Jurnal Ilmiah Teknik Elektro Komputer dan Informatika (JITEKI)
Vol. 10, No. 4, December 2024, pp. 1021–1036
ISSN: 2338-3062, DOI: 10.26555/jiteki.v10i4.30013

Tailoring Data Storage Configuration for Efficient Fraud Detection Model Training

Abdusy Syarif, Muhammad Haikal Satria, Hanene Gabteni
Universitas Nasional, Indonesia
ASTEK, France

ARTICLE INFO

Article history:

Received September 02, 2024
Revised December 11, 2024
Accepted February 13, 2025

Keywords:

Machine Learning;
Fraud Detection;
Model Prediction;
Fine Tuning;
Model Update;
Real-Time Detection

ABSTRACT

The rapid growth of e-commerce in Indonesia, with a record 88.1% growth rate, has been accompanied by a surge in online fraud, leading to an estimated loss of 4.62 trillion rupiahs. Current fraud prevention methods, such as the widely used 3D-Secure system, though effective, result in a high rate of transaction abandonment (approximately 16%), which is undesirable for merchants. To address this, we propose an AI-based fraud detection system that leverages machine learning models to identify potentially fraudulent transactions. By employing a combination of classification algorithms, including logistic regression and neural networks, security protocols are activated only for high-risk transactions, optimizing transaction processing efficiency and improving detection accuracy. Our study focuses on fine-tuning key parameters of the AI-Fraud Detector model, specifically some parameters such as Δt_{train}, Δt_{lag} and $frac\_hr\_pass$, to enhance detection performance over time. Simulation performances using ROC-AUC, false positive rate (fpr), and true positive rate (tpr) metrics show that a configuration with a training period (Δt_{train}) of 180 days, a lag period (Δt_{lag}) of 90 days, and a high-risk pass fraction ($frac\_hr\_pass$) of 10% yields a balance between detection efficiency ($\sim 50\%$) and a reduced false positive rate. It means that the model is able to identify approximately 50% of the actual high-risk events while minimizing the number of times it incorrectly identifies a low-risk event as high-risk. However, further research is required to refine these results, explore parameter optimization strategies, and enhance the model's adaptability to evolving fraud patterns. Future work will focus on optimizing thresholds, improving model robustness over time, and ensuring effective detection of new fraud schemes. This research improves model performance by optimizing key parameters and enhancing detection accuracy while minimizing false positives

Corresponding Author:

Abdusy Syarif, Universitas Nasional, Indonesia
Email: abdusy.syarif@civitas.unas.ac.id

1. INTRODUCTION

The rapid growth of disruptive technologies like smartphones, mobile payments, and cloud computing has been accompanied by an alarming rise in large-scale data breaches. These events have given rise to new fraud strategies that evolve more quickly over time, making current detection methods less effective. Society, in general, views fraud as a minor crime, and its effects are mitigated by reimbursement of the victim by service providers. However, it has been identified that criminal organizations use credit card fraud to finance their activities (arms, drugs, and terrorism) [1]. Our society today is a cyber society that depends on the availability, accuracy, and confidentiality of information stored in databases, and fraud threatens trust in its systems. Fraud prevention and identification systems are therefore a necessity to guarantee the stability and progress of our society.

The global surge in online transactions has also significantly impacted Indonesia. The online trading volume has been increasing for the last decade. According to The Ministry of Communications and Informatics, the growth in e-commerce in Indonesia reached 88.1%, which is the highest in the world. While the exact prevalence of fraud in online payments varies, it remains a significant challenge for e-commerce businesses. Recent studies have shown that a substantial percentage of online transactions are targeted by fraudsters, resulting in significant financial losses and reputational damage. The fraud's cost is estimated at 4.62 trillion rupiahs [4].

Indonesia's booming e-commerce industry has experienced exponential growth in recent years. However, this rapid expansion has also made it a prime target for fraudulent activities. The unique challenges posed by the Indonesian e-commerce landscape, such as diverse payment methods, complex logistics, and a rapidly evolving digital ecosystem, necessitate innovative and tailored fraud detection solutions. In Indonesia, For instance, if a customer is a victim of fraud, he or she can report or claim it to the bank or the police. The bank often must investigate, refund the customer, and dedicate resources to fraud prevention.

On the other side of the world, like in France, in order to avoid credit card fraud, the common strategy used is to apply the 3D-Secure (3DS) system [2]. In this case, the system will ask the user for additional information. If the user can provide the information asked correctly, the transaction is accepted, otherwise, the transaction is refused. However, according to statistics [1], such an authentication system leads to around 16% of abandon transactions rate, which is not the ideal method for merchants. Because the 3DS system often involves multiple steps, including password entry and One-Time Password (OTP) verification [3], which can disrupt the smooth checkout flow. This friction can frustrate customers, leading them to abandon their purchases. Additionally, the system may not always be seamlessly integrated with various devices and browsers, further hindering the user experience. These factors collectively contribute to the high abandonment rates associated with the 3DS system in France.

Despite significant advancements in fraud detection techniques, the ever-evolving strategies of fraudsters pose a persistent challenge. Existing methods often struggle to detect sophisticated fraud patterns, especially in real-time scenarios. This research aims to address this gap by developing a robust and efficient fraud detection system that can accurately identify and prevent fraudulent activities, even in the face of evolving fraud techniques.

In the Indonesian e-commerce landscape, fraud poses a significant threat to both businesses and consumers. The financial implications of fraud, including chargebacks, refunds, and lost revenue, could be substantial. Moreover, fraudulent activities can erode consumer trust, damage brand reputation, and hinder business growth. To address these pressing issues, a robust and efficient fraud detection system is essential.

That leads us to a second strategy: using a classification algorithm and then activating security protocols only for high-risk transactions, i.e. high fraud probability. This strategy allows companies to take benefit from a highly efficient way to process transactions and provides better accuracy than manual techniques.

This study contributes to the ongoing development of fraud detection systems by introducing an AI-based model that leverages machine learning techniques to enhance the accuracy and efficiency of detecting fraudulent transactions in e-commerce. Unlike traditional methods, such as 3D-Secure, which often lead to a high rate of transaction abandonment (approximately 16%), our approach minimizes disruptions for legitimate transactions by activating security protocols only for high-risk transactions. This selective approach improves overall transaction efficiency while reducing the likelihood of false positives. A significant contribution of this research lies in the fine-tuning of key parameters—specifically, a training period (Δt_{train}), the lag period (Δt_{lag}) and the high-risk pass fraction ($frac\_hr\_pass$), which optimize the performance of the model over time, and through extensive simulations, we demonstrated that specific configurations of these parameters can balance detection efficiency and reduce false positive rates.

2. RELATED WORK

Cashless payments can be used to pay for goods or services using a payment card or e-money without the need for physical banknotes. However, this payment method has some criminal acts of deception or fraud. A fraud vector consists of a specific sequence of procedures to undertake payment card fraud which has been subsequently detected or recognized by law or fraud experts and reported.

Tuyls *et al.* [5] mentioned some challenges regarding fraud detection applications, such as the highly unbalanced datasets, where only a small percentage of the available data is fraud, which makes training efficient models quite difficult to achieve. In addition, there are some related studies that have applied deep learning and machine learning models in this area. Some machine learning methods have been used to detect frauds such as K-Nearest Neighbors (KNN) [6], Support Vector Machines (SVMs) [7], K-Means [8], Naïve Bayes [9], Random Forest [10], and many more. Most of these studies deal with unbalanced datasets where the amount of fraud data is very small. Moreover, the main evaluation metrics that they used are True Positive Rate (TPR), False Negative Rate (FNR), and Matthews Correlation Coefficient (MCC). However, some studies provide a recommendation to use Neural Networks as an alternative solution for unbalanced dataset issues.

For almost three decades, a large amount of research has been conducted in the area of fraud detection, but it has had a marginal impact on the market. Even if the results of the research remain academically valid, it has not attracted market attention for the following reasons [11]:

- Many kinds of research are carried out on synthetic data and/or with a reduced size which do not represent a real situation, which calls into question the relevance of the study;
- Results are expressed in terms of academic metrics and not in terms of more market-oriented metrics, for example, impact on cost reduction;
- Studies do not show the viability of implementation and scaling up (millions of transactions per day) of algorithms in a fraud management system;
- Studies do not show the ability to respond in real-time (less than 1 second);
- Most of the proposed algorithms are complex and opaque, which means that the reason for their decisions is not easily understandable by human beings.

