Reliable Logistic Regression for Credit Card Fraud Detection

Yassine Hmidy, Mouna Ben Mabrouk
Sogetilabs at Capgemini, Paris, France

***Abstract*—Credit card fraud poses a significant threat to financial institutions and consumers worldwide, necessitating robust and reliable detection methods. Traditional classification models often struggle with the challenges of imbalanced datasets, noise, and outliers inherent in transaction data. This paper introduces a novel fraud detection approach based on a discrete non-additive integral with respect to a non-monotonic set function. This method not only enhances classification performance but also provides an interval-valued output that serves as an index of reliability for each prediction. The width of this interval correlates with the prediction error, offering valuable insights into the confidence of the classification results. Such an index is crucial in high-stakes scenarios where misclassifications can have severe consequences. The model is validated through extensive experiments on credit card transaction datasets, demonstrating its effectiveness in handling imbalanced data and its superiority over traditional models in terms of accuracy and reliability assessment. However, potential challenges such as increased computational complexity and the need for careful parameter tuning may affect scalability and real-time implementation. Addressing these challenges could further enhance the practical applicability of the proposed method in fraud detection systems.**

***Keywords*—Credit card fraud; fraud detection; computational complexity**

I. Introduction

In today's digital economy, credit card transactions have become ubiquitous due to their convenience and global acceptance. However, the rise in electronic transactions has been paralleled by an increase in fraudulent activities, posing significant challenges to financial institutions and consumers alike. Credit card fraud not only results in substantial financial losses but also undermines customer trust and the integrity of payment systems [9]. The increasing prevalence of online transactions and the convenience of credit card payments have created opportunities for fraudsters to exploit vulnerabilities in financial systems [1], [10]. Global fraud losses reached £21.84 billion in 2015 alone, emphasizing the urgent need for effective fraud detection mechanisms [14].

A. Challenges in Credit Card Fraud Detection

Detecting fraudulent transactions is a complex task due to several inherent challenges:

- **Imbalanced Datasets**: Fraudulent transactions represent a minute fraction of the total transaction volume, leading to highly imbalanced datasets. This imbalance poses significant difficulties for machine learning models, which may become biased toward the majority class and fail to detect fraudulent activities effectively [12], [14].
- **Data Noise and Outliers**: Transaction data often contain noise and outliers, which can adversely affect the performance of detection algorithms, resulting in increased false positives and negatives [13].
- **Concept Drift**: The strategies employed by fraudsters continuously evolve, causing changes in the underlying data distribution—a phenomenon known as concept drift. Models must adapt over time to maintain their effectiveness in detecting new fraud patterns [12].

B. Existing Approaches and Limitations

Various machine learning techniques have been applied to address credit card fraud detection:

- **Statistical