \text{TakeKbest}\ (f, k) \tag{10}$$

Where f is function mapping x to y.

3.4 *Drift detection and Adaptation phase*

To detect and track concept drift, the framework uses the drift detection module to continually monitors two aspects (a) Cumulative distribution function CDF of the data points to observe concept drift in input data streams. This is archived by constantly comparing the stored features and newly extracted features using KS (Kolmogorov-Smirnov) as in the equation (11). (b) Accuracy of the predictions archived by tacking Area Under Curve (AUC) of the model for each give time window as illustrated by equation (12).

$$D_{ks} = max|F_t(x) - F_{t-1}(x)| \tag{11}$$

$$D_{acc} = max|\mathrm{A}_t(x,y) - \mathrm{A}_{t-1}(x,y)| \tag{12}$$

where A is the AUC at a given time window

A data drift is considered to happen when D_{ks} reaches a particular threshold which can be adjusted based on the prediction requirements. When this happens the model checks in its directory whether an associated model state exists to be used for forecast as LSTM stores the previously used model states. If a model state exists to be utilized by the drifted data stream, then the models switch to this model state otherwise the otherwise a new layer is trained based on the changes observed and merged with the most related model state. This is archived by freezing the already existing layers of the model which are not affected and training the layers with important features by adjusting the weights.

This step can be summarized by the below equation

Let $\mathrm{P}\left(\frac{\mathrm{y_i}}{\mathrm{x_i}}\right)$ be the original model's prediction of the dependent variable y given the input feature x and $\mathrm{P}\left(\frac{\mathrm{y_i}}{z_\mathrm{i}}\right)$ be the addition information to the original model's prediction of the dependent variable y given the new input feature z then a new model state represented by equation (13) below is created

$$mod_{ns} = \mathrm{P}\left(\frac{\mathrm{y_i}}{\mathrm{x_i}}\right) + \mathrm{P}\left(\frac{\mathrm{y_i}}{z_\mathrm{i}}\right) \tag{13}$$

This new model state is designed to improve the model's prediction performance when a concept drift that changes data characteristics and patterns occurs. By combining the previous knowledge from earlier models' state with new knowledge this drift adaptation is achieved.

A real concept drift is considered to happen when the accuracy parameter measure D_{acc} threshold is reached which occurs when the predictions AUC parameters go below a certain level. This can happen in both cases of a changed data distribution or changed expectation of the target value y even when the distribution of the features remains the same. In both these cases an incremental retraining on the entire model is performed to address this. During the incremental training, the main training cycle is paused to avoid conflicting training processes that might result in an unstable network (Chong et al., 2018; Udayakumar et al., 2023)

3.5 *Classification phase*

The core of MED-LSTM framework utilizes the stacking of Long Short Term Memory (LSTM) layers with each layer having different configuration. With this capability, temporal dependencies within the data can be captured by operating on 3D to generate output tensors that carry hidden state for each timestep from input tensors representing sequences. Then by determining the number of memory cells or hidden units within each layer, we can manage the complexity of the LSTM output space. we use hyperparameter known as "units to archive this parameterization

To capture relationships and the underlying pattens in the data, the MTED-LSTM model employs two key strategies:

i. Full Sequence of Hidden States Full Sequencing: MTED-LSTM returns the whole sequence of the hidden states for LSTM layers which allows the model to enrich its understanding of the temporal dynamics within the data stream. This approach is different from the traditional LSTM which only uses the final hidden state.

ii. ReLU Activation for Dense Layers: The use of ReLU (Rectified Linear Unit) activation introduces non-linearity in the learning enabling the model identify and learn more detailed and complex relationship between the dependent variables and the attributes of the features.

In addition, to maintain the stability of the model and efficient training the following callbacks are used:

i. Model Checkpoint: This callback ensures that the best performing model is saved and can be used for future predictions. This is done during the training phase where validation loss metric is used to determine the model efficiency.

ii. Early Exit: This callback prevents overfitting of the model by terminating the training once the validation loss metric does not improve after a number of epochs. The ensures that the model is not over tailored for a particular date set.

iii. Dynamic Learning Rate: This callback ensures that the model learning rate is adjustable throughout the training phase to ensure efficient convergence of the model when better approaches are found.

Another key feature of MTED-LSTM framework is the ability to utilize the LSTM transfer learning ability when a retrain is triggered. This is attained by freezing the pre-trained layers of the model preventing them from being retrained. This essentially allows the fine tuning of only the newly added layers for specific data pattens so the that the general data patterns are retained. This approach enables the pre-trained or pre-refined model map the general representation and pattens of the data even before the training is done. This will result to faster convergence of the model and efficient utilization of computation resources in case of a retrain in comparison with retraining the model afresh from scratch. This approach can also be referred to as incremental retrain.

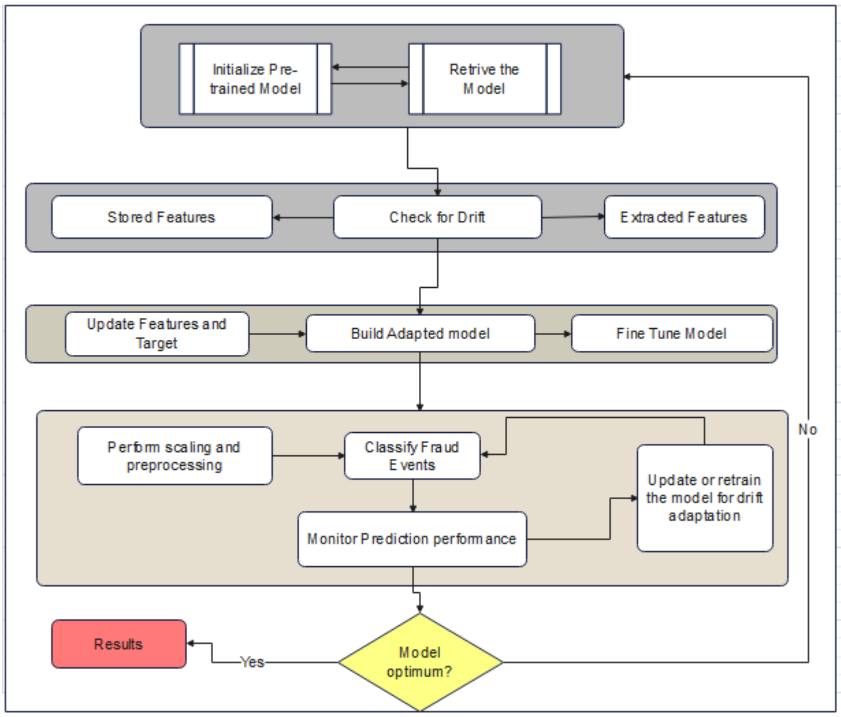

Initialize Pre-trained Model
Retrive the Model
Stored Features
Check for Drift
Extracted Features
Update Features and Target
Build Adapted model
Fine Tune Model
Perform scaling and preprocessing
Classify Fraud Events
Update or retrain the model for drift adaptation
Monitor Prediction performance
No
Results
Yes
Model optimum?

Figure 5. Classification Layer

In summary we can state that the model is designed to ingest data stream from the network then performs data encoding using the Markov transition to maintain the attributes relationships information. The relevant features with relation to the dependent variable are extracted from the pre-processed data stream. A predictive model is the build which adopts to both drift in data distribution and also change in the expectation of the target variable give the features. The performance of the model is enhanced by the transfer learning. The framework begins with encoding of the data stream then performs a feature engineering process that identifies the features that have a major impact on the target variable using the *TakeKbest* method to ensure that only the most impactful features are used for the downstream analysis.

To initiate the framework, a baseline model is used which is trained using the initially available dataset. A drift detection mechanism is later initiated to track and monitor both the data stream and the prediction accuracy aimed at detecting concept drift. The framework continually analyzes and evaluates the incoming stream of data against the older data steam to identify concept drift caused by data distribution changes. When a concept drift is detected two actions are triggered base on the source of the drift. If the drift is as a result of data distribution change, an adaptation process it triggered which involve fetching relevant data from the pre-trained models, revise the feature scales to maintain the distribution consistency, create and update model state aimed at predicting the new data stream characteristics and finally adjusting the model to accommodate the drift. If the drift is as a result of change in the expectation of the target variable, then an incremental model retrain is initiated. To maintain adaptability and high performance of the model, the framework utilizes transfer learning principles. By making use of knowledge and experience attained from previous models and data, the system can efficiently work with new data new data patterns and improve the learning process.

The framework continually evaluates the performance and efficiency of the model, monitoring signs of concept drift and then adjust the model accordingly. A dynamic update on the model in relation to new data and error calculations is performed with the goal of maintaining and improving the predictive accuracy. Once the model attains stability with the new data stream conditions, the it is used to perform prediction of the data stream.

We can have two training or learning modes: offline learning and online learning. In offline learning, the whole training data must be available at the time of model training. Only when training is completed can the model be used for prediction. In online mode, the model processes data sequentially, that is, it produces a model and puts it in operation without having the complete training data set available at the beginning. The model is then continuously updated during operation as more training data arrives (Metz et al., 2016). In production mode, the proposed model can be trained online as the model keeps track of its internal state, and the incremental retraining can be achieved.

4. Experimental evaluation, result and discussion

We define a DataStream as a dataset that contains a timestamp indicating the occurrence of the event, and it is processed in a sequential manner with regard to the timestamp (Geoffrey et al., 2016).

