Executive Summary of the Thesis

FraudBench: A Benchmarking Software for Fraud Detection Systems

Laurea Magistrale in Computer Science Engineering - Ingegneria Informatica

Author: Luca Maniscalchi

Advisor: Prof. Michele Carminati

Co-advisor: Tommaso Paladini

Academic year: 2022-2023

1. Introduction

In the digital age, online banking is an integral part of modern life. While it offers unprecedented convenience in managing our finances, it also presents new avenues for fraudulent activities, posing significant risks to both customers and financial institutions. Every day, millions of transactions take place, making manual review by humans impractical due to the time and resources required. To tackle this challenge, cybersecurity experts have developed automated Fraud Detection Systems (FDSs) that can process a large volume of transactions in real time. These systems utilise Machine Learning algorithms to identify and prevent fraudulent activities more efficiently and on a much larger scale than is possible through human intervention. Previous works have explored the application of Machine Learning techniques to the fraud detection domain [3, 9, 10]. However, despite the numerous challenges and ongoing developments in the fraud detection domain, the results proposed by these works are hard to compare properly. In particular, the described experiments achieve different goals, models are tested on different data sets, and the experimental source code is often not shared. A common experimental ground can be provided by benchmarking tools, which guide the construction, evaluation and comparison of experimental approaches. Related solutions have been proposed in the area of Adversarial Machine Learning (AML) [7, 8], but, to the best of our knowledge, none have been proposed in this search area.
This work introduces FraudBench, an open-source framework specifically designed to serve as a platform for developing, evaluating, and benchmarking FDSs. Unlike existing solutions, FraudBench aims to provide a holistic approach, covering every aspect of the development process, from data preprocessing to model evaluation and performance benchmarking. By doing so, it seeks to standardise the fraud detection process, making the results more replicable and comparable across different studies.

2. Background and Related Works

State-of-the-art Fraud Detection Systems use Machine Learning algorithms to automate the detection process effectively. Machine Learning, a subset of artificial intelligence, develops models that improve over time by learning from data. These algorithms are typically categorised into three main types: supervised learning, unsupervised learning, and reinforcement learning.

Supervised learning uses labeled data for training, unsupervised learning works with unlabeled data to find patterns, and reinforcement learning trains algorithms to make decisions based on environmental feedback to maximise rewards. In addition, we can also find hybrid techniques, such as active earning, which uses both labeled and unlabeled data for training. Ensemble models can enhance performance by combining different Machine Learning approaches, making the system more robust and improving the overall performance. In the current literature, several research papers about fraud detection offer ad hoc benchmarks of Machine Learning algorithms. For instance, Cheng et al. [5] propose a fraud detection framework based on Convolutional Neural Networks (CNNs) to recognise inherent behavioural patterns of fraud from annotated data. Their experiments on a dataset containing over 260 million credit card transactions from a major commercial bank show that the proposed approach outperforms some state-of-the-art methods.
N. E-Arefin [4], on the other hand, explores the performance of several types of classification algorithms in the context of credit card fraud detection. These include Bayesian classifiers, function-based classifiers, lazy algorithms, meta-algorithms, rule-based classifiers, and tree-based algorithms. The experimental findings indicate that meta and tree classifiers perform better than other types of classifiers.
Similarly, G. K. Kulatilleke [6] conducts a study on credit card fraud detection and proposes a data-driven approach to dynamically select an appropriate model based on evaluation metric scores and balancing strategies. His framework creates a collection of 530 classifiers by combining different models, sampling strategies, and feature selection techniques. He then evaluates these classifiers using eight performance metrics. These papers offer valuable reference points; however, their methods cannot be directly compared due to the absence of a common benchmarking framework. This limitation makes the cross-comparison of methods challenging and hinders the cumulative progress in fraud detection research. In the literature, we can find publicly available frameworks for evaluating Machine Learning algorithms, but they are often geared toward testing resilience against adversarial attacks and mainly focus on image data. Some researchers have exploited these libraries in their process of model evaluation [2, 11], but they had to apply substantial modifications to them.