One of the earliest and most widely applied techniques in fraud detection involves the use of supervised learning algorithms, such as logistic regression, decision trees, and random forests. These models are trained on historical data, where labels indicate whether a transaction was fraudulent or legitimate. For example, [12] demonstrated that decision trees, neural networks, and support vector machines (SVM) are effective for credit card fraud detection, achieving high accuracy when trained on large, well-labeled datasets .

Recent advances in deep learning have also been applied to this domain. Variational Autoencoders (VAE) and Generative Adversarial Networks (GANs) have been used for fraud detection by learning complex, latent representations of legitimate transactions, enabling the model to flag anomalous behaviors that do not conform to these learned patterns. A study by [13] applied deep learning techniques to identify fraudulent transactions in real-time with an enhanced ability to detect complex fraud patterns that traditional methods might miss [14].

Furthermore, unsupervised learning approaches, such as clustering and anomaly detection, have gained prominence for fraud detection when labeled data is scarce. Techniques like K-means clustering and Isolation Forests have proven useful in identifying outliers in transaction data. For instance, [15] developed an anomaly detection model using Isolation Forests to detect credit card fraud in scenarios where fraudulent transactions are highly imbalanced compared to legitimate transactions. This approach allowed for more effective detection in cases where labeled data was insufficient for supervised training.

Additionally, hybrid models that combine supervised and unsupervised learning have shown promise in improving detection performance. Hybrid techniques leverage the strengths of both approaches to balance accuracy and detection efficiency. [16] proposed a hybrid deep learning model combining convolutional neural networks (CNNs) with long short-term memory (LSTM) networks to detect fraudulent transactions based on both temporal and spatial features of transaction data. The model demonstrated improved performance over traditional ML techniques, particularly in identifying evolving fraud patterns in dynamic online environments.

In the field of fraud detection, fine-tuning has gained importance because of the dynamic and evolving nature of fraudulent behaviors. As fraud patterns constantly change, it is crucial that fraud detection models are continuously adapted to recognize new and emerging threats. Fine-tuning enables models to adjust to these new patterns without the need for complete retraining, which is both time-consuming and computationally expensive. For instance, Zhuang et al. [17] highlight the effectiveness of fine-tuning machine learning models in detecting financial fraud by continuously updating the models with recent transactional data, improving their accuracy and reducing false positives over time.

A common approach to fine-tuning involves pre-training models on a large, generic dataset and then fine-tuning them on a more specialized dataset relevant to the task at hand. For example, BERT (Bidirectional Encoder Representations from Transformers) and other transformer-based models have been widely used in various natural language processing tasks, where fine-tuning on domain-specific data has shown to significantly improve model performance [18]. This concept is now being applied to fraud detection, where a model pre-trained on general transactional data can be fine-tuned using more recent and domain-specific fraud data. By fine-tuning on updated datasets, models become more adept at identifying patterns indicative of fraud that may have been absent in the original training data.

Fine-tuning is also vital for handling the challenge of imbalanced datasets, which is a common issue in fraud detection where legitimate transactions vastly outnumber fraudulent ones. In the work of Chawla et al. [19] explored this challenge through the application of fine-tuning techniques such as Synthetic Minority Over-sampling Technique (SMOTE) combined with ensemble methods to enhance fraud detection in imbalanced datasets. Fine-tuning parameters of ensemble models, such as XGBoost or Random Forests, has proven effective in balancing the trade-offs between sensitivity and specificity, particularly in highly imbalanced scenarios [20].

Furthermore, model fine-tuning can also help mitigate model degradation over time. In real-time fraud detection systems, model performance may degrade as new fraud patterns emerge and legitimate behaviors shift. To address this, continuous fine-tuning of models is necessary. Zhou *et al.* [21] demonstrated that fine-tuning models on periodic intervals with newly accumulated data improves their adaptability, thereby maintaining detection accuracy and minimizing false positives in changing environments.

These various approaches highlight the ongoing research and innovation in applying AI and machine learning to fraud detection. While substantial progress has been made, challenges remain, particularly in handling evolving fraud patterns and imbalanced datasets. This motivates the need for continuous exploration of new methods and model fine-tuning to maintain fraud detection accuracy in real-world applications.

3. PROPOSED WORK

The state of the art of this research for fraud detection must therefore target results that have shown some relevance to the sector, and also keep an open mind for new ideas that might well apply to our problem. The development of an anti-fraud solution requires the choice of methods/algorithms in all stages of selection, cleaning, and preparation of data as well as the identification algorithms to be deployed. The following discussion addresses these topics which will be discussed below:

- Supervised versus unsupervised algorithms. The problem of identifying fraud is translated in the context of data science as a binary classification problem, where we will classify a new transaction either as fraudulent or as legitimate. A supervised fine-tuning process is employed, wherein a supervised model is retrained on the detected anomalies or patterns to optimize its precision and recall. In reality, the algorithms produce a score or probability of fraud, and if this exceeds a predefined threshold, the transaction is considered high risk.
- More tailored optimization metrics. As mentioned earlier, most research optimizes and evaluates their algorithms in terms of metrics popular in academia, which refers to the development of specific performance measures that are highly relevant to the unique challenges and goals of a particular fraud detection system. Several research teams use the accuracy metric [22], which is the precision in the classification by the algorithm. The problem with this metric is that it ignores the highly imbalanced nature of the data, 1 fraudulent transaction per 1000 legitimate ones. We can therefore have an algorithm that considers all transactions as legitimate, which gives an accuracy of 99.9%.
- Feature Engineering Strategies. In most cases, a single transaction does not provide enough information to separate legitimate transactions from fraudulent ones. Most fraud detection methods apply aggregation strategies that use customer histories to better understand customer habits [23]. Current transaction information is compared with customer habits to detect fraud more effectively. The definition of aggregates as well as the frequency to calculate them are key parameters in the optimization of these models.
- Concept/variance-drift and updated models. A trained model loses performance as new fraud strategies are implemented (concept-drift), or as customer habits change over time (variance-drift). Models must update with a certain regularity to maintain acceptable performance over time. So-called passive methods trigger an update with a predefined frequency. The optimal frequency is very dependent on the economic sector (e-commerce, banking, etc.). It must be obtained with prior analysis of historical data. Furthermore, there are no guarantees that this optimal frequency will remain constant over time. A better approach is to trigger an update dynamically when one or more Key Performance Indicators (KPIs) in the model degrade beyond a predefined threshold. This approach requires monitoring all KPIs over time. Most interest KPIs should be calculated with a set of labeled transactions within a time range. A real-time calculation of KPIs is not possible because it takes a long time to definitively label transactions as fraudulent. We will study the possibility, in the absence of definitive labeling, of applying statistical methods to calculate and correct KPIs based on historical data. There are methods that propose to include concept variables and variance drift between the inputs passed to the classification algorithms. These methods promise to remove the need to trigger automatic model bets but require constant calculation (once per day, week, etc.) of concept and variance drift indicators. Several statistical methods and monitoring systems can be employed to track drift over time, such as machine learning techniques, performance metrics and continues monitoring are used in this research. We will explore the strategies mentioned here to ensure good performance over time of the deployed models.

As discussed in [24], the LTM-memory model allows to train various standard algorithms and the approach has provided good results. Nevertheless, it considers credit card transactions as isolated events and not as a sequence of transactions.

A very important aspect of the algorithm optimization, is the hyperparameters/parameters fine tuning. In this case, there are usual ones: regularization importance, algorithm learning rate, optimizer, and initialization. There are the ones involved by our choice of algorithm: number of block, number of layers in each block, type of activation function, minimum size for the sub-series, to name a few. Good tracks can be found in [25]. However, deep neural network architectures are hard to train and suffer from problems such as vanishing gradients [26].

Due to the large research area in this project, we decided to focus on the last one, i.e. updating the model. The objectives of this research are related to extracting data for model training and data storage for future model updates. The mission is to study and optimize the parameters for data storage and extraction for model training, in order to find the best model performance and stability over time. Ethical considerations, notably the risk of false positives, are of utmost importance. While the primary aim is to detect fraudulent transactions, mitigating false positives and their potential impact on legitimate customers is a key objective. A robust model training strategy is employed to achieve this goal. To do so, some aspects need to be investigated and optimized.