4.1 *Data description*

The dataset used in our study represents 6-month data streams from a Kenyan telco company and contains about 400,000 events or records. The data stream was generated by a process that randomly selected 65,000 events for each month (Jan 2023 to June 2023) from the complete dataset.

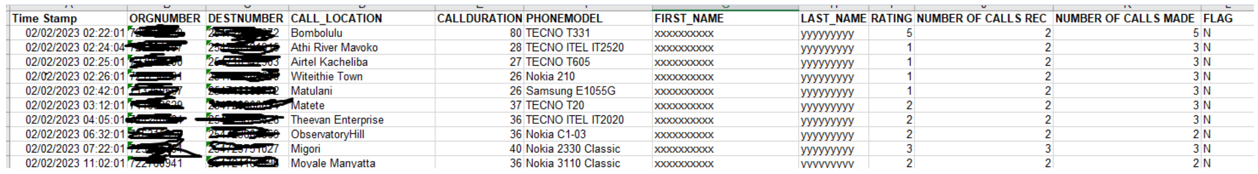
Table *1* below represents record counts of the sampled dataset from the Call Detail Record (CDR) generated by the Telco company that was used for this experiment

Table 1. Sample data summary

Data Stream	P (Fraudulent Events)	N(non-fraudulent)	Missing Data	Total Records
Jan_2023	1292	64679	78	66049
Feb_2023	1186	65235	3	66424
Mar_2023	2392	65098	47	67537
Apr_2023	5399	62356	2	67757
May_2023	1224	64389	36	65649
Jun_2023	467	65267	178	65912
Total	11960	387024	344	399328
% Total	2.995031653	96.91882362	0.086144723	100

The data set contains 10 attributes i.e. *Timestamp, Originating Number, Destination number, Call Location, Call duration, Phone model, First name, Last name, Rating, Number of calls received, Number of calls made.*

Figure 6 shows a part of the dataset.



Time Stamp	ORGNUMBER	DESTNUMBER	CALL_LOCATION	CALLDURATION	PHONEMODEL	FIRST_NAME	LAST_NAME	RATING	NUMBER OF CALLS REC	NUMBER OF CALLS MADE	FLAG
02/02/2023 02:22:01	[redacted]	[redacted]	Bombolulu	80	TECNO T331	xxxxxxxxxx	yyyyyyyyy	5	2	5	N
02/02/2023 02:24:04	[redacted]	[redacted]	Athi River Mavoko	28	TECNO ITEL IT2520	xxxxxxxxxx	yyyyyyyyy	1	2	3	N
02/02/2023 02:25:01	[redacted]	[redacted]	Airtel Kacheliba	27	TECNO T605	xxxxxxxxxx	yyyyyyyyy	1	2	3	N
02/02/2023 02:26:01	[redacted]	[redacted]	Witeithie Town	26	Nokia 210	xxxxxxxxxx	yyyyyyyyy	1	2	3	N
02/02/2023 02:42:01	[redacted]	[redacted]	Matulani	26	Samsung E1055G	xxxxxxxxxx	yyyyyyyyy	1	2	3	N
02/02/2023 03:12:01	[redacted]	[redacted]	Matete	37	TECNO T20	xxxxxxxxxx	yyyyyyyyy	2	2	3	N
02/02/2023 04:05:01	[redacted]	[redacted]	Theevan Enterprise	36	TECNO ITEL IT2020	xxxxxxxxxx	yyyyyyyyy	2	2	3	N
02/02/2023 06:32:01	[redacted]	[redacted]	ObservatoryHill	36	Nokia C1-03	xxxxxxxxxx	yyyyyyyyy	2	2	2	N
02/02/2023 07:22:01	[redacted]	[redacted]	Migori	40	Nokia 2330 Classic	xxxxxxxxxx	yyyyyyyyy	3	3	3	N
02/02/2023 11:02:01	[redacted]	[redacted]	Moyale Manyatta	36	Nokia 3110 Classic	xxxxxxxxxx	yyyyyyyyy	2	2	2	N

Figure 6. Sample dataset

4.2 *Evaluation metrics*

The following evaluation criteria were used, informed by the previous studies (Huiying et al.,2018; Geoffrey et al, 2016; Tung-Duong et al, 2021; Dennis et al,2018; Baena-Garcia et al., 2006; S. Yu et al., 2017): - confusion matrix (*Accuracy, Sensitivity/Recall*, *Precision, Specificity, F1 Score), Kolmogorov-Smirnov (K-S), AUC (Area Under the Curve), GINI, Combined Metric.*

4.3 *Experimental setup*

The experimentation was done utilizing the below libraries and platform.

4.3.1 *Platforms*

The models' development and the initial testing was carried out on a bare metal server running on Linux version 8.9 operating system with 4 CPU cores and 8GB of memory. The required Deep Learning software and libraries were installed on the machine. However, the actual training and testing of the models were carried out on AWS EC2 machine (m8g.medium) running on Gaviton CPU. The choice of this hardware was due to the high CPU and memory demands required to train and test the models.

2 programing languages were used in the experimentations, that is Python 3 and R Version 4.5. The reason for these programming languages choice it their stability in models' training and availability of the needed libraries. Some models were developed in R while others

were developed using Python depending on the availability of the respective libraries and ease of integration with other model components.

4.3.2 *Environment setup*

i. Python (Version 3.12.1)
ii. Anaconda 3 (Jupyter Notebook)
iii. Keras and TensorFlow for Deep learning

4.3.3 *Libraries setup*

i. Keras | Tensorflow
ii. Scikit-Learn for machine learning tasks
iii. Pandas and Numpy for data processing
iv. Matplotlib and Seaborn for visualization of results

4.3.4 *Experimentation procedure*

Below five steps approach was followed in the experimentation.

i. Exploratory Data Analysis
ii. Data preprocessing
iii. Models training
iv. Models testing
v. Results analysis

4.4 *Models testing*

Once the most optimal model (fine-tuned) was achieved, the models were tested using data streams for the subsequent months (Feb 2023 to June 2023) and results later analyzed. The aim of this was to: (a) Establish if concept drift occurs with time and what is its effect when it comes to the prediction or classification performance; (b) Establish how MTED-LSTM framework model performed in comparison with the other models with reference to concept drift.

4.4.1 *Experiment one: Before Concept Drift*

The first experiment was conducted by training all the models with Jan 2023 training partition data streams then testing them using the testing partition of the same dataset. The aim of the experiment was to evaluate performance of the models in the absence of drift.

4.4.2 *Experiment two: After Concept Drift*

The second experiment was conducted by passing the subsequent months datasets through the trained models then evaluate the performances of each model. The assumption in the test is that concept drift might have occurred through the months and might result to degradation of performance of the models, The cause of the concept drift might be as a results of data distribution change of the change in the expectation of the target variable. In both scenarios, the experiment was aimed at evaluating the effectiveness of the proposed MTED-LSTM framework in handling drift irrespective of its cause. The other models mentioned above were also tested using the same dataset and the results analyzed to determine their effectiveness of managing concept drift.

4.5 *Results analysis*

The summarized results for both experiment 1 and experiment 2 illustrates models' performance through the six-month data to determine the models' stability and the effect of the concept drift on the models.

The Area Under Curve (AUC) was the key metric used measure the effectiveness of the models in in the classification of fraudulent events. The AUC for the different models over the 6-month data is represented below

Random forest:

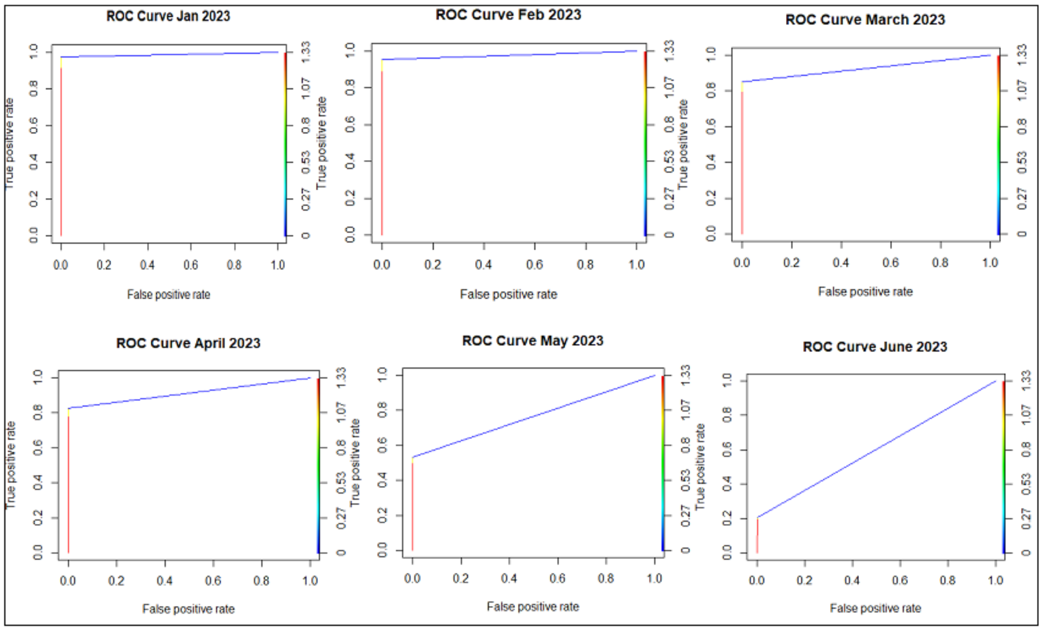


Figure 7. Random Forest AUC

Naïve Bias

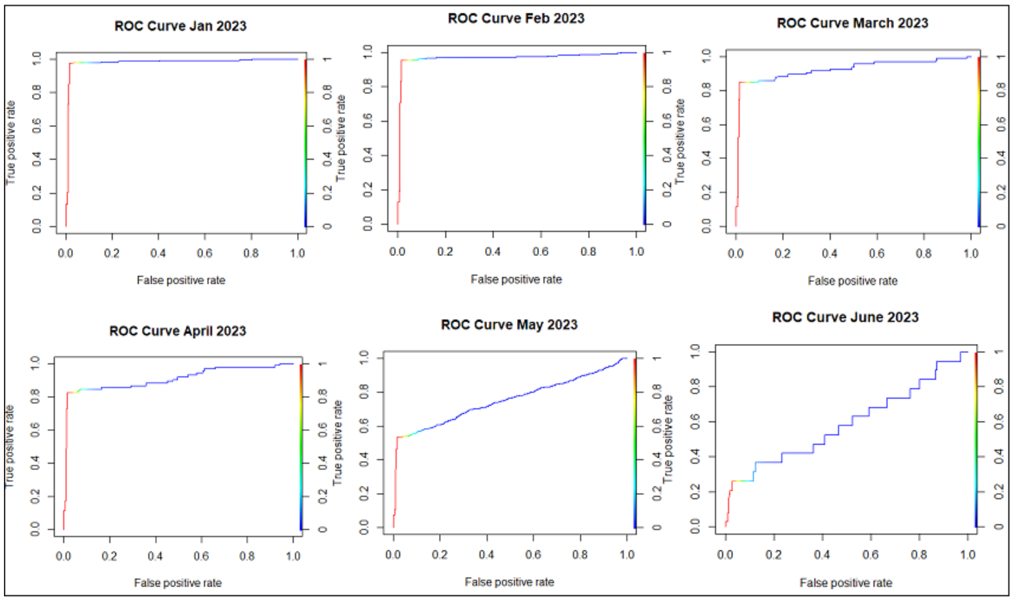


Figure 8. Naive Bias AUC

Support Vector Machine

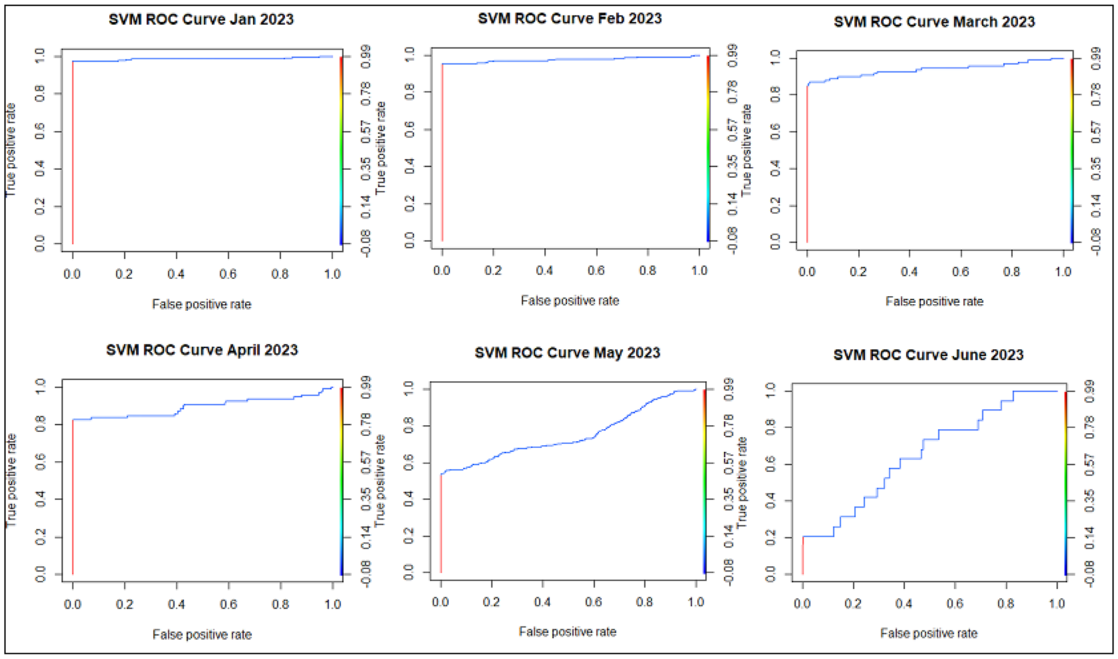


Figure 9. Support Vector AUC

Logistic Regression

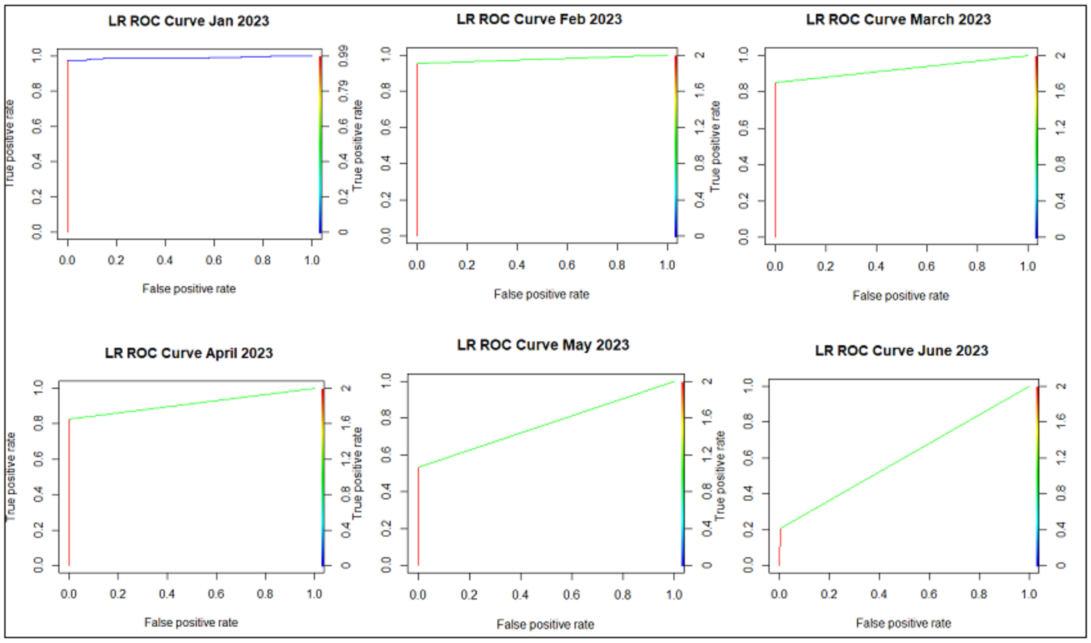


Figure 10. Logistic Regression AUC

MTED-LSTM

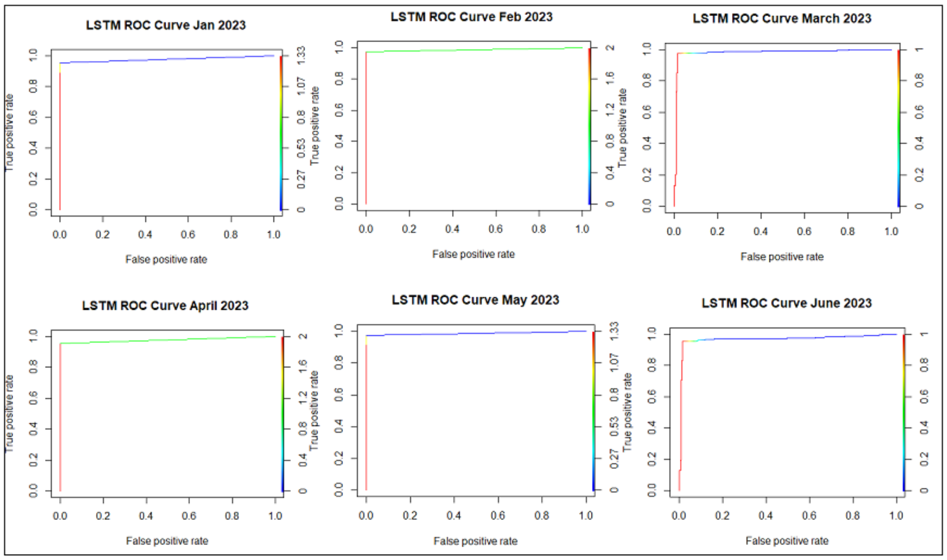


Figure 11. MTED-LSTM

The summarized results are represented in the tables, with each table representing each of the evaluation criteria

Accuracy:

Table 2. Accuracy Summary

Time	**Random Forest**	**Naïve Bias**	**SVM**	**Logistic Regression**	**MTED-LSTM**
Initial (EXP 1)	99.90844	96.26	100	100	99.9084
Jan-23	99.98321	95.4975	99.9832	99.98321	99.9634
Feb-23	99.96337	94.3018	99.9634	99.96337	99.9832
Mar-23	99.86571	89.0017	99.8657	99.86571	95.4975
Apr-23	99.84739	88.8812	99.8474	99.84739	99.9634
May-23	99.6078	73.6109	99.6078	99.6078	99.9832
Jun-23	99.30564	59.3911	99.3056	99.30564	94.3018