3. Motivation and Goals

To the best of our knowledge, there is no publicly available framework designed to benchmark Machine Learning models in the fraud detection field. In light of this limitation, we introduce FraudBench, a dedicated framework explicitly tailored for fraud detection tasks. This novel framework aims to provide researchers with a valuable tool for evaluating and analysing the performance and capabilities of diverse Machine Learning models. By offering a more thorough understanding of their effectiveness, it enables researchers to make informed decisions regarding their applicability in real-world fraud detection scenarios.
In this work, we aim to achieve a set of specific goals:

- Design a versatile, modular and easily extendible framework to facilitate the development and testing of FDSs. The framework must be flexible and highly configurable, allowing researchers to tailor the execution process to their unique needs and preferences. Furthermore, it has to be modular so that different components can be modified or replaced without affecting the entire system, facilitating the integration of new detection techniques or updates in the future.
- Use the framework to investigate the behaviour of FDSs when subjected to different types of fraud attacks, providing valuable insights and analysis.

4. Threat Model

Our work is based on the fundamental assumption that attackers possess the capability to carry out transactions on behalf of unsuspecting victims. This assumption acknowledges the existence of various techniques that enable such unauthorised actions [1]. We identify and explore two principal types of fraudulent activities:

1. **Information Stealing**. With this term, we refer to an attacker who has already obtained a user's credentials. With these

stolen credentials, the attacker gains unauthorised access to the user's bank account and, once there, can perform transactions and redirect funds to the preferred location. In this particular scenario, the attacker executes the transaction from his or her device.

2. **Transaction Hijacking**. In this scenario, the attacker compromises the victim's device (typically through the installation of malware) to gain control over their mobile or browser-based banking application. When the user initiates a transaction, the malware intercepts and hijacks it, making the user believe the legitimate transaction has succeeded. In this particular scenario, the attacker executes the transaction from from the victim's device.

5. Dataset Analysis

Acquiring datasets from financial institutions presents a significant challenge due to confidentiality considerations. However, we have the privilege of having access to an anonymised real-world labelled dataset provided to our research group by a major Italian bank. The dataset covers 125 days, specifically from October 22, 2014, to February 23, 2015. Out of a total of 471,787 transactions, only 0.124% are classified as fraudulent. Each transaction of the extracted dataset is characterised by 32 features. In Table 1, we briefly describe the most relevant.

Feature Name	Description
TransactionID	Unique identifier of the transaction
IP	The IP address of the user's connection
SessionID	Value (assigned by the online banking platform) that identifies the session
Timestamp	Date and time at which the transaction is executed
Amount	Transaction amount in euros
ErrorMsg	Error message (in case of a transaction that was not correctly executed)
UserID	Unique identifier assigned to the user
IBAN	Beneficiary account number
Confirm_SMS	Flag that indicates whether the transaction required a confirmation code to be completed
IBAN_CC	Country code of the beneficiary IBAN
CC_ASN	Country code of the Autonomous System Number associated with the connection of the device from which the transaction is performed
Fraud	Flag that indicates whether the transaction is fraudulent

Table 1: Most relevant dataset features

6. Approach

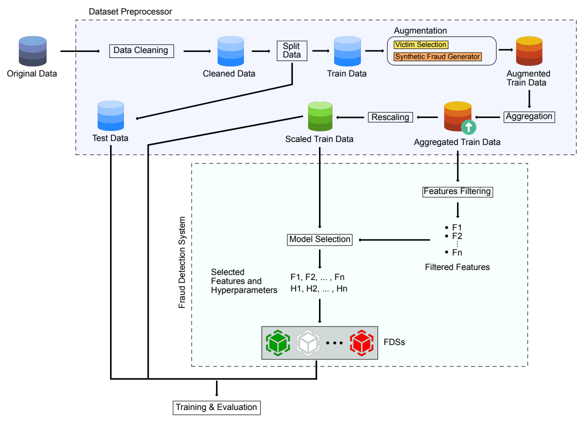

Figure 1: Framework logical components

Our framework consists of two primary modules. The first is the **Dataset Preprocessor** module responsible for data preprocessing tasks like removing irrelevant features, splitting data into training and test sets, feature aggregation, scaling, and data augmentation. The second is the **Fraud Detection System** module, which focuses on selecting the most informative features, optimising hyperparameters for detection algorithms, and creating the final detection systems. Completing the framework is the module dedicated to the performance evaluation of the resulting detection systems. Figure 1 shows the complete architecture with all the subcomponents detailed in the following sections.