This proposed scheme, called Artificial Intelligent Fraud Detector (AI-Fraud Detector), has chosen machine learning (ML) technology because it could provide a good detection efficiency of fraudulent transactions while keeping a reasonable rate of miss-classified healthy transactions. In addition, machine learning falls within a fully automated process, and allows to detect variations of fraudster's habits and to update the models accordingly.

The main target clients of the AI-Fraud Detector application are banks and e-commerce sites. In such cases, the AI-Fraud Detector will intervene between the bank and the e-commerce site to assess the incoming transactions. AI-Fraud Detector's operation workflow is described in Fig. 1. Firstly, an incoming transaction gets scored in terms of its probability of being a fraud or a legit. AI-Fraud Detector has an interface through which the clients send all the transactions into AI-Fraud Detector's database, and then these transactions are evaluated in terms of fraud risk by a Machine Learning model. Secondly, the 3DS is triggered only in case of high-risk transactions. Then, the incoming transaction enriches the database used for Machine Learning (ML) and Artificial Intelligence (AI) model training for fraud detection.

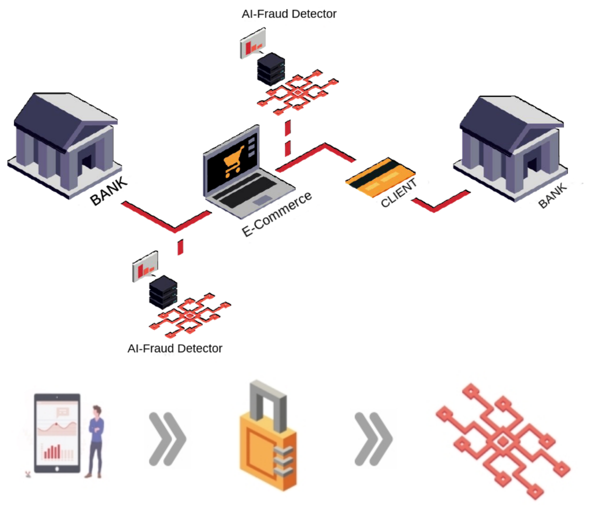

Fig. 1. Top: AI-Fraud Detector's Intervention for Transactions Assessment, Bottom: AI-Fraud Detector's Operation Work-Flow

The research steps taken to reach the objective are extracting the data into train, validation, and test sets; simulating the process of training, predicting, data storing, tagging, and retraining processes; and evaluating algorithm performance using relevant metrics. We repeat this steps to see the performance between one and another configuration (see Fig. 2).

3.1. Transaction Storage and Tagging Dynamics

The task at hand is a binary classification task, which consists of classifying client's recent history either as fraudster or legitimate. As explained in previous section, each client who presents a fraudulent transaction is tagged as a fraudster. As a consequence, the data is splitted according to the clients (and not the transactions). A client is defined by a set of transactions ordered in time. In order to start simple, we're going to consider the amount and the time of the client's transactions, as the recommendations given in [35].

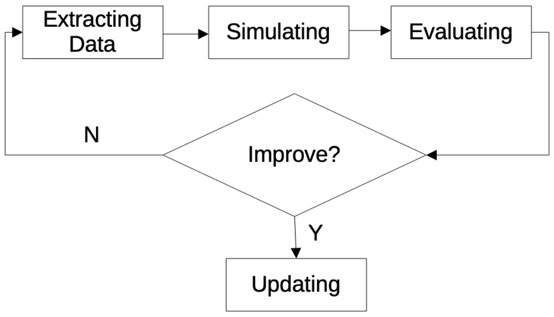

Fig. 2. Research Flowchart

In order to train a model the data needs to be tagged, and the transactions used for training have to include a variable that identifies them as either fraud or legitimate. This will be certainly the case when the first batch of transactions from the clients arrive at the AI-Fraud Detector's database. We could then use them to train and deploy a model to assess the fraud probability of new transactions.

As the new transactions arrive, the AI-Fraud Detector assesses them and they get stored in a database. The stored information will be the transaction parameters as well as the corresponding fraud probability. This is not the end of the story, as new information is created as the transactions continue to be processed.

Depending if the fraud probability (P_{fraud}) is higher or lower than a predefined threshold ($pThr$), the client decides what action to take:

1. low-risk transaction ($P_{fraud} < pThr$), then just let it pass;
2. high-risk transaction ($P_{fraud} >= pThr$), trigger a 3DS.

In the case of 3DS, the transaction could pass or be stopped, depending if the required information is correctly provided.

The result of all these additional steps could be used to start the tagging process of the incoming transactions. For example, a transaction for which a 3DS was applied and passed is very likely to be a legitimate transaction. However, for a transaction in which a 3DS was triggered is stopped, there is no certainty that it is fraudulent as there will be some legitimate transactions that by chance will not pass the 3DS due to (internet connection problems, cellphone bad signal, etc).

A credit card transaction for which no 3DS is required and passes, could be either legitimate or fraudulent. A legitimate transaction never gets tagged, as no client will report it as a fraud. However, a fraudulent transaction gets tagged after a certain time which depends on the fraud processing procedure as follows;

1. a fraudulent transaction is executed at time $t_{transac}$;
2. a legitimate client reports the fraud;
3. the bank/e-commerce investigates;
4. the investigation confirms a fraud;
5. the transaction gets tagged, *i.e.* $fraud = True$ at time t_{tag}.

Furthermore, a transaction could be tagged either as legitimate or fraudulent, it will take some time. If we take a look at to panel of Fig. 3, it shows the tagging delay (in days) distribution of fraudulent transactions. Almost all fraudulent transactions ($\sim$ 99%) are tagged after 116 days ($\sim$ 4 months) since they were executed. By using the normalized cumulative distribution (NCD) curve (see bottom panel of Fig. 3), we can define a tag-time window to declare all untagged transactions as legitimate. For instance, in our current case, declaring all untagged transactions older than Δt_{tag} = 94 days incurs a tagging error 10%, that is, after this time period there are still 10% of fraudulent transactions waiting to be tagged, which in this case would be considered legit for the purposes of model training.

The optimal tag-time window should be calculated from the data to maximize prediction performances. It will be a compromise between the tagging precision and the frequency of model updates.

To train a model $today$ (td), we can not just use all the previous data. We need to go back sometime at a lag period (Δt_{lag}) in the past and extract the date after $(date = td + \Delta t_{tag})$, in order to avoid a significant tagging error. Fig. 4 illustrates the process of data extraction. We first need to go back in time an amount Δt_{lag} and then extract the data at time window Δt_{train} before that, *i.e.* the data for training will be between the times $t_1 = td - \Delta t_{lag} - \Delta t_{train}$ and $t_2 = td - \Delta t_{lag}$.

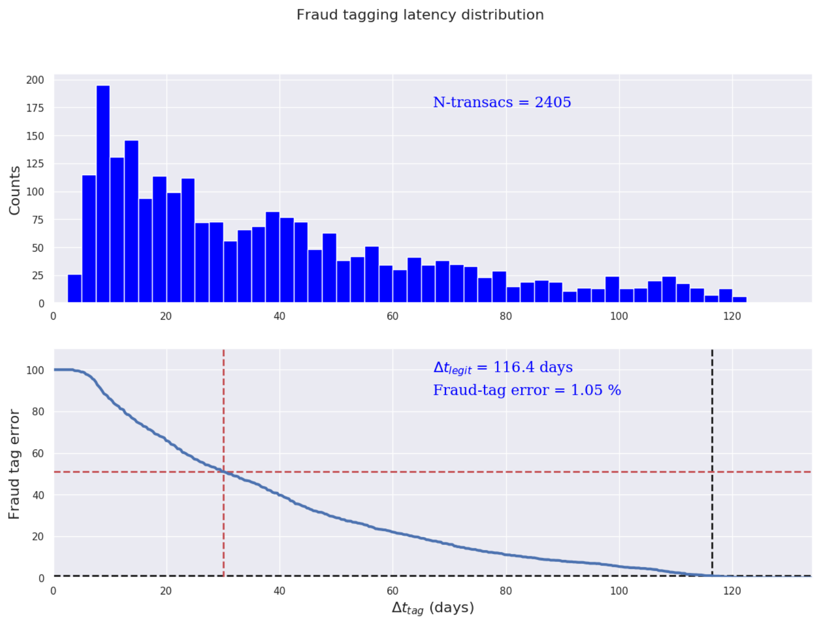

Fig. 3. Top: Tagging Delay Distribution. Bottom: Normalized Cumulative Tagging Delay Distribution Obtained From the Top Panel. the Black-Dotted Lines Show That 116 Days After the Transaction Execution Date, Only 0.1% of Fraudulent Transactions Are Still Waiting to Be Tagged