AUC:

Table 3. AUC Summary

Time	**Random Forest**	**Naïve Bias**	**SVM**	**Logistic Regression**	**MTED-LSTM**
Initial (EXP 1)	0.9883348	0.9900294	1	1	0.9883348
Jan-23	0.9879886	0.9801203	0.98609	0.987983	0.9760351
Feb-23	0.9760351	0.9667679	0.97648	0.9760326	0.987983
Mar-23	0.9247426	0.9232518	0.93871	0.9247422	0.9801203
Apr-23	0.9131171	0.9080081	0.89558	0.9131116	0.9760326
May-23	0.7666551	0.750207	0.74878	0.7666648	0.9879886
Jun-23	0.6021098	0.5858532	0.65392	0.6021106	0.9667679

K-S

Table 4. K-S Summary

Time	**Random Forest**	**Naïve Bias**	**SVM**	**Logistic Regression**	**MTED-LSTM**
Initial (EXP 1)	0.9757342	0.9848764	1	1	0.9757342
Jan-23	0.9759661	0.9608211	0.9759661	0.9759661	0.9520651
Feb-23	0.9520651	0.9369199	0.9520651	0.9520651	0.9759661
Mar-23	0.8494845	0.8343406	0.8622513	0.8494845	0.9608211
Apr-23	0.8262232	0.8110789	0.8262232	0.8262232	0.9520651
May-23	0.5333296	0.5181426	0.5379767	0.5333296	0.9759661
Jun-23	0.2042213	0.2421398	0.2609293	0.2042213	0.9369199

Gini:

Table 5. Gini Summary

Time	Random Forest	Naïve Bias	SVM	Logistic Regression	MTED-LSTM
Initial (EXP 1)	0.9766697	0.9800589	1	1	0.9766697
Jan-23	0.9759771	0.9602405	0.9721782	0.9759661	0.9520701
Feb-23	0.9520701	0.9335359	0.9529527	0.9520651	0.9759661
Mar-23	0.8494852	0.8465036	0.8774112	0.8494845	0.9602405
Apr-23	0.8262341	0.8160162	0.7911683	0.8262232	0.9520651
May-23	0.5333101	0.5004139	0.4975665	0.5333296	0.9759771
Jun-23	0.2042196	0.1717063	0.3078432	0.2042213	0.9335359

Recall:

Table 6. Recall Summary

Time	Random Forest	Naïve Bias	SVM	Logistic Regression	MTED-LSTM
Initial (EXP 1)	0.999437973	0.925199264	1	1	0.999437973
Jan-23	0.999938684	0.930498498	0.999938684	0.999938684	0.999846708
Feb-23	0.999846708	0.930405457	0.999846708	0.999846708	0.999938684
Mar-23	0.99931025	0.929860057	0.99931025	0.99931025	0.930498498
Apr-23	0.999233587	0.929873235	0.999233587	0.999233587	0.999846708
May-23	0.999431259	0.929845057	0.999431259	0.999431259	0.999938684
Jun-23	0.996474772	0.927257679	0.996474772	0.996474772	0.930405457

Precision:

Table 7. Precision Summary

Time	Random Forest	Naïve Bias	SVM	Logistic Regression	MTED-LSTM
Initial (EXP 1)	0.999642273	1	1	1	0.999642273
Jan-23	0.999892702	0.999901166	0.999892702	0.999892702	0.999785404
Feb-23	0.999785404	0.99978586	0.999785404	0.999785404	0.999892702
Mar-23	0.999340885	0.999291691	0.999340885	0.999340885	0.999901166
Apr-23	0.999233587	0.999275219	0.999233587	0.999233587	0.999785404
May-23	0.996627784	0.996441985	0.996627784	0.996627784	0.999892702
Jun-23	0.996551143	0.996540818	0.996551143	0.996551143	0.99978586

Specificity:

Table 8. Specificity Summary

Time	Random Forest	Naïve Bias	SVM	Logistic Regression	MTED-LSTM
Initial (EXP 1)	0.91954023	1	1	1	0.91954023
Jan-23	0.976027397	0.979452055	0.976027397	0.976027397	0.95221843
Feb-23	0.95221843	0.955631399	0.95221843	0.95221843	0.976027397
Mar-23	0.850174216	0.850174216	0.850174216	0.850174216	0.979452055
Apr-23	0.826989619	0.847750865	0.826989619	0.826989619	0.95221843
May-23	0.533898305	0.542372881	0.533898305	0.533898305	0.976027397
Jun-23	0.207746479	0.26056338	0.207746479	0.207746479	0.955631399

F1 Score

Table 9. F1 Score Summary

Time	Random Forest	Naïve Bias	SVM	Logistic Regression	MTED-LSTM
Initial (EXP 1)	0.999540112	0.961146497	1	1	0.999540112
Jan-23	0.999915693	0.963952233	0.999915693	0.999915693	0.999816055
Feb-23	0.999816055	0.963848725	0.999816055	0.999816055	0.999915693
Mar-23	0.999325567	0.963326426	0.999325567	0.999325567	0.963952233
Apr-23	0.999233587	0.963325843	0.999233587	0.999233587	0.999816055
May-23	0.998027553	0.961992303	0.998027553	0.998027553	0.999915693
Jun-23	0.996512956	0.960651677	0.996512956	0.996512956	0.963848725

Combined metric

Table 10. Combined Metric Summary

Time	Random Forest	Naïve Bias	SVM	Logistic Regression	MTED-LSTM
Initial (EXP 1)	0.987869704	0.978684099	1	1	0.987869704
23-Jan	0.987956798	0.968297878	0.98732393	0.987954931	0.975972085
23-Feb	0.975972085	0.955845508	0.97612039	0.975971252	0.987954931
23-Mar	0.924517556	0.906972942	0.93342896	0.924517422	0.968297878
23-Apr	0.912857962	0.894137614	0.90701226	0.912856129	0.975971252
23-May	0.766004084	0.743447301	0.76159475	0.766007318	0.987956798
23-Jun	0.600948019	0.596214892	0.63712075	0.600948285	0.955845508

The comparative plots for the models over the 6 months data are done with each plot representing the evaluation criteria.