6.1. Dataset Preprocessor Module

Data cleaning. Transactions in our dataset are characterised by 32 features. However, some of them do not provide any meaningful information. We remove empty, incomplete, and non-informational fields. Additionally, we filter out duplicates and transactions incorrectly processed because of an error.

Dataset augmentation. One of the main challenges we face in the fraud detection context is the extremely low percentage of fraudulent transactions within the dataset. To address this problem, we provide an augmentation module that incorporates a synthetic fraud generator. Using this generator, we can replicate realistic fraudulent transactions and inject them inside the dataset, thereby enhancing it and achieving the targeted proportion of fraudulent transactions in relation to legitimate ones. The synthetic fraud crafting process starts with the victims selection. We categorise the users into three distinct profiles based on the average amount and number of performed transactions. We show the characteristics of each profile in Table 2. After the selection of potential victims, we proceed with the synthesis of fraudulent transactions. According to the threat model we defined in Chapter 4, with this generator we synthesise two fraud schemes: Information Stealing and Transaction Hijacking. During the generation process, we assign values to the fields that compose the transaction based on the characteristics of the scheme we simulate. Regarding the `Amount` of the fraudulent transactions, we provide the generator with a range of values within which it will pick the transaction amount. As in the case of victims, we define three profiles: High, Medium and Low. We provide the amount range for each profile in Table 3. Depending on what we want to achieve, we can use the synthetic fraud generator to inject either frauds belonging to a single profile only or different profiles.

Victim Profile	Amount Mean (€)	Transaction volume
High	> 3,000	> 35
Medium	1,500 - 2,999	15 - 34
Low	0 - 1,499	5 - 14

Table 2: Victims profiles

Fraud Profile	Amount Range (€)
High	30,000 - 50,000
Medium	2,000 - 5,000
Low	100 - 1,000

Table 3: Fraud profiles

Transaction aggregation. In banking datasets, each transaction is characterised by specific features that provide detailed information about it. However, to build a powerful model for fraud detection, it is necessary to train Machine Learning algorithms on a dataset that encompasses a broader range of relevant information. Original features alone are insufficient because they do not capture the complete picture of a user's spending patterns and behaviour over a certain period. Therefore, as we illustrate in Figure 1, we employ transaction aggregation to create a more holistic view of each user's activities. This process combines information from multiple transactions to generate new features that capture more relevant details from the original data, like the average amount spent within a specific timeframe. This additional data provides insights into consumers' spending habits, supporting the detection of anomalous behaviours that significantly diverge from their established norms. Initially, we calculate the aggregated features for legitimate transactions to assess the regular spending patterns of users. Next, we compute the aggregated features for frauds, which necessitate considering the previous transactions of the victims. So, we merge legitimate and fraudulent transactions and we apply the computation.

Features scaling. This is a preprocessing technique used in Machine Learning to standardise or normalise the numerical features of a dataset. It is essential because many Machine Learning algorithms base their calculations and derivations on the Euclidean distance, which is sensitive to the scale of input features. The primary purpose of feature scaling is to bring all the features to a common scale so that they all contribute equally to the learning process. Common features scaling methods include Min-Max scaling and Standardization. The former technique scales the features to a specific range, typically between 0 and 1. It transforms the data by subtracting the minimum value and dividing it by the range (maximum value minus minimum value). The latter, instead, rescales the features to have a mean of 0 and a standard deviation of 1 by subtracting the mean and dividing by the standard deviation of each feature. Due to the specific characteristics of banking datasets, we choose standardisation over Min-Max scaling as the feature scaling technique. Transaction amounts can exhibit a wide variance, ranging from very low amounts (e.g., 0.01 €) to very high amounts (e.g., 50000 €). Moreover, banking datasets typically comprise a larger number of transactions with smaller amounts and only a few with exceptionally high amounts. Given that, using Min-Max scaling could lead to a disproportionate compression of lower values towards 0. Consequently, the influence of transactions with lower amounts would be severely diminished, making them virtually indistinguishable.