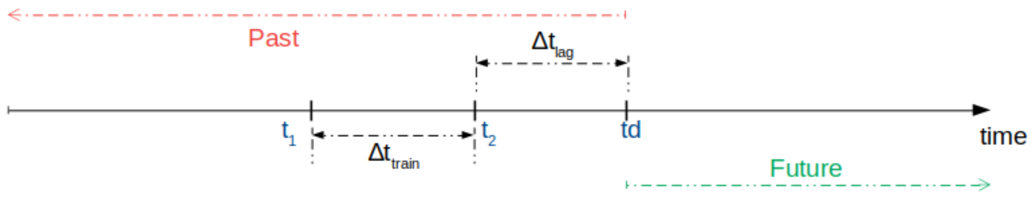

Fig. 4. Illustration of the Dataset at an Interval of Time

However, this might raise problems when the lag period (Δt_{lag}) is too small or too big. If Δt_{lag} is too small (for example 1 month), there are too many untagged fraud transactions in the dataset. However, while Δt_{lag} is too big (for example 5 years), the training dataset is not up-to-date anymore and as a consequence, the model would not be very good at present. Moreover, another issue might come up when Δt_{train} is too small or even too big. if the training period (Δt_{train}) is too small (for example 1 month), there will be not enough data for training and it will give poor performances. Nonetheless, if Δt_{train} is too big (for example 5 years), most of the data for training is not updated since the fraudulent transaction may vary from time to time. Thus, there is a need for optimization to be found in this case.

3.2. High-Risk Transactions and Model Update

There is another issue of data storage which is very important for model updates. All the transactions with a fraud probability beyond a certain threshold (called high-risk transactions) will be either require a 3DS, as seen in Fig. 5. Legitimate transactions are very likely to pass the 3DS requirement. Conversely, high-risk fraudulent transactions will be blocked, thus they won't be available for model training in the future.

This will significantly bias the data sets for model updates, as they will lack these high-risk fraudulent transaction examples. Models trained with this biased dataset will then "forget" how to identify high-risk fraudulent transactions, and will have poor performances with new data. In addition, in the context of machine learning, particularly when dealing with imbalanced datasets, it's crucial to assign appropriate weights to different samples to ensure fair representation during training. This scheme implies a scenario where certain samples (those that triggered a 3DS) are overrepresented due to a random selection process.

One solution to this problem could be to randomly select a fraction f (for example 5 or 10%) of the high-risk transactions and artificially assign them a low fraud probability (see Fig. 6). In this way all of them will pass, including the fraudulent transactions. This last ones will then follow the natural process of fraud processing and tagging, and will then be part of the datasets used for model update. During training, however, we would need to assign them a weight of $1/f$ to take into account the random selection and give them the proper relative importance they would have if no 3DS was triggered. To ensure that samples with 3DS triggers are not overrepresented, we employ a weighting mechanism. First, we calculate the fraction f of 3DS triggers within the dataset. Subsequently, we assign weights to each sample: samples without 3DS triggers receive a weight of 1, while samples with 3DS triggers are assigned a weight of $1/f$. This weighting scheme effectively normalizes the influence of overrepresented samples and ensures that each sample contributes to the loss function in proportion to its actual frequency in the population.

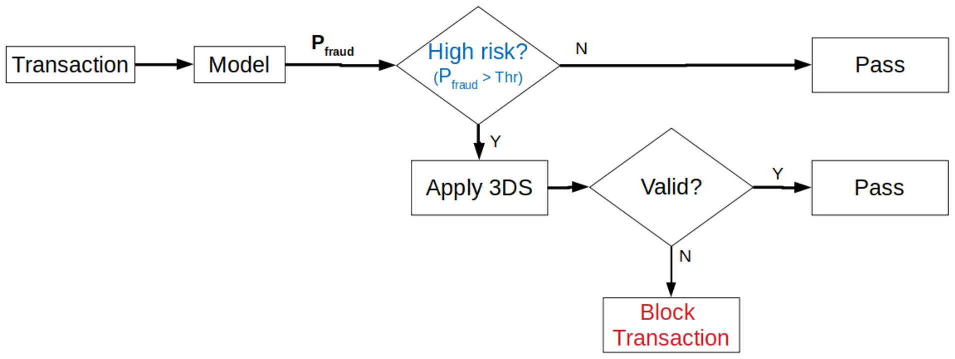

Fig. 5. Common 3DS Workflow for High-Risk Transactions

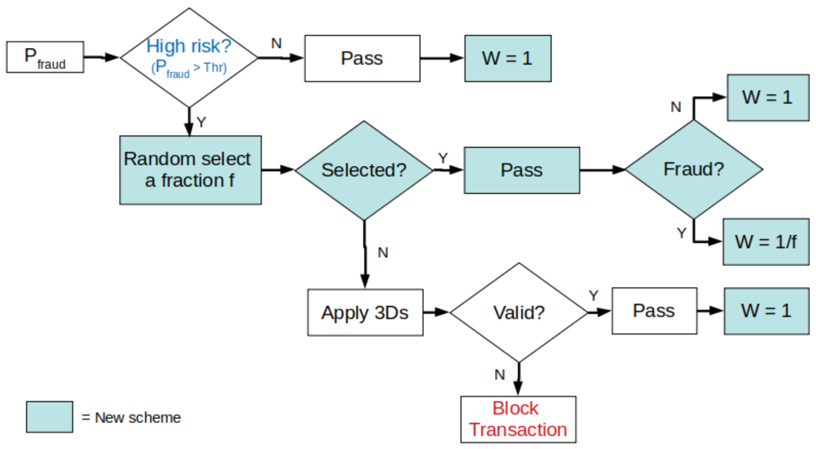

Fig. 6. Weighting and High-Risk Pass Selection Proposed Scheme

A disadvantage of this procedure is that fraud detection performances will be degraded. For instance, let's suppose that we set up $pthr$ such that it corresponds to a fraud detection efficiency of $\epsilon_{fraud} = 90\%$ and a False Positive Rate of $fpr = 10\%$. Randomly selecting a fraction $f = 5\%$ of high-risk transactions and letting them pass means that we will call a fraction f of these high-risk fraudulent transactions legitimate. This is equivalent to a deterioration of the fraud detection efficiency, giving an effective value of $\epsilon_{fraud}^{eff} = \epsilon_{fraud}(1-f) = 85.5\%$ (a reduction of 4.5 %). This will have a milder effect on the fpr, giving an effective value of $fpr^{eff} = fpr(1-f) = 9.5\%$ (a reduction of 0.5 %).

Thus, the optimal fraction f for high-risk transactions should be estimated from the data to maximize prediction performances. It will be a compromise between the current and future model performances.

To summarize, the purpose of this study is to find a way to optimally select the values of the data extraction parameters Δt_{train} and Δt_{lag}, as well as the fraction f for the data storage. The method to achieve this is described in the next section.

4. FINE TUNING OF DATA EXTRACTION AND DATA STORAGE PARAMETERS

In this project, we will optimize (tune) some parameters. First, we need to adjust the time window of the training data (Δt_{train}) and how far we are going in the past from today (td) to define Δt_{lag}. And the second one, what's the fraction of high-risk transactions that we will let pass or high-risk pass fraction ($frac\_hr\_pass$). These parameters are needed to find an optimal combination of the parameters. We simulate the process of data extraction, model training, model deployment (performing prediction for a certain time), and data storage on the evaluated model in order to find the best and most stable performance over time. Fig. 7 illustrates the main method we perform in fine-tuning of data storage parameters in order to find the optimum model.