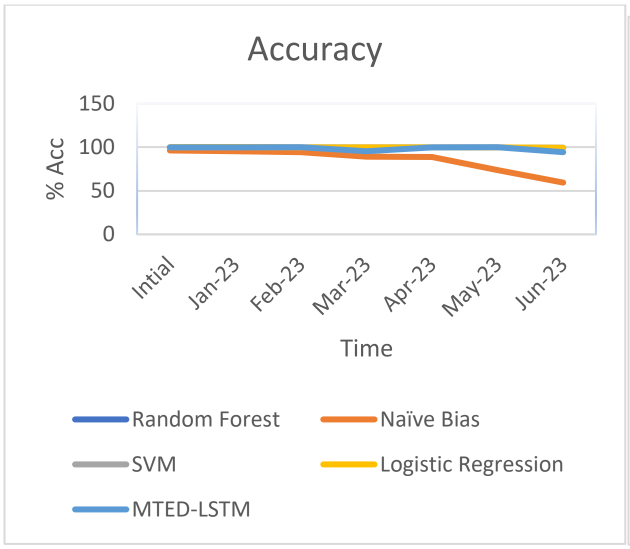


Figure 12. Accuracy Comparison

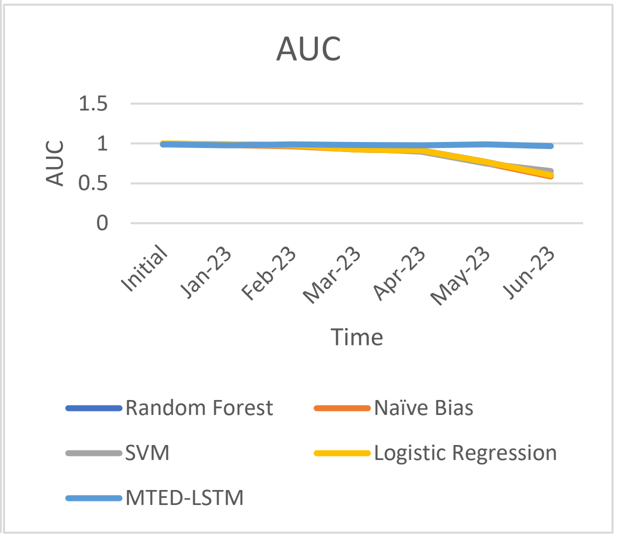


Figure 13. AUC Comparison

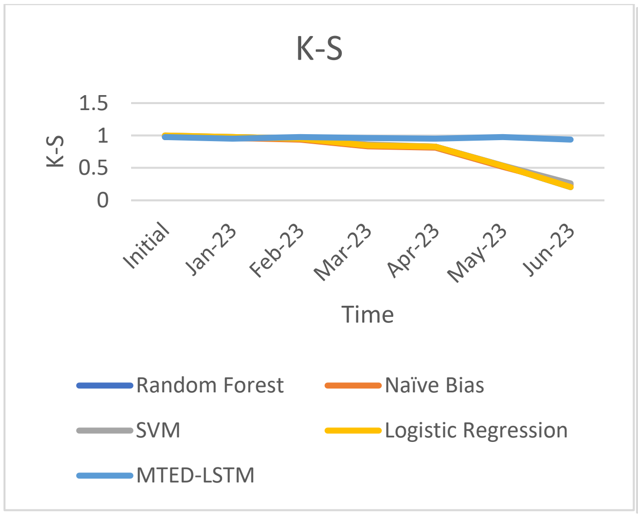


Figure 14. K-S Comparison

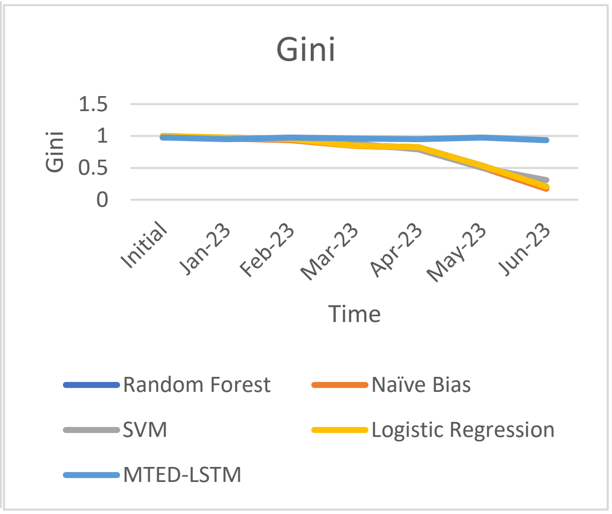


Figure 15. Gini Comparison

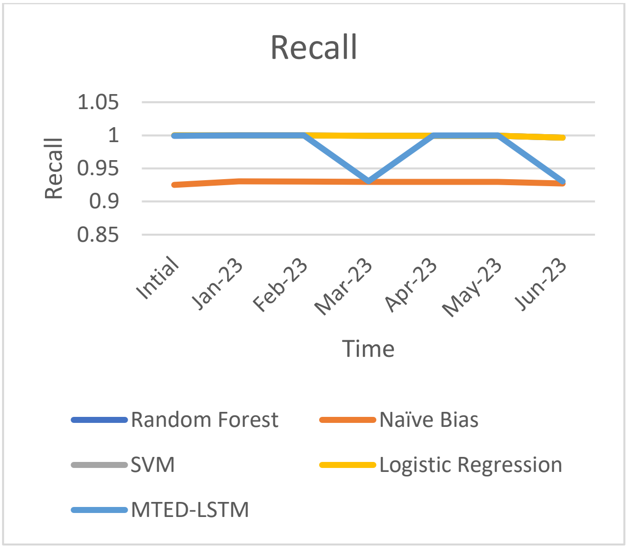


Figure 16. Recall Comparison

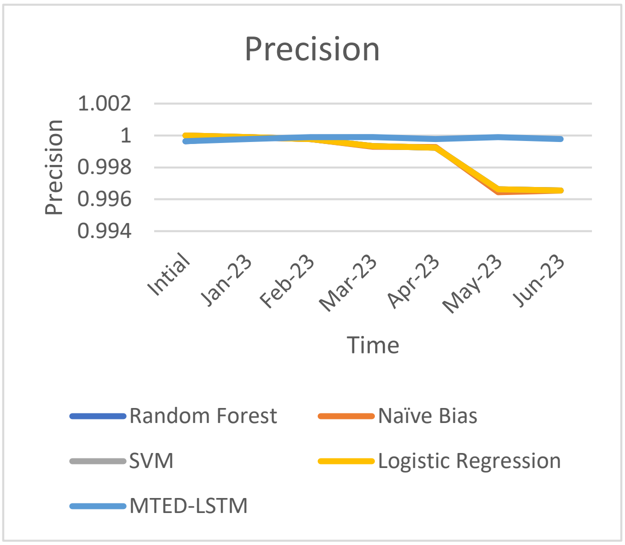


Figure 17. Precision Comparison

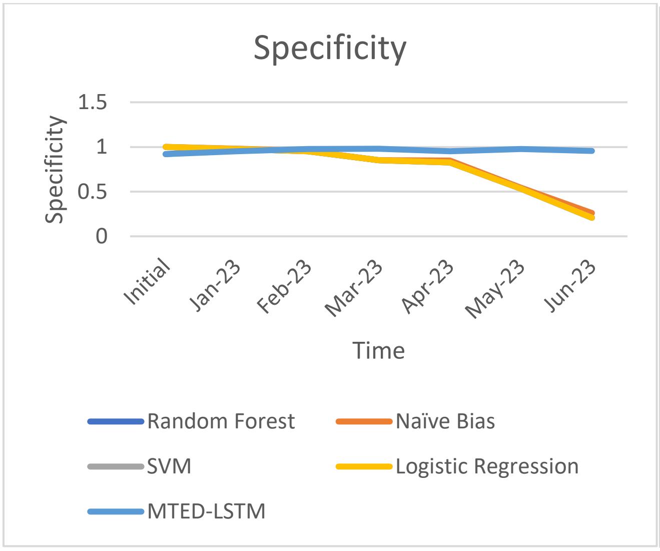


Figure 18. Specificity Comparison

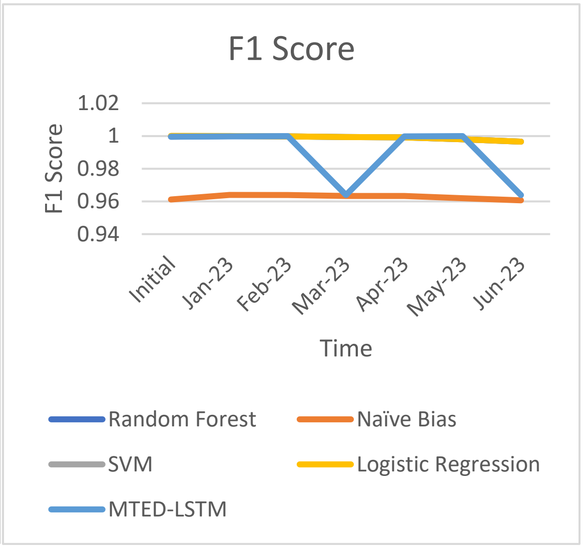


Figure 19. F1 Score Comparison

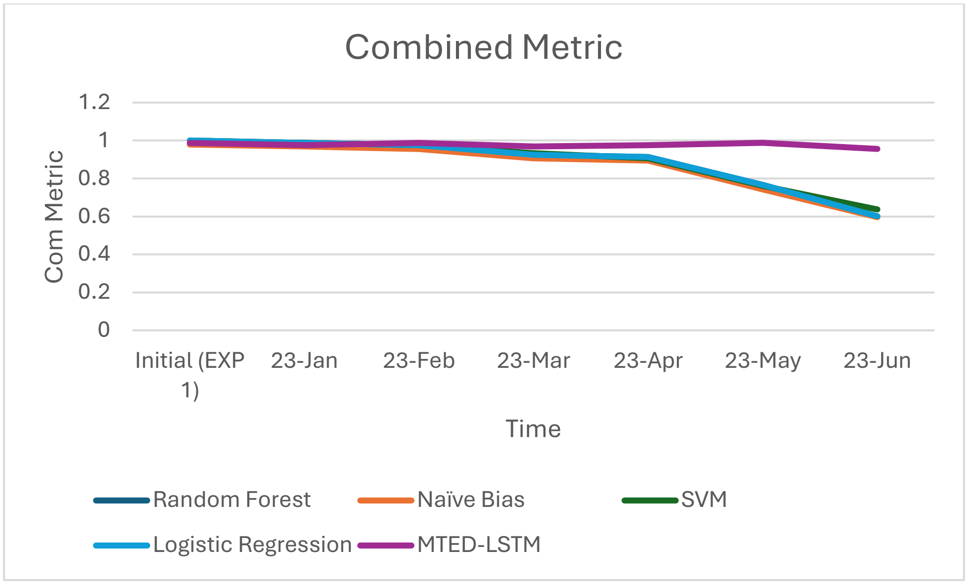


Figure 20. F1 Score Comparison

5. Discussion

From the results, the accuracy of the models over the six months slightly reduces with time except for the Naïve Bias model which reduces drastically after the 4th month to a low of 59% which can be attributed to the main drawback of Naïve Bias models that assumptions that all the independent variables are not related and from the data exploratory analysis, some of the variables are correlated to each other and this can explain the drastic reduction of accuracy with time. This characteristic in the mobile network dataset makes the Naïve Bias model not very appropriate in classification problems. Accuracy model evaluation criteria might not be the best criteria, as the data is very imbalanced, i.e., the % of non-fraudulent instances is 97.6%. If a model classifies all the instances as non-fraudulent, the accuracy would still be 97.6%, which might be translated as a good model, but it is not. and with this, other methods of evaluation should be employed to determine the effectiveness of the model.

Area Under Curve (AUC) which represents the probability that the model, if given a randomly chosen positive and negative example, will rank the positive higher than the negative would be the best criteria to evaluate the effectiveness of the model over time as the model's capacity to correctly classify the instances would be investigated. From the results AUC for the MTED-LSTM model over time remains constant, which indicates that the model's capability to classify the instances remains unchanged as compared to the other models. MTED-LSTM model attains an AUC value of 96.6% after 6 months, which is a slight drop from the initial AUC of 98.8% in the first month, it is worth highlighting that at the 4th month, the AUC improved as compared to the third month showing that the model slight had an improvement with time. All the other models registered huge drops in the AUC values with time and especially from the 4th month with Naïve Bias, Random Forest, Logistic Regression and finally Support Vector machine registering AUC values of 75%,76.6%,76.6% and 74.6% in the 5th month and 58.5%,60.2%,60.2% and 65.3% in that order in the 6th month. This means that the MTED-LSTM model a less affected by the changing data distribution or concept drift with time as compared to the other evaluated ML algorithms. It is worth noting that the initial values of AUC for Logistic Regression and Support Vector machines are 100%, but reduce rapidly with time as a result of drift.

Kolmogorov-Smirnov (K-S), which measures the level of separation between the distributions of the negative and positive responses, evaluated how good the model will be able to separate the instances with time. From the results, MTED-LSTM remained constant through the 6-month duration data with the lowest K-S value of 93.6% on the 6th month in comparison with 97.5% in the 1st month. The other models recorded a drastic degradation in the K-S value with time, with Random Forest and Logistic Regression models recording a low of 20.4% in the 6th month, Support Vector and Naïve Bias models recorded lows of 26% and 24% respectively in the 6th month. This result indicates the MTED-LSTM model remains stable and less susceptible to the effect of data distribution change of concept drift, as the model's capability to separate the instances remains nearly constant with time.

From the results, the Gini value, which measures the area between the 45-degree line and the cumulative response curves, is almost constant for the MTED-LSTM model in comparison with the rest of the models. The lowest Gini value for the LSTM model is 93.3% for the 6th month in comparison with an initial Gini value of 97.6%. The other models record a drastic decline of the Gini value with time, with Naïve Bias model recording a low value of 17.1% in the 6th month, with Random Forest, Logistic Regression, and Support Vector Machine recording a lowest Gini value of 20.4%,20.4%, and 30.7%, respectively. It is worth noting that the initial values of Gini for Logistic Regression and Support Vector machines are 100%, but reduce rapidly with time as a result of drift.

Specificity, which measures the portion of the sample that is classified as negative while using the classifier that is genuinely negative. This measure can also be used to measure the model's ability to reduce false negative classifications. From the result MTED-LSTM model indicates an almost constant value of specificity with time, with the lowest value being 95.5% as compared to the highest value of 97.2% in the 2nd month. It is noted that the initial specificity of the MTED-LSTM is 91.9% but increases with time, indicating that the model is less affected but the data distribution change of drift. The other models recorded a gradual reduction in specificity with time, with Logistic Regression, SVM, and Random Forest recording a lowest of 20.7% in the 6th month, while Naïve Bias recorded a low of 26% in the 6th month. Though the initial specificity value of Logistic Regression, SVM, and Naive Bias is 100%, this value reduces drastically as the data distribution changes. This means that with time, the other models, apart from MTED-LSTM, start recording more False negatives with time as they classify new instances.

Sensitivity or Recall, which measures the proportion of the samples that are classified as positive while using the classifier that are genuinely positive. From the results, Recall of the models does not degrade with time, as the lowest degradation happens on the Naïve Bias model, which recorded a low of 92.7% from a high of 93.1%. MTED-LSTM model also records a drop in the recall value to 93%, which is not gradual with time, as compared to the other performance measures. Recall, though, is not a very suitable measure of performance in this case due to the huge data imbalance. i.e., the % of the positive instances (fraudulent cases) is only 2.4%. This case applies for precision and F1 Score, for which the values across all the models remain constant or slightly decrease with time, i.e., the lowest F1 score recorded is 96% on Naïve Bias from an initial 96.1%. In the 6th month, the F1 score for the MTED-LSTM model was 96.3%, which is a drop from the initial F1 score of 99.9% in the first month, but this measure is not gradual with time as compared to the other performance measures.

Using a combined metric which takes consideration of all the metrics by averaging the AUC, F-score and K-S, MTED-LSTM model out performs all the other model over time indicating that the model adapts to concept drift over time hence resulting to better performance. The lowest combined metric for MTED-LSTM model is 95.5% in the 6th month. Compared to the other models, 63.7% for SVM, 60% for both Random Forest and Logistic Regression and 59.6% for Naïve bias.

The results also show the graphical representation of different performance metrics for the models as plotted against time, which visualizes the performance of each model over time.

It is evident that the models become less effective with time due to drift, but MTED-LSTM model, although they also degrade with time, it is more stable in comparison with the rest. With these performance measures, thresholds can be set on the models to trigger a retraining of the model. MTED-LSTM model can be incrementally trained as the previous state on the model is retained due to it memory retention capability, making the retrain fast and achieve model stability after a retrain (Hochreiter et al., 1997; Gers et al., 2000).

5.1 *Limitations*

The study only considered the dataset from one telecommunication service provider, which might not be a full representation of the entire population.

The sample dataset selection was not controlled by the researcher, as the dataset used in this study was randomly selected and provided by the service provider, which means we cannot provide the sample selection criteria, bias management, and the methodology used to extract the sample dataset from the entire population/dataset.

5. Conclusion

The objective of the study was to analyze the effects of concept drift with respect to the fraud detection efficiency over time in the mobile network. The other objective was to develop and evaluate an approach that minimizes the effect of concept drift. In the study, six months of data from a mobile network service provider were considered with the aim of evaluating how different models performed in classifying fraudulent mobile network cases over the six months' time. A total of 5 models were analyzed for the effect of concept drift in the classification of fraud cases in the mobile network ecosystem. The selection of the models was informed by the literature, which identified Random Forest, Naïve Bias, Support Vector Machine, and logistic regression as the most commonly used approaches when it comes to classifying and detecting fraud cases. In the study, a dynamic model based on the LSTM approach and using Markov Transition while encoding the data to preserve the inter-relationships information in the data stream was developed. This was informed by the EDA, which revealed that the variables in the dataset were. The architecture of the MTED-LSTM and its components have been well discussed, as well as the proposed data flow approach. A total of 8 model performance indicators were applied to evaluate the performance of the different modes with time to understand the effects of concept drift with time. The selection of the performance evaluators was also informed by literature on the common model performance evaluation criteria. These evaluation criteria are Area Under Curve (AUC), Kolmogorov-Smirnov (K-S), Gini, Accuracy, Specificity, Sensitivity or Recall, Precision, and F1-score.

From the results, the question of when the drift occurs and its effect is evidenced by the reduction of multiple performance indicators of different models with time. When using Naïve Bias model, the accuracy drops from 96.26% to 59.4% with is 6-month duration, although the other models' accuracy did not significantly change during the same duration. The main reason for the poor performance of the Naïve Bias model is its assumption that the variables as totally independent, which is not the case with the mobile network data streams as per the EDA. Though accuracy performance evaluator is one of the most used evaluation criteria, it does not clearly reflect the real performance of the models due to the massive imbalance of the dataset, whereby the fraud cases only account for 2.4% of the dataset. Area under Curve (AUC) which evaluates how effective is a model in defining the decision line indicated that all the models reduced gradually in this front with Random Forest model reducing from 98.8% to 60.2%, Naïve Bias reduced from 99% to 58.5%, SVM reduced from 100% to 65.3%, Logistic Regression reduced from 100% to 60% and MTED-LSTM model reduced from 98.8% to 96.6% within a 6-month time frame. This is a clear indication of concept drift, but the MTED-LSTM model appeared to effectively manage the drift as its ability to define the decision boundary remained high in comparison to the other models. Gini evaluation criteria also indicate the same picture, with the MTED-LSTM model being least affected by the drift.

While using Kolmogorov-Smirnov (K-S) performance evaluator all the models gradually dropped in this measure with Random Forest dropping from 97.5% to 20.4%, Naïve Bias dropping from 98.4% to 24.1%, SVM dropping from 100% to 26%, Logistic Regression dropping from 100% to 20.4% and MTED-LSTM model dropping from 97.5% to 93.6%. This means that the ability to separate instances with time reduces, highlighting the effect of concept drift. Using this evaluation criterion, the MTED-LSTM model seems to be the least affected by the drift, with, in some cases, showing improvement with time.

When the performance of the models is measured using a combined metric which put into consideration all the other metrics, MTED- LSTM model out performs all the other model over time indicating that the model adapts to concept drift over time hence resulting to better performance. The lowest combined metric for MTED- LSTM model is 95.5% in the 6th month.

In conclusion MTED-LSTM model effectively manages the effects of concept drift due to its ability to store the previous memory model state on the time series DataStream. This is evidenced by the results, where all the performance indicators show the model is least affected by the drift with time. It is worth noting that between the 4th and 5th months, all the other models drastically reduce their effectiveness, but the MTED-LSTM model actually improved during this time frame, which can be attributed to its ability to store the memory state of the previous time series streams, resulting in a stable and smooth performance.

6.1 *Future work*

The study didn't answer the other concept drift analysis questions, which are: How is the drift? And where did the drift occur? In the mobile network concept drift analysis. Hence, a future study can be conducted to understand what causes the drift, where it occurs, and its magnitude. Answering those questions will allow us to have a full circle on the analysis of the concept drift as it relates to the mobile network fraud detection.

Acknowledgements

This research did not receive any specific grant from funding agencies in the public commercial, or not-for-profit sectors.

The author declares no competing interests.

References

Abdul et al., 2024, *Transfer-learning enabled adaptive framework for load forecasting under concept-drift challenges in smart-grids across different-generation-modalities*. Available at: https://www.sciencedirect.com/science/article/pii/S2352484724006176.

Adele et al., 2011, *Random Forests*. Available at: https://link.springer.com/chapter/10.1007/978-1-4419-9326-7_5.

Ajay et al., 2019, *Review of Deep Learning Algorithms and Architectures*. Available at: https://ieeexplore.ieee.org/abstract/document/8694781/.

Ajith Abraham, 2005, *Artificial Neural Networks*. Available at: http://wsc10.softcomputing.net/ann_chapter.pdf.

Alberto et al., 2018, *SMOTE for Learning from Imbalanced Data: Progress and Challenges, Marking the 15-year Anniversary*. Available at: https://www.jair.org/index.php/jair/article/view/11192.

Angshuman et al., 2017, *Improved Random Forest for Classification*. Available at: https://ieeexplore.ieee.org/abstract/document/8357563/.

Apapan et al., 2018, *Credit Card Fraud Detection using Deep Learning based on Auto-Encoder and Restricted Boltzmann Machine*. Available at: https://thesai.org/Downloads/Volume9No1/Paper_3-Credit_Card_Fraud_Detection_Using_Deep_Learning.pdf.

Baena-Garcia et al,2006, *Early drift detection method*. Available at: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=2747577a61c70bc3874380130615e15aff76339e.

Basseville et al., 1994, *Early warning of slight changes in systems*. Available at: https://www.sciencedirect.com/science/article/pii/0005109894902313.

Biju, 2022, *SMS Fraud Detection Using Machine Learning*. Available at: https://www.researchgate.net/profile/Soumya-Prusty/publication/360371905_SMS_Fraud_Detection_Using_Machine_Learning/links/628781c5cd5c1b0b34e95817/SMS-Fraud-Detection-Using-Machine-Learning.pdf.

C. Alippi et al., 2008. *An effective just-in-time adaptive classifier for gradual concept drifts*. Available at: https://ieeexplore.ieee.org/abstract/document/6033426/.

Campanharo et al. 2011, *Duality between Time Series and Networks*. Available at: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0023378.

Chong et al., 2018, *Classification algorithmof Chinese sentimentorientation based on dictionary and LSTM*. Available at: https://dl.acm.org/doi/abs/10.1145/3291801.3291835.

Delany et al., 2005, *A Comparison of Ensemble and Case-Base Maintenance Techniques for Handling Concept Drift in Spam Filtering*. Available at: https://cdn.aaai.org/FLAIRS/2006/Flairs06-067.pdf.

Dennis et al., 2018, *Adapting to Concept Drift in Credit Card Transaction Data Streams Using Contextual Bandits and Decision Trees*. Available at: https://ojs.aaai.org/index.php/AAAI/article/view/11411.

Distributions. Available at: https://epubs.siam.org/doi/abs/10.1137/1.9781611972771.1

Dries et al., 2009. *Adaptive concept drift detection*. Available at: https://onlinelibrary.wiley.com/doi/abs/10.1002/sam.10054.

Duda et al., 200, *Book review: " Pattern classification"*. Available at: http://www.worldscinet.com/google/S1469026801000251.pdf.

Evgeny et al., 2018, *Modern approaches to prevent fraud in mobile communications networks*, Available at: https://ieeexplore.ieee.org/abstract/document/8317111.

Fidelis et al., 2024, *Enhancing the Random Forest Model via Synthetic Minority Oversampling Technique for Credit-Card Fraud Detection*. Available at: https://www.researchgate.net/profile/Margaret-Okpor/publication/379324623_Enhancing_the_Random_Forest_Model_via_Synthetic_Minority_Oversampling_Technique_for_Credit-Card_Fraud_Detection/links/6630201b7091b94e93e7086b/Enhancing-the-Random-Forest-Model-via-Synthetic-Minority-Oversampling-Technique-for-Credit-Card-Fraud-Detection.pdf.

Firas et al., 2023, *DA-LSTM: A dynamic drift-adaptive learning framework for interval load forecasting with LSTM networks*. Available at: https://www.sciencedirect.com/science/article/pii/S0952197623006644.

Gallego et al., 2017, *A survey on data preprocessing for data stream mining: Current status and future directions*. Available at: https://www.sciencedirect.com/science/article/pii/S0925231217302631.

Gama et al., 2013, *A Survey on Concept Drift Adaptation*. Available at: https://dl.acm.org/doi/abs/10.1145/2523813.

Gama et al., 2004, *Concept Drift in Decision Trees Learning from Data Streams*. Available at: https://www.researchgate.net/profile/Joao-Gama/publication/236007671_Concept_Drift_in_Decision-Tree_Learning_from_Data_Streams/links/00b7d53959ecb5e3e2000000/Concept-Drift-in-Decision-Tree-Learning-from-Data-Streams.pdf.

Gao et al., 2007, *A General Framework for Mining Concept-Drifting Data Streams with Skewed* Enzo et al,2007, *Introduction to artificial neural networks*. Available at: https://journals.lww.com/eurojgh/FullText/2007/12000/Introduction_to_artificial_neural_networks.5.aspx.

Geoffrey et al., 2016, *Characterizing Concept Drift*. Available at: https://arxiv.org/pdf/1511.03816

Georgios et al., 2019, *Augment to Prevent: Short-Text Data Augmentation in Deep Learning for Hate-Speech Classification*. Available at: https://dl.acm.org/doi/abs/10.1145/3357384.3358040.

Gerard et al., 2015, *Random Forest Guided Tour*. Available at: https://link.springer.com/article/10.1007/s11749-016-0481-7.

Gers, et al, 2000, *Learning to forget: Continual prediction with LSTM*. Available at: https://ieeexplore.ieee.org/abstract/document/6789445/.

Hochreiter et al., 1997, *Long short-term memory*. Available at: https://ieeexplore.ieee.org/abstract/document/6795963/.

Hochreiter et al., 2001, *Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. A Field Guide to Dynamical Recurrent Neural Networks*. Available at: https://ieeexplore.ieee.org/document/5264952.

H. Raza et al., 2015, *Adaptive learning with covariate shift-detection for motor imagery-based brain–computer interface*. Available at: https://link.springer.com/article/10.1007/s00500-015-1937-5.

Huiying et al.,2018, *Adaptive Fraud Detection System Using Dynamic Risk Features*. Available at: https://arxiv.org/abs/1810.04654.

JOAO et al., 2014, *A Survey on Concept Drift Adaptation*. Available at: https://dl.acm.org/doi/abs/10.1145/2523813.

Jie Lu et al, 2020, *Data-driven decision support under concept drift in streamed big data*. Available at: https://link.springer.com/article/10.1007/s40747-019-00124-4.

Judi, 2002, *Dealing with Missing Data*. Available at: https://mro.massey.ac.nz/handle/10179/4355.

J. Shao et al., 2014, *Prototype-based learning on concept-drifting data streams*. Available at: https://dl.acm.org/doi/abs/10.1145/2623330.2623609.

Kala, 2021, *Assessment of SIM Box Fraud: An Approach to National Security Threat*. Available at: https://www.researchgate.net/profile/Kala-Baskar/publication/356914651_Assessment_of_SIMBox_Fraud_An_Approach_to_National_Security_Threat/links/61b71797a6251b553ab51ccc/Assessment-of-SIMBox-Fraud-An-Approach-to-National-Security-Threat.pdf.

Kelly et al., 1999, *The impact of changing populations on classifier performance*. Available at: https://dl.acm.org/doi/pdf/10.1145/312129.312285.

Kenji, 2011, *Artificial neural networks: methodological advances and biomedical applications*. Available at: https://books.google.com/books?hl=en&lr=&id=JuaODwAAQBAJ&oi=fnd&pg=PR11&dq=Kenji,+2011+ann&ots=XWjQ_S9KXd&sig=Ph8jFONuvXvX08yM8OzSZpi-v8.

K.-I. Kim et al., 2015 Toll Fraud Detection of VoIP Service Networks in Ubiquitous Computing Environments, Int. J. *Distrib. Sens. Networks*, vol. 2015, 2015. Available at: https://journals.sagepub.com/doi/full/10.1155/2015/276408.

Lakshmi et al., 2018, *Machine Learning For Credit Card Fraud Detection System*. Available at: https://www.academia.edu/download/106080716/ijaerv13n24_18.pdf.

Liu et al, 2017, *Regional Concept Drift Detection and Density Synchronized Drift Adaptation*. Available at: https://opus.lib.uts.edu.au/handle/10453/126374.

Mais et al., 2018, *Detection of Wangiri Telecommunication Fraud Using Ensemble Learning*. Available at: https://www.researchgate.net/profile/George-Sammour/publication/333232206_Detection_of_Wangiri_Telecommunication_Fraud_Using_Ensemble_Learning/links/5d07d7f5a6fdcc35c155c208/Detection-of-Wangiri-Telecommunication-Fraud-Using-Ensemble-Learning.pdf.

Metz et al., 2016, *Discrete sequential prediction of continuous actions for deep rl*. Available at: https://arxiv.org/abs/1705.05035.

Mukta et al., 2009, *Neural networks and statistical techniques: A review of applications*. Available at: https://www.sciencedirect.com/science/article/abs/pii/S0957417407004952

Mundia et al., 2024, *Assessing Mobile Network Fraud Threats and Prevention Strategies in Kenya*. Available at: https://www.journals.eanso.org/index.php/eajit/article/view/2212.

Nassir, 2020, *Study to use NEO4J to analysis and detection SIM-BOX fraud*. Available at: https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=Study+to+use+NEO4J+to+analysis+and+detection+SIM-BOX+fraud&btnG=.

Nitesh et al., 2002, *SMOTE: synthetic minority over-sampling technique*. Available at: https://cir.nii.ac.jp/crid/1370017282217666450.

Niveditha et al., 2019, *Credit Card Fraud Detection Using Random Forest Algorithm*. Available at: https://www.academia.edu/download/65294900/CSEIT195261.pdf.

N. Lu et al., 2014, *Concept drift detection via competence models*. Available at: https://www.sciencedirect.com/science/article/pii/S0004370214000034.

N. Lu et al., 2016, *An online competence-based concept drift detection algorithm*. Available at: https://link.springer.com/chapter/10.1007/978-3-319-50127-7_36.

Ogwueleka, 2009, *fraud detection in mobile communications networks using user profiling and classification techniques*. Available at: https://core.ac.uk/download/pdf/37370566.pdf.

Paliwal, M., & Kumar, U. A., 2009. *Neural Networks and Statistical Techniques: A Review of Applications. Expert System with Application*. 36: 2–17. Available at: https://www.sciencedirect.com/science/article/pii/S0957417407004952.

Philip et al., 2015, *On arbitration, arbitrage and arbitrariness in financial markets and their governance: unpacking LIBOR and the LIBOR scandal*. Available at: https://www.researchgate.net/profile/Philip-Ashton-2/publication/275556489_On_Arbitration_Arbitrage_and_Arbitrariness_in_Financial_Markets_and_Their_Governance_Unpacking_LIBOR_and_the_LIBOR_Scandal/links/5b10028faca2723d99775354/On-Arbitration-Arbitrage-and-Arbitrariness-in-Financial-Markets-and-Their-Governance-Unpacking-LIBOR-and-the-LIBOR-Scandal.pdf.

P. Saravanan et al., 2014 Data mining approach for subscription-fraud detection in telecommunication sector, *Contemp. Eng.Sci.*, vol. 7, no. 9–12, pp. 515–522, 2014. Available at: http://m-hikari.com/ces/ces2014/ces9-12-2014/subramaniyaswamyCES9-12-2014.pdf.

Raghavendra et al., 2011, *Credit Card Fraud Detection Using Neural Network,* Available at: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=0419c275f05841d87ab9a4c9767a4f997b61a50e.

Roger, 2023, *SIM Swapping Attacks for Digital Identity Theft: A threat to financial services and beyond*. Available at: https://www.researchgate.net/profile/Roger-Hallman/publication/376612643_SIM_Swapping_Attacks_for_Digital_Identity_Theft_A_threat_to_financial_services_and_beyond/links/658091b20bb2c7472bf3dd84/SIM-Swapping-Attacks-for-Digital-Identity-Theft-A-threat-to-financial-services-and-beyond.pdf.

Rok et al., 2013, *SMOTE for high-dimensional class-imbalanced data*. Available at: https://link.springer.com/article/10.1186/1471-2105-14-106.

Roselina et al., 2016, *Potential ANN prediction model for multiperformances WEDM on Inconel 718*. Available at: https://link.springer.com/article/10.1007/s00521-016-2796-4.

R. Sallehuddin et al, 2015. Detecting SIM box fraud by using support vector machine and artificial neural network, J. *Teknol.*, vol. 74, no. 1, pp. 137–149, 2015. Available at: https://journals.utm.my/jurnalteknologi/article/view/2649.

Ruinan et at., 2018, *Sequential Behavioral Data Processing Using Deep Learning and the Markov Transition Field in Online Fraud Detection*. Available at: https://arxiv.org/abs/1808.05329.

Salganicoff, 1997, *Tolerating Concept and Sampling Shift in Lazy Learning Using Prediction Error Context Switching*. Available at: https://link.springer.com/chapter/10.1007/978-94-017-2053-3_5.

Samira et al., 2018, *A Survey on Deep Learning: Algorithms, Techniques, and Applications*. Available at: https://dl.acm.org/doi/abs/10.1145/3234150.

Sheo et al.,2022, *Credit Card Fraud Detection Using Support Vector Machine*, Available at: https://link.springer.com/chapter/10.1007/978-981-16-6407-6_3.

S. Xu et al., 2017, *Concept drift learning with alternating learners*. Avalable at: https://ieeexplore.ieee.org/abstract/document/7966109/.

Snehal et al., 2019, *Awareness of Sim Swap Attack*. Available at: https://d1wqtxts1xzle7.cloudfront.net/59921743/215_Awareness_of_Sim_Swap_Attack20190703-77135-1etachf-libre.pdf?1562159287=&response-content-disposition=inline%3B+filename%3DAwareness_of_Sim_Swap_Attack.pdf&Expires=1718043928&Signature=MK9zvJb6Gps287s3jEyjpkvO9~9bEfb~dwP4l3Co2O90qWO96DrTn7wzCH29Ua3Bm~1tOmFAxlD9IdYAgkXsfaXIgMW40kfjaR1dxmelM7qAVaWuU8VjCxgp2dOGUmc5K43DmebUySV~6s-29e6NQOmI5deRql02yvQzRlMnanNZ-x5p7KR7ldVxQ0cidc6XhUCGCxtzVO8x~aie3Wgt0a2QCLyqi~40CG1c8Tr7mlRFRb2m00bB1lGWqXxTM0qjuJschVcxs-gzD2qydq-6jb1cuCwqux6~LSABs9LtazrW1UkQ3EuPbq1eXy7kw7xonn2Q1Rd4zkGcUx-bllV6Gg__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA.

S. Priya1et al., 2023, *Deep learning framework for handling concept drift and class imbalanced complex decision-making on streaming data*. Available at: https://link.springer.com/article/10.1007/s40747-021-00456-0.

Steven et al.,2018, *Artificial Neural Networks*. Available at: https://www.igi-global.com/chapter/artificial-neural-networks/183727.

Suryani et al., 2022, *Fraud Triangle Perspective: Artificial Neural Network Used in Fraud Analysis*. Available at: https://search.proquest.com/openview/fb2104c325ba267b9ca22ab48cb7589b/1?pqorigsite=gscholar&cbl=1046413&casa_token=sgH0Ka6YncsAAAAA:iWgnQ9x6rTbZFsbi5YDMykuJ4M1M2x_3sCg0qG_u-Z_Iw4rnq1slaWFMXi2LB4F-sSQFDgkjUt4.

S. Yu, et al., 2017, *Concept drift detection with hierarchical hypothesis testing*. Available at: https://epubs.siam.org/doi/abs/10.1137/1.9781611974973.86.

Tajwar et al., 2024, *LSTMDD: an optimized LSTM-based drift detector for concept drift in dynamic cloud computing*. Available at: https://peerj.com/articles/cs-1827/.

T.J. Watson Research Center, 2001, *An empirical study of the naive Bayes classifier*. Available at: https://www.cc.gatech.edu/home/isbell/classes/reading/papers/Rish.pdf.

Tsymbal, 2004, *The problem of concept drift: definitions and related work*. Available at: https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=30eac73e9b482bc28b5b68cd585557de48d0618f.

Tung-Duong et al., 2021, *Customs Fraud Detection in the Presence of Concept Drift*. Available at: https://ieeexplore.ieee.org/abstract/document/9679911/.

Udayakumar et al., 2023, *Integrated SVM-FFNN for Fraud Detection in Banking Financial Transactions*. Available at: https://www.researchgate.net/profile/Pbharath-Kumar-Chowdary/publication/376179858_Integrated_SVM-FFNN_for_Fraud_Detection_in_Banking_Financial_Transactions/links/657365f26610947889aec12a/Integrated-SVM-FFNN-for-Fraud-Detection-in-Banking-Financial-Transactions.pdf.

Vapnik V. 1995. *The Nature of Statistical Learning*. Available at: https://cir.nii.ac.jp/crid/1370846644385113856.

W. Heng et al., 2015, *Concept Drift Detection for Streaming Data*. Available at: https://ieeexplore.ieee.org/abstract/document/7280398/.

Widmer, 1993, *Learning in the presence of concept drift and hidden contexts*. Available at: https://link.springer.com/article/10.1023/A:1018046501280

Xizhao, et al., 2020, *Recent advances in deep learning*. Available at: https://link.springer.com/article/10.1007/s13042-020-01096-5.

XIN et al., 2024, *Markov transition fields and auto-encoder-based preprocessing for event recognition of Φ-OTDR*. Available at: https://yadda.icm.edu.pl/baztech/element/bwmeta1.element.baztech-5c079b94-fce1-42c9-a840-612843b9a04f.

X. Song et al., 2007, *Learning under concept drift: an overview*. Available at: https://arxiv.org/abs/1010.4784.

Yang et al., 2018, *Mining Fraudsters and Fraudulent Strategies in Large-Scale Mobile Social Networks*, Available at: https://ieeexplore.ieee.org/document/8744319.

Yoshua, 2015, *Deep Learning and its application to CV and NLP*, Available at: http://udrc.eng.ed.ac.uk/sites/udrc.eng.ed.ac.uk/files/attachments/Deep%20Learning%20and%20its%20application%20to%20CV%20and%20NLP_0.pdf.

Y. Zhang et al., 2017, *Three-layer Concept Drifting Detection in Text Data Streams*. Available at: https://www.sciencedirect.com/science/article/pii/S0925231217307981.

Zhaohui et al., 2022, *Data Replay Method for Detecting Fraud Concept Drift in Online Transactions*. Available at: https://www.researchsquare.com/article/rs-1404082/latest.