6.2. Fraud Detection System Module

Within FraudBench, we include the implementation of six commonly utilised algorithms in the literature for fraud detection: Logistic Regression (LR), Support Vector Machine (SVM), Neural Network (NN), Extreme Gradient Boosting (XGB), Random Forest (RF) and a variant of Active Learning (AL). Additionally, we also include two ensemble models based on two different approaches: Majority Voting (MV) and Multiplicative Weight Update (MWU). In this section, we detail the steps that, starting from the outputs of the Dataset Preprocessor module, lead to the development of the final FDSs.

Features filtering. The aggregation process we presented in Section 6.1 produces several potential features, but not all of them are equally valuable. Our goal is to identify the most informative and discriminative features that contribute significantly to the model's predictive power while discarding irrelevant or redundant ones. The feature filtering component aims to perform a preliminary feature skimming. Our feature filtering process is model-independent and uses a correlation-based method. As illustrated in Figure 1, we start with the aggregated dataset, we compute the correlation matrix to understand how features are related, and based on those values, we select the most important ones. To do this, we analyse one pair of features at a time, and if these two are highly correlated among themselves (more than 95% in our case), we remove the one that is less correlated with the target.

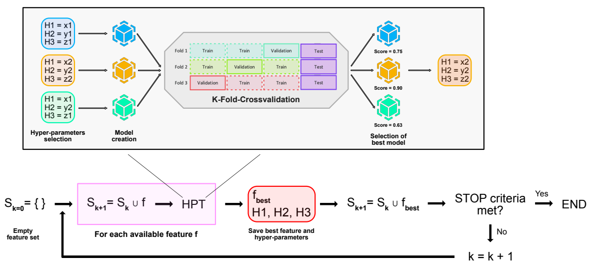

Figure 2: Model selection scheme

Model selection. In the Model Selection component of FraudBench, we use a technique that simultaneously selects the best feature set and hyperparameters for each detection system. As Figure 2 illustrates, we employ a forward selection method that starts with an empty feature set and iteratively adds the most beneficial features. We use as a set of available features the one resulting from the filtering technique described in the previous step, and we assess the contribution of each feature (when added to the final feature set) using a set of m models and k-Fold-Crossvalidation (with m and k configurable parameter). We determine the models' hyperparameters via a Random Search approach on a predefined grid. After each iteration, we choose the best-performing feature in terms of a selectable metric (F1-score in our case) and its corresponding optimal hyperparameters, and we store the results. We focus on optimising the F1-score rather than accuracy due to the unbalanced nature of fraud datasets. We then integrate the selected feature into the final feature set, and the process iterates again. Our method has a high computational cost, so we allow a limit on the number of attempts to add new features. If a new feature doesn't improve the model's performance beyond a certain point, the process stops to save computational resources.

6.3. Training and Evaluation Module

Once we select the final set of features and hyperparameters and we instantiate the final Fraud Detection Systems, we proceed with the ultimate training and evaluation. In the context of banking transactions, statistical properties and underlying concepts of the target variable or feature distribution can change over time, affecting model performance. This is a highly significant phenomenon known as concept drift. To mitigate the impact of concept drift, in this work, during the detection systems' training process, we adopt a weighted approach that assigns varying importance to samples based on their temporal occurrence. Specifically, we utilise a decreasing exponential function that accords greater weightage to more recent transactions:

$$weights = e^{-\frac{t}{k}}$$

The parameter t denotes the time interval, measured in hours, between the training timestamp and the timestamp of the transactions under consideration. Meanwhile, the parameter k governs the rate at which the weights diminish as time progresses. To assess the performance of the FDSs, we evaluate them by processing one week of transactions at a time, starting from the week following the training time. These transactions comprise both legitimate and fraudulent instances, where we synthetically generate the fraudulent ones using the fraud generator we detailed in Section 6.1. Detection systems are evaluated both in terms of standard metrics and a custom loss function described in Section 7.1.