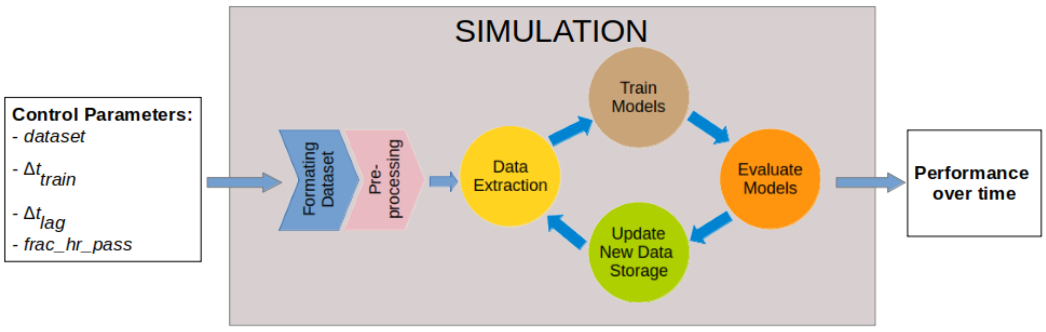

Fig. 7. Fine tuning data storage parameters for model training

As we can see we have control parameters as the inputs to adjust. The control parameters are as follows:

- $data\_path$ is a full path to the dataset directory. The dataset used in this study is taken from AI-Fraud Detector's database, dated from 01 January 2016 to 01 January 2018. The dataset contains 10913014 transactions (lines) with 26 variables (columns). Table 1 shows some data variables in the dataset. In addition, ethical considerations were paramount throughout the research process. We ensured strict adherence to data privacy regulations and implemented robust security measures to protect sensitive information. The dataset used in this study was anonymized and aggregated to maintain confidentiality [36].

Table 1. Data Variables

Variable	Description
transact id	Transaction's identifier
one-click	Yes/No, if the transaction is a subscription or one-click
3ds	Authentication step
ip country	Internet Protocol's country where the transaction has been effected
date	Transaction date
date cb	Date of the charge back
cb reason	Reason of the charge-back
amount cb	Amount of charge-back
client id	Client's identifier
card country	Geographical origin of the card
site id	Site identifier
site name	Site name

- Δt_{train} is the number of days for training. It is mandatory. For example, we can give Δt_{train} of 180 days ($\sim$ 6 months) which means that we are going to take 180 days of time-window data of the dataset for training models (see again Fig. 4).

- Δt_{lag} is the number of days for a given error corresponding to tagging delay. For example, if we give an error of 10%, it means that we add about 93 days of time-window data of the dataset into training models. Further, we create a function to calculate Δt_{lag} from a given error which is based on the normalized cumulative distribution (see Fig. 3).
- $frac\_hr\_pass$ is the fraction of high-risk passed. For instance, we could set up to 5%, meaning that it will randomly select 5% of high-risk transactions pass and will then be part of the dataset used for the model update.

The selection of parameters like Δt_{train}, Δt_{lag}, and $frac\_hr\_pass$ involves a careful balance between model performance and computational efficiency. In this case, for example, a Δt_{train} of 180 days was chosen after extensive experimentation and considering the some factors. i.e. data characteristics, model complexity, and model performance.

The simulation performs some operations with historical data over a period of time. There is a loop process inside the simulation where the loop is over several iterations depending on the given control parameters. Each iteration corresponds to performing the above process but with a shift in time window on the dataset, which in our current case is 30 days of interval, meaning that it periodically does the cycle every 30 days. The simulation executes some routines as follows:

- Formating dataset and Pre-processing. This stage involves several critical steps to prepare the dataset for model training. Firstly, the raw data is loaded from the specified $data\_path$. Missing values are handled by employing appropriate techniques, such as imputation or removal, depending on the nature of the missing data. Secondly, data cleaning is performed to identify and rectify inconsistencies, errors, or outliers. Subsequently, feature engineering is applied to extract relevant information from the raw data, which may involve creating new features or transforming existing ones. Finally, the dataset is formatted to ensure compatibility with the model, including tasks like dropping irrelevant columns, resetting the index, and initializing necessary parameters such as weights (w), fraud probabilities (P_{fraud}), and thresholds (Thr).
- Data extraction. This step involves extracting relevant training data from the dataset based on specific temporal parameters Δt_{train}, Δt_{lag} and $today$ (td).
- Train models. This step involves training the logistic regression model [27] on the prepared training dataset. The model is trained to learn the underlying patterns and relationships between the input features and the target variable (fraudulent or non-fraudulent). During the training process, the model's parameters (weights and biases) are iteratively adjusted to minimize the prediction error. The optimization process typically involves techniques such as gradient descent, which calculates the gradient of the loss function with respect to the model parameters and updates them in the direction of minimizing the loss.
- Evaluate models. After the models are trained, they are evaluated on a separate test dataset to assess their performance. The evaluation process involves predicting the probability of fraud (P_{fraud}) for each transaction. Based on these probabilities, transactions are categorized into low-risk and high-risk groups. To balance the need for accurate fraud detection with the potential impact on legitimate transactions, a weighting scheme is applied. Low-risk transactions are assigned lower weights, while high-risk transactions are assigned higher weights. A fraction ($frac\_hr\_pass$) of high-risk transactions is allowed to pass, while the remaining high-risk transactions are flagged for further investigation or blocked. This selective approach helps minimize false positives and false negatives. The specific implementation of this weighting and high-risk transaction selection scheme is detailed in Fig. 6, which provides a visual representation of the decision-making process.
- Update new data storage. Once the model evaluation is complete, the latest dataset is incorporated into the data storage system. This ensures that the model has access to the most recent and relevant data for future training and prediction cycles. By continuously updating the data storage, the model can adapt to evolving fraud patterns and improve its performance over time. This iterative process, involving data extraction, model training, evaluation, and data storage update, forms the core of the proposed fraud detection system.

We perform the simulation repeatedly by changing the control parameters values as seen in Table 2. We conduct 2 types of simulations, *i.e.* single fit, and retraining. Single fit means that a single model is trained at the beginning of the time period, and then deployed for the rest of the time period. This means that the model is static, as no retraining (model updates) is performed. On the other hand, retraining refers to re-running the process that generated the previously selected model on a new training dataset. Should incoming data exhibit signs of bias, a model retraining is necessary. These simulations are conducted to proactively mitigate such risks.

By the end of the simulation, it will give metrics that are used to score the performance of the model over time. The scoring metrics are false positive rate (fpr), true positive rate (tpr) and ROC AUC. These metrics are going to be explained in Section 5.

Table 2 provides the tuning parameters values. For instance, we can define a configuration parameters for Δt_{train} of 90 days ($\sim$ 3 months), Δt_{lag} of 93 days and $frac\_hr\_pass$ of 10%.

5. PERFORMANCES AND RESULTS

The implementation is based on, well-known and largely used, TensorFlow's sub-library called Keras [28], because of its efficiency and user friendly API. Concerning classes implementation (transformers and estimators), the chose design tends to stick with the scikit-learn API [29], known for its simplicity and coherence (see [30] for more details on the design of the API). To provide an easy to read and understandable code, we have tended to stick to the PEP8 style guide [31] and as programming principles to the PEP20 zen of Python [32].

Table 2. Tuning Parameters

Parameters	Value
Δt_{train}	[90, 120, 180, 270] days
Δt_{lag}	[110, 93, 71, 55] days
$frac\_hr\_pass$	[None, 5, 10, 20, 30] %

To evaluate the performance of the proposed approach, we conducted a series of experiments using a comprehensive dataset, detailed in Table 1. To ensure rigorous evaluation, we employed a k-fold cross-validation technique [33]. The best model selection is performed by applying grid-search-Cross-Validation (GridSearchCV) [34], which mainly consist in fitting the data with all possible values of model parameters, choosing the one with the best performances according to a predefined metric, as summarized in Table 2. The overall experimental process, from data preprocessing to model evaluation, is illustrated in Fig. 7.

A trained model predicts the fraud probability for any given transaction. We can always define a threshold above or below which the transaction could be considered as fraud or legitimate. For a given threshold, we will then have a false positive rate (fpr), which is the fraction of legitimate transactions considered fraud.

The best performance model that we expect to achieve here can reduce fpr in fraud prediction, thus it will reduce losses of money due to fraud. In the same token, we will have a fraud detection efficiency or true positive rate (tpr), which is the fraction of fraudulent transactions correctly identified. A high fraud detection efficiency value or a high of tpr means that great fraud transactions will be detected by our models. We use a tool called Receiver Operating Characteristic - Area Under Curve (ROC-AUC) from scikit-learn [29] to evaluate the performance of classification model's performance. ROC-AUC is the fpr versus tpr for every possible value of the threshold. We calculate fpr and tpr by using equations as follows:

$$fpr = \frac{Nl\_passed}{Nl\_total} * 100\% \tag{1}$$

$$tpr = \frac{Nf\_passed}{Nf\_total} * 100\% \tag{2}$$

where Nl_passed refers to number of legit transactions passed, Nl_total is total number of legit transactions, Nf_passed means number of fraud transactions passed, and Nf_total is total number of fraud transactions.