7. Experimental Validation

7.1. Evaluation Metric

While the primary goal of a FDS is to detect and prevent as many fraudulent transactions as possible, it is essential to establish a threshold for the acceptable number of false positives. We assess the impact of frauds on each model by employing a custom loss function that takes into account both false negatives and false positives. This specialised loss function assigns a specific monetary value to each type of mispredicted transaction. In particular, we estimated a cost for the bank institution of 100 € in case of a false positive, and in the case of a false negative, the loss is equal to the actual amount of the transaction.

$$Loss = (true_{label} - predicted_{label})^2 * \\ (100 * predicted_{label} + Amount * true_{label})$$

7.2. Experimental Settings

The primary objective of our experimental validation is to conduct a comprehensive performance evaluation of multiple FDSs based on different algorithms and under different training settings. To explore the impact of different fraudulent activities on these models, we adopt a systematic approach. Starting from a common dataset, we augment it by introducing two distinct schemes of fraud: Information Stealing (IS) and Transaction Hijacking (TH). We further subdivide each fraud scheme into three fraud profiles: Low (L), Medium (M), and High (H). This categorisation leads to the creation of six unique datasets, each representing a specific combination of fraud scheme and profile. To optimise the performance and efficiency of our models, we employ a two-step process for each dataset. Initially, we perform feature reduction by evaluating the correlation among the features, resulting in a refined set of features specific to each dataset. Then, we fine-tune the models by applying the model selection procedure described in Section 6.2 to each model-dataset combination. As a result of this approach, we develop 36 distinct models corresponding to the Cartesian product of the six datasets and the six model types. We expand our analysis by including an additional set of 6 models. These models undergo the same procedure as the other 36 models, following the same methodology. However, in this case, we train them on a comprehensive dataset that includes instances of all types of fraud. We state models performances by testing their resistance to three types of attacks, in which we generate frauds each time following a different policy. In addition to the range of individual models, we explore two ensemble models based on MV and MWU. In their prediction process, the latter combine the outcomes of all 42 individual models. We compare the ensemble models against a baseline model which is the empirical mean of the losses of the base FDS learners. In all attacks scenarios, the loss is calculated according to the metric defined in Section 7.1.

7.3. Attacks

Each attack is designed to evaluate the performance of the models under different scenarios. We inject fraudulent transactions on a weekly basis, and at the end of each week, we calculate the estimated loss. To identify the classifiers, we adopt a naming convention where we formulate the name by combining the acronym of *base model*, *fraud profile*, and *fraud scheme* associated with the system. For instance, if we consider a detection system that uses Random Forest as its base model and has been trained to identify High-profile Information Stealing frauds, we name the system as `RF_H_IS`.

7.3.1. Attack 1

In this experiment, we simulate an attacker who commits fraudulent transactions by following a random policy, that is, randomly choosing fraud schemes and profiles. In the long run, the detection system based on the random forest model and trained on all types of fraud (`RF_AF`) performs the best, with an overall loss of 1,052,771.86 € and an F1-score of approximately 80%. Its low rate of false positives (0.08%) likely contributed to its overall effectiveness. The model with the highest number of detected frauds is the SVM trained on frauds belonging to the Information Stealing scheme and having a High profile (`SVM_H_IS`), which scores a recall of 84.3%. However, it has a very low precision (13.9%), resulting in a high false positive rate (2.6%) and a greater loss (2,373,895.89 €) when compared to the previous model. Almost all systems trained exclusively on High-profile Information Stealing frauds generally perform well in minimising monetary loss. This was expected because this type of fraud has the highest financial impact. Regarding ensemble models, the `MV` ensemble achieves a final loss of 7,270,732.92 €, which represents a 49% reduction when compared to the average loss of the individual models. This ensemble is highly precise (95%) but has low recall (37.6%). On the other hand, `MWU` exhibits a slightly lower precision (88%) but, at the price of an increased false positive rate, can identify a larger number of frauds, reaching a recall of 76.5% and a final loss of 1,135,369.99 €, a 92% reduction compared to the average loss.

7.3.2. Attack 2

In this experiment, we simulate attackers who commit fraudulent transactions according to the policy that causes the greatest impact on the best-performing model of every week. Except for the first week, during which we randomly select the type of fraud to inject, for each subsequent week, we generate a set of fraudulent transactions for each type of fraud. We then evaluate their impact on the model that shows the best performance up until that point, and we select the fraud type that resulted in the greater increase in the estimated loss. Surprisingly, the SVM trained on all fraud types (`SVM_AF`) emerges as the best-performing model in the long run. Despite its simplicity, this model manages to maintain an overall estimated loss just below 5 million euros (4,977,643.33 €). However, this detection system has a very low precision (14.4%) and a high false positive rate (∼2%). The detection system with the highest F1-score (84.1%) is `RF_AF`. When comparing its standard metric values with those of the overall best system, `RF_AF` outperforms it in every aspect. Nevertheless, it does not prove to be the best when we evaluate it using our specific loss function. When analysing the false negatives amounts of `SVM_AF` and `RF_AF`, we observe that, on average, the latter's false negatives amounts are approximately three times higher (26,886.44 € vs 1,656.15 €). This explains why, at the end of the period, the loss of `RF_AF` is higher. Regarding the ensemble models, `MWU` once again demonstrates superior performance with respect to `MV`, highlighting the effectiveness of an online learning approach in this context.

7.3.3. Attack 3

In this experiment, we systematically apply all fraud types cyclically to examine the impact of each type on the detection systems we analyse. Unsurprisingly, High-profile Information Stealing frauds are the ones that contribute the most to the increase in loss for most of the models, with an average increase of 5,688,022.48 € compared to the week before their introduction. Following this, we find High-profile Transaction Hijacking frauds that cause an average increase of 1,698,591.52 €, Medium-profile Information Stealing frauds that lead to an average increase of 672,657.10 €, Medium-profile Transaction Hijacking frauds that results in an average increase of 280,355.32 €, Low-profile Information Stealing frauds that contributes an average increase of 151,758.85 €, and finally, Low-profile Transaction Hijacking frauds that leads to an average increase of 122,322.37 €. As we show in Figure 3, the number of models that can correctly identify High-profile Information Stealing frauds, on average, is higher with respect to other types of fraud. However, given the high amounts associated with this fraudulent transaction profile, it is unsurprising to see their significant impact on the overall losses when not detected.

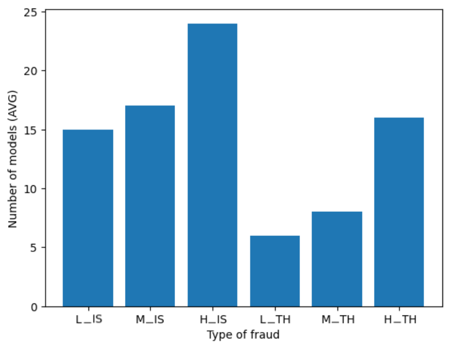

Figure 3: Average number of models able to correctly identify a fraudulent transaction belonging to a specific type