We have conducted several simulations by developing programs with Python and bash scripts. For clarity purposes, taking as an example of a configuration Δt_{train} of 180 days, Δt_{lag} of 93 days, and $frac\_hr\_pass$ of 10%. If we take a look at the log file, it provides some statistics for the first training as follows:

```
Extracting first td
1st td   = 2017-01-25 00:00:08
tmin     = 2016-01-01 00:00:08
tmax     = 2017-12-31 23:59:49

Training 1
```

```
  Update td = 2017-01-25 00:00:08

Training data:
  Nlegit total          : 2387804
  Nfraud total          : 8892
  Nlegit train-range    : 1462246
  Nfraud train-range    : 5322
  Nlegit lag-range      : 925558
  Nfraud lag-range      : 3570
  Nlegit total (select) : 1462246
  Nfraud total (select) : 5322
```

As we can see, with the given control parameters above, the data extraction phase gives the `1st td` on the 25th of January 2017, `tmin` at 1st of January 2017 and `tmax` at 31st of December 2017, which means 1-year dataset extracted. At the first training, $today$ is updated on the 25th of January 2017, and then it will provide training data for the first training.

For the retraining models, since the `interval` is fixed at 30 days and we have `performance delta time` of the dataset of 340.99 days, then we will have 12 training models for each configuration of fine-tune data storage parameters. At the phase of evaluation models of simulation, it will give performance metrics at the first interval as follows:

```
Performances delta time = 340.99983796296294 days
Data time window: (2017-01-25 00:00:03,2017-12-31 23:59:49)
interval = 30.0 days
1 Looking at interval: (2017-01-25 00:00:03,2017-02-24 00:00:03) delta 30 days 00:00:00
  Nf_passed = 743, Nf_total = 1148, tpr = 64.721
  Nl_passed = 35024, Nl_total = 337049, fpr = 10.391
  roc_auc = 86.705 %
  Nf_passed_eff = 669, Nf_total = 1148, tpr_eff = 58.275
  Nl_passed_eff = 31522, Nl_total = 337049, fpr_eff = 9.352
  roc_auc_eff = 81.176 %
  days_since_training = 30.0 days
  date pred = 2017-02-24 00:00:03
  Ntransacs = 338.197 k
```

As we can see, `performance delta time` of dataset is 340.99 days, `data time window` is from the 25th of January 2017 till the 31st of December 2017, with `interval` of 30 days, and at the first training we will have number of fraud passed (`Nf_passed`) is equal to 743 over 1148 of total number of fraud (`Nf_total`), which gives `tpr` of 64.721%. The number of legit passed (`Nl_passed`) is 35024 transactions over 337049 of `Nl_total`, which provides `fpr` of 10.391% and the `roc_auc` of 86.705%. In terms of fraud detection efficiency, we have number of fraud passed (`Nf_passed_eff`) of 669 over 1148 of total fraud transactions, giving an effective value of `tpr_eff` = 58.275%. The effective number of legit passed (`Nl_passed_eff`) obtained 31522 over 337049 transactions, which gives an effective value of `fpr_eff` = 9.352%.

The analysis of the results suggests several avenues for optimization. To enhance the model's sensitivity, adjusting decision thresholds and incorporating additional relevant features can be considered. Additionally, fine-tuning the data window size and interval length can further optimize the model's performance, balancing the need to capture recent trends with computational efficiency.

To enable comparison in terms of false positive rate (fpr), we assess the model comprehensively. Our simulations involve single fit and retraining models with different values of parameters. The optimal approach depends on various factors, including the frequency of data updates, the complexity of the model, the available computational resources, and the desired level of performance. In many real-world scenarios, a hybrid approach that combines elements of both single-fit and retraining can be effective. A potential approach is to first train a model on a substantial dataset. Subsequently, the model can be refined through periodic retraining on smaller subsets of new data to strike a balance between computational efficiency and model performance. As depicted in Fig. 8, for example, we can see the comparison of single fit and retraining with configuration Δt_{train} of 180 days, Δt_{lag} of 93 days, and $frac\_hr\_pass$ of (10%, 30%). We use a boolean parameter ex_{lag} where it is assumed to be equal to True, meaning that it will exclude tagged fraudulent transactions in the time interval $(td - \Delta t_{lag}, td)$ for model training. With 30 days of interval, meaning that the model will be evaluated and predicted once every 30 days. This simulates the process of periodic model updates and their effect on model performances vs time.

The Single fit model (blue line) can achieve around 10% of fpr at the first 4 time-window and then it decreases till the end of the period of data which is around 4% of fpr. In addition, the single fit looks more stable over time compared to the retraining models (red and green lines). It shows that the retraining models have not improved significantly the model in general. This could be due to the variations on fraudulent are not many in the dataset we used. A sensitivity analysis of the model's performance suggests that the Single Fit model exhibits remarkable stability over time, maintaining a consistent low False Positive Rate (FPR) throughout the evaluation period. In contrast, the Retraining models, while initially promising, demonstrate fluctuating performance, potentially due to the limited variations in fraudulent patterns within the dataset. These findings highlight the importance of carefully selecting the appropriate training strategy, balancing the need for adaptability with the risk of overfitting. Further investigation into the dataset's characteristics and potential biases may provide insights into the underlying reasons for the observed performance differences.

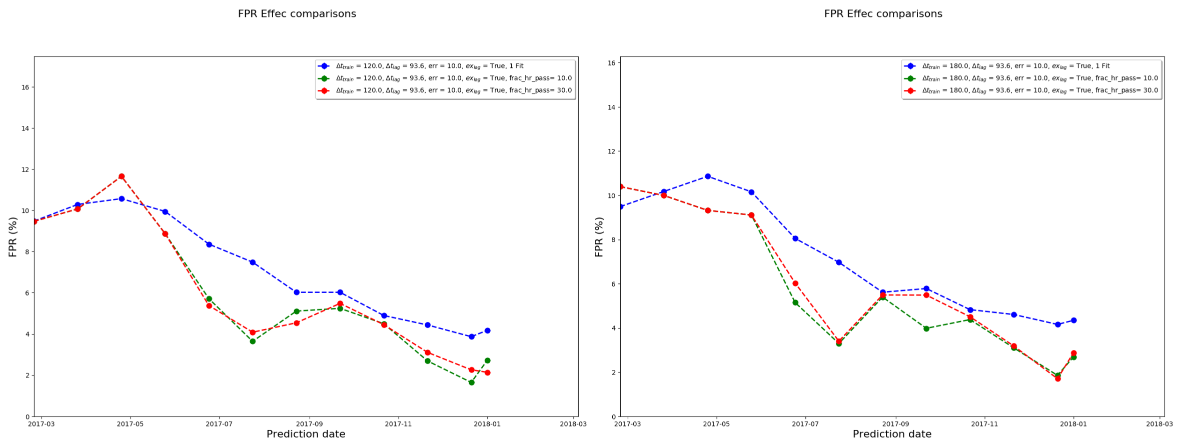

Fig. 8. Effective FPR of Single fit Vs Retraining Models for configuration Δt_{train} = 120 days and 180 days, Δt_{lag} = 93.6 days, with $frac\_hr\_pass$ of 10% and 30%

Fig. 9 depicts the detection fraud efficiency over time between single-fit and retraining models for different configurations. As we can see, in general, the single fit model (blue line) gives the preferable performance of fraud efficiency over time ($\sim$ 60%). It means that the single fit model can detect transactions that were in fact fraudulent. For a given tpr, the models give what losses will be due to fpr. In our study, the single fit has a tpr of around 60%, while retraining models (red and green lines) have a lower effective fraud efficiency in general. This suggests that the single fit model is more adept at accurately identifying fraudulent transactions, while maintaining a relatively low False Positive Rate (FPR). The retraining models, while exhibiting some variation, generally demonstrate lower effective fraud efficiency. This disparity may be attributed to factors such as the limited diversity of fraudulent patterns in the dataset or the inherent challenges associated with retraining models on evolving data distributions.

Fig. 10 shows the ROC AUC curves for single fit and retraining models with different configuration parameters. A model is better as its ROC AUC gets higher, meaning that a higher fraud detection efficiency or tpr for a lower fpr. As it can be seen that single fit model with Δt_{train} of 180 days gives a stable performance over time, which results in fraud detection efficiency of about 60% for a fpr of 10%. Our short observation is that Δt_{train} will depend on the transaction flux. In our studies, we found that configuration with Δt_{train} of 180 days obtained good results, but by that moment the transaction flux was much higher than the current one. This indicates its ability to effectively distinguish between fraudulent and legitimate transactions. The impact of Δt_{train} on the model's performance is likely influenced by the transaction flux. In our analysis, a Δt_{train} of 180 days yielded favorable results when the transaction flux was significantly higher. However, as the current transaction flux has decreased, further investigation is needed to determine the optimal Δt_{train} value for achieving optimal performance.