8. Conclusions

In this work, we introduced FraudBench, a publicly available, versatile, and highly customisable framework to evaluate a range of Fraud Detection Systems in the banking and financial sectors. Thanks to its modular design, various components can be easily modified, replaced, or extended to suit different research purposes. Moreover, its customisable nature allows researchers to fine-tune parameters, adjust configurations, and incorporate domain-specific knowledge. We validated our framework through an experimental evaluation on a dataset from an Italian banking group, allowing us to obtain experimental results demonstrating real-world applicability. Our results showed that a a higher model complexity does not always guarantee higher performance. Contrary to expectations, we consistently observed that simpler detection systems based on Support Vector Machines and Logistic Regression can achieve comparable or, sometimes, even better performance compared to more complex models. Secondly, since in the banking and financial context, false positives and false negatives have distinct impacts in terms of monetary loss, commonly used evaluation metrics alone may not provide an accurate assessment of a detection system performance. Our experiments also highlighted the importance of carefully considering the choice of ensemble techniques. Surprisingly, the Majority Voting approach did not prove to be an effective alternative in minimising losses. In contrast, a more informed strategy, such as the Multiplicative Weight Update ensemble technique, demonstrated its ability to dynamically adapt and learn from the data, resulting in superior performance. Future works include the integration of new Machine Learning models and ensemble strategies, additional feature selection and model selection procedures, and the extension of the framework to allow testing models against adversarial attacks.

References

[1] A Alsayed and Anwar Bilgrami. E-banking security: Internet hacking, phishing attacks, analysis and prevention of fraudulent activities. *International Journal of Emerging Technology and advanced engineering*, 7(1):109–115, 2017.

[2] Francesco Cartella, Orlando Anunciacao, Yuki Funabiki, Daisuke Yamaguchi, Toru Akishita, and Olivier Elshocht. Adversarial attacks for tabular data: Application to fraud detection and imbalanced data, 2021.

[3] Sahil Dhankhad, Emad Mohammed, and Behrouz Far. Supervised machine learning algorithms for credit card fraudulent transaction detection: A comparative study. In *2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pages 122–125, July 2018.

[4] Md. Nur E-Arefin. A comparative study of machine learning classifiers for credit card fraud detection. *International Journal of Innovative Technology and Interdisciplinary Sciences*, 3(1):395–406, Mar. 2020.

[5] Kang Fu, Dawei Cheng, Yi Tu, and Liqing Zhang. Credit card fraud detection using convolutional neural networks. In Akira Hirose, Seiichi Ozawa, Kenji Doya, Kazushi Ikeda, Minho Lee, and Derong Liu, editors, *Neural Information Processing*, pages 483–490, Cham, 2016. Springer International Publishing.

[6] Gayan K. Kulatilleke. Credit card fraud detection - classifier selection strategy, 2022.

[7] Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian M. Molloy, and Ben Edwards. Adversarial robustness toolbox v1.0.0, 2019.

[8] Nicolas Papernot, Fartash Faghri, Nicholas Carlini, Ian Goodfellow, Reuben Feinman, Alexey Kurakin, Cihang Xie, Yash Sharma, Tom Brown, Aurko Roy, Alexander Matyasko, Vahid Behzadan, Karen Hambardzumyan, Zhishuai Zhang, Yi-Lin Juang, Zhi Li, Ryan Sheatsley, Abhibhav Garg, Jonathan Uesato, Willi Gierke, Yinpeng Dong, David Berthelot, Paul Hendricks, Jonas Rauber, Rujun Long, and Patrick McDaniel. Technical report on the cleverhans v2.1.0 adversarial examples library, 2018.

[9] Abhimanyu Roy, Jingyi Sun, Robert Mahoney, Loreto Alonzi, Stephen Adams, and Peter Beling. Deep learning detecting fraud in credit card transactions. In *2018 Systems and Information Engineering Design Symposium (SIEDS)*, pages 129–134, 2018.

[10] Dejan Varmedja, Mirjana Karanovic, Srdjan Sladojevic, Marko Arsenovic, and Andras Anderla. Credit card fraud detection - machine learning methods. In *2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pages 1–5, 2019.

[11] Ping Wang, Qianyu Wang, Yuting Zhang, and Yifan Wu. Defense mechanism against adversarial attacks based on chaotic map encryption. *Journal of Physics: Conference Series*, 2037(1):012025, sep 2021.