6. CONCLUSION

We have shown that the current AI-Fraud Detector's framework provides machine learning to detect fraudulent transactions. Some fine-tuning parameters have been optimized and simulated. In this fine-tuning simulations of data storage parameters for model updates of AI-Fraud Detector, we should constantly fine-tune parameters which are Δt_{train}, Δt_{lag} and $frac\_hr\_pass$ in order to achieve the optimum model over time. We use ROC-AUC, false positive rate (fpr), and true positive rate (tpr) as the metrics score since they are major criteria for fraud detection systems.

According to the results of our simulations, heuristically we could say that the preferable model we can get is a given configuration for the single fit model with Δt_{train} of 180 days, Δt_{lag} of 90 days, and $frac\_hr\_pass$ of 10%. This model achieved a robust performance with an ROC-AUC score exceeding 80% and a false positive rate (fpr) below 6%. This indicates strong accuracy in detecting fraudulent transactions while minimizing false alarms. While this configuration demonstrated reasonable detection efficiency ($\sim 50\%$), it was initially expected to perform less favorably. Further analysis is necessary to fully understand the underlying factors contributing to these results. Additional simulations with varied datasets and parameter settings will provide further insights. Additionally, a more refined methodology for selecting the optimal configuration is required.

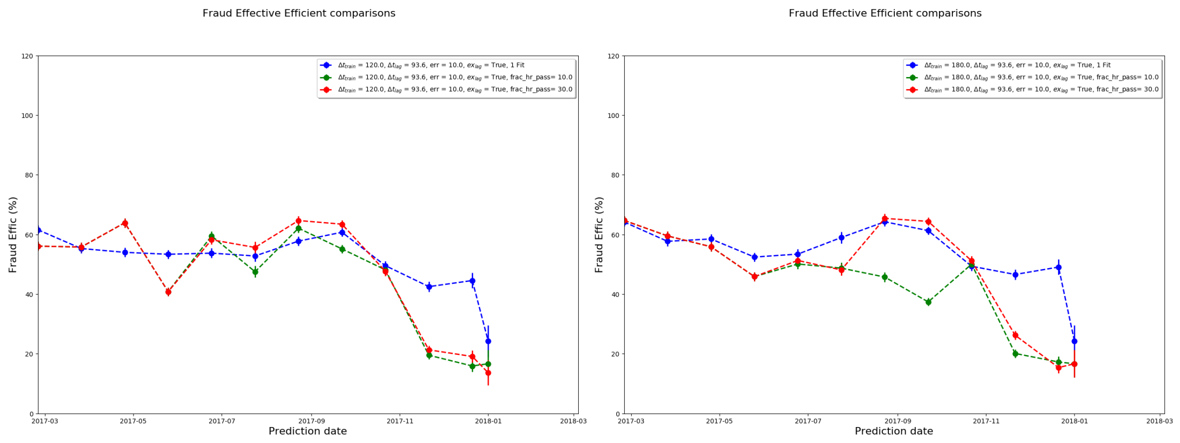

Fig. 9. Effective Fraud Efficiency of Single fit Vs Retraining Models for configuration Δt_{train} = 120 days and 180 days, Δt_{lag} = 93.6 days, with $frac\_hr\_pass$ of 10% and 30%

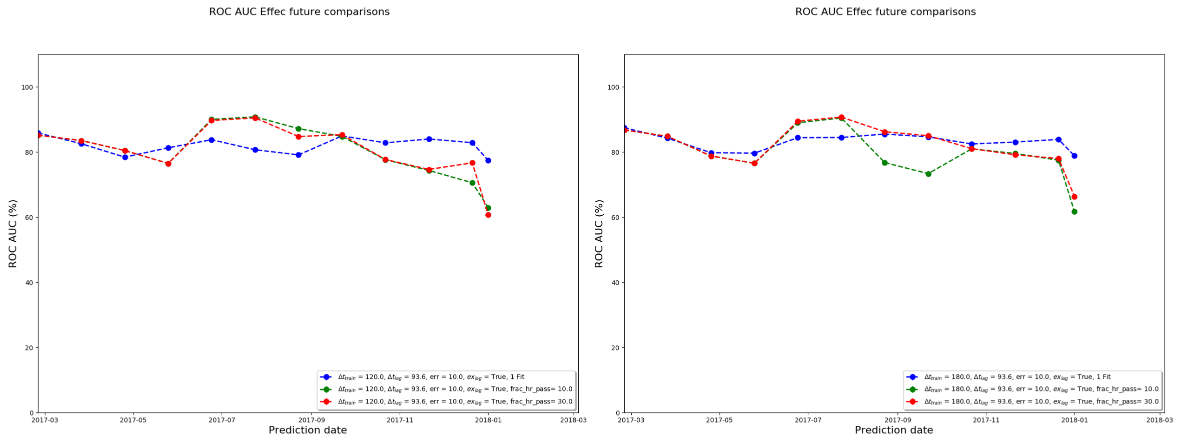

Fig. 10. Effective ROC-AUC of Single fit Vs Retraining Models for configuration Δt_{train} = 120 days and 180 days, Δt_{lag} = 93.6 days, with $frac\_hr\_pass$ of 10% and 30%

Some future works that might be undertaken such as how to define threshold optimally, understand how to detect new and evolving fraud patterns, and model performance variations over time, and also to find a way to define the parameter values of Δt_{train}, Δt_{lag} and $frac\_hr\_pass$ optimally.

REFERENCES

[1] Cybersource Article, Online - last access 14 april 2024, https://www.cybersource.com/content/dam/cybersource/2017_Fraud_Benchmark_Report.pdf.

[2] M. A. Ali and A. V. Moorsel, "Designed to be broken: A reverse engineering study of the 3D Secure 2.0 Payment Protocol," *Financial Cryptography and Data Security*, vol. 11598, pp. 201–221, 2019, https://doi.org/10.1007/978-3-030-32101-7_13.

[3] C. -L. Tsai, C. -J. Chen and D. -J. Zhuang, "Secure OTP and Biometric Verification Scheme for Mobile Banking,"

2012 Third FTRA International Conference on Mobile, Ubiquitous, and Intelligent Computing, Vancouver, pp. 138-141, 2012, https://doi.org/10.1109/MUSIC.2012.31.

[4] "Data Indonesia", Online - Last access 29 June 2024, https://dataindonesia.id/digital/detail/daftar-ecommerce-dengan-pengunjung-terbanyak-per-kuartal-i2022.

[5] S. Maes, K. Tuyls, B. Vanschoenwinkel, and B. Manderick,"Credit Card Fraud Detection Using Bayesian and Neural Networks," *Engineering Applications of Artificial Intelligence*, vol. 261, 2020, https://www.researchgate.net/profile/Karl-Tuyls/publication/254198382_Machine_Learning_Techniques_for_Fraud_Detection/links/555f695508ae6f4dcc926e88/Machine-Learning-Techniques-for-Fraud-Detection.pdf.

[6] M. Khodabakhshi and M. Fartash, "Fraud detection in banking using knn (k-nearest neighbor) algorithm," In *International conference on research in science and technology*, vol. 5, pp. 26–34, 2016.

[7] H. Zhang, Y. Shi, X. Yang, and R. Zhou, "A firefly algorithm modified support vector machine for the credit risk assessment of supply chain finance," *Research in International Business and Finance*, vol. 58, 2021, https://doi.org/10.1016/j.ribaf.2021.101482.

[8] Z. Huang, H. Zheng, C. Li, and C. Che, "Application of machine learning-based k-means clustering for financial fraud detection," *Academic Journal of Science and Technology*, vol. 10, no. 1, pp. 33-39, 2024, https://doi.org/10.54097/74414c90.

[9] A. Gupta, M. C. Lohani, and M. Manchanda, "Financial fraud detection using naive bayes algorithm in highly imbalance data set," *Journal of Discrete Mathematical Sciences and Cryptography*, vol. 24, no. 5, pp. 1559-1572, 2021, https://doi.org/10.1080/09720529.2021.1969733.

[10] C. Liu, Y. Chan, S. H. A. Kazmi, and H. Fu, "Financial fraud detection model: Based on random forest," *International journal of economics and finance*, vol. 7, no. 7, 2015, https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2625215.

[11] N. F. R. Tubb, and F. Nick, "How Artificial Intelligence and Machine Learning Research Impacts Payment Card Fraud Detection: A Survey and Industry Benchmark," *Engineering Applications of Artificial Intelligence*, vol. 76, pp. 130–157, 2018, https://doi.org/10.1016/j.engappai.2018.07.008.

[12] C. Phua *et al.*, "A comprehensive survey of data mining-based fraud detection research," *Artificial Intelligence Review*, vol. 34, no. 4, pp. 1-19, 2010, https://doi.org/10.48550/arXiv.1009.6119.

[13] U. Fiore *et al.*, "Deep learning-based approaches for credit card fraud detection," *Expert Systems with Applications*, vol. 117, pp. 267-277, 2019.

[14] U. Fiore *et al.*, "Using generative adversarial networks for improving classification effectiveness in credit card fraud detection," *Information Sciences*, vol. 479, pp. 448-455, 2019, https://doi.org/10.1016/j.ins.2017.12.030.

[15] P. Singh, K. Singla, P. Piyush and B. Chugh, "Anomaly Detection Classifiers for Detecting Credit Card Fraudulent Transactions," *2024 Fourth International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT)*, pp. 1-6, 2024, https://doi.org/10.1109/ICAECT60202.2024.10469194.

[16] K. Leena Kurien and A. Chikkamannur, "An Ameliorated hybrid model for Fraud Detection based on Tree based algorithms and Benford's Law," *2020 Third International Conference on Advances in Electronics, Computers and Communications (ICAECC)*, pp. 1-6, 2020, https://doi.org/10.1109/ICAECC50550.2020.9339471.

[17] A. A. Almazroi and N. Ayub, "Online Payment Fraud Detection Model Using Machine Learning Techniques," in *IEEE Access*, vol. 11, pp. 137188-137203, 2023, https://doi.org/10.1109/ACCESS.2023.3339226.

[18] J. Devlin *et al.*, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 4171-4186, 2019, https://eva.fing.edu.uy/pluginfile.php/524749/mod_folder/content/0/BERT%20Pre-training%20of%20Deep%20Bidirectional%20Transformers%20for%20Language%20Understanding.pdf.

[19] N. V. Chawla *et al.*, "SMOTE: Synthetic Minority Over-sampling Technique,", *Journal of Artificial Intelligence Research*, vol. 16, pp. 321-357, 2020, https://doi.org/10.1613/jair.953.

[20] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 785-794, 2016, https://doi.org/10.1145/2939672.2939785.

[21] V. R. K. Changalreddy and A. Jain, "Evolving Fraud Detection Models with Simulated and Real-World Financial Data," *International Journal of Research and Analytical Reviews (IJRAR)*, vol. 11, no, 4, pp. 182–202, 2024, https://www.researchgate.net/profile/Vybhav-Reddy-Kammireddy-Changalreddy/publication/388177379_Evolving_Fraud_Detection_Models_with_Simulated_and_Real-World_Financial_Data/links/678d5bf475d4ab477e4fc3f6/Evolving-Fraud-Detection-Models-with-Simulated-and-Real-World-Financial-Data.pdf.

[22] J. Zhou *et al.*, "Evaluating the Quality of Machine Learning Explanations: A Survey on Methods and Metrics.", *Electronics*, vol. 10, no. 5, 2021, https://doi.org/10.3390/electronics10050593.

[23] D. Varmedja, M. Karanovic, S. Sladojevic, M. Arsenovic and A. Anderla, "Credit Card Fraud Detection - Machine Learning methods," *2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pp. 1-5, 2019, https://doi.org/10.1109/INFOTEH.2019.8717766.

[24] T. Akilandeswari, D. Kamalesh, H. S. Bontha and D. M. Kumar, "Ensuring Secure Transactions through Cutting edge Machine Learning and Preprocessing Techniques in Credit Card Fraud Detection," *2024 Third International*

Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), pp. 1-5, 2024, https://doi.org/10.1109/ICEEICT61591.2024.10718456.

[25] S. Falkner, A. Klein, and F. Hutter, "Bohb: Robust and efficient hyperparameter optimization at scale," *arXiv*, 2018, https://arxiv.org/pdf/1807.01774.

[26] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Proceedings of Machine Learning Research*, vol. 9, pp. 249–256, 2010, http://proceedings.mlr.press/v9/glorot10a.html.

[27] E. Bisong, *Logistic Regression. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform*, Apress Berkeley, CA, 2019, https://doi.org/10.1007/978-1-4842-4470-8.

[28] F. J. J. Joseph, S. Nonsiri, and A. Monsakul, "Keras and TensorFlow: A hands-on experience," *Advanced deep learning for engineers and scientists*, pp. 85–111, 2021, https://doi.org/10.1007/978-3-030-66519-7_4.

[29] "Scikit-Learn site", Online - Last access 09 July 2024, https://scikit-learn.org.

[30] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and Tensorflow: Concepts, Tools, and Techniques to Build Intelligent Systems*, O'REILLY, 2019, https://books.google.co.id/books?id=X5ySEAAAQBAJ&hl=id&source=gbs_navlinks_s.

[31] B. W. G. van Rossum and N. Coghlan, "Pep 8 – style guide for python code," 2001, https://www.python.org/dev/peps/pep-0008/.

[32] Tim Peters, "The zen of python," 2004, https://www.python.org/dev/peps/pep-0020/#the-zen-of-python.

[33] R. B. Junior and J. Batista, *Overcoming Imbalanced Class Distribution and Overfitting in Financial Fraud Detection: An Investigation Using A Modified Form of K-Fold Cross Validation Approach to Reach Representativeness*, Doctoral dissertation, 2023, https://dspace.library.uvic.ca/handle/1828/15268.

[34] G. N. Ahmad, H. Fatima, S. Ullah, A. Salah Saidi and Imdadullah, "Efficient Medical Diagnosis of Human Heart Diseases Using Machine Learning Techniques With and Without GridSearchCV," in *IEEE Access*, vol. 10, pp. 80151-80173, 2022, https://doi.org/10.1109/ACCESS.2022.3165792.

[35] H. I. Fawaz *et al.*, "Deep learning for time series classification: a review," *Data Mining and Knowledge Discovery*, vol. 33, pp. 917–963, 2019, https://doi.org/10.1007/s10618-019-00619-1.

[36] T. Hagendorff and K. Meding, "Ethical considerations and statistical analysis of industry involvement in machine learning research," *AI & Society* vol. 38, pp. 35–45, 2023, https://doi.org/10.1007/s00146-021-01284-z.

BIOGRAPHY OF AUTHORS

Abdusy Syarif is an associate professor at Universitas Nasional, Jakarta, Indonesia. He received his Bachelor's degree in 1999, and his Master's degree from Universitas Indonesia in 2005. He obtained his PhD in 2015 from University of Haute Alsace, France. During 2016-2019, he was a Post-doc researcher at University of Haute-Alsace and LINEACT-CESI, France. He is the author/co-author of 1 intellectual property right, and more than 30 international publications in refereed journals and conferences, some of them achieved Best Paper Awards. His research interests involve Machine Learning, Artificial Intelligence, Multimedia Systems, Embedded Systems, Internet of Things, Routing Protocol, and Quality of Services. He has been serving as Technical Program Committee and Reviewer for some International Conferences and Journals since 2015.

Muhammad Haikal Satria received his Bachelor's degree in 2003 and his double Master's degree from Universitas Indonesia and University of Duisburg-Essen, Germany in 2006. He obtained his PhD in 2014 from Universiti Teknologi Malaysia. He is the author/co-author of over 50 international publications in refereed journals and conferences, 4 book sections. He has been working in Malaysia since 2007 on various projects. He obtained 7 gold medals, 4 bronze medals, and 12 patents for his inventions.

Hanenne Gabteni received her computer engineer degree in 2006 from University of Tunis El Manar, Tunisia. She obtained her Master's degree from University of Polytechnique Hauts-de-France in 2011 and her PhD in 2015 from University of Haute-Alsace, France. Currently, she is working in the Research and Innovation department at Astek Group, Paris, France. Her research focused on intelligent transportation systems and vehicular networks (VANETs). She specializes in mobility scenario design, and data fusion techniques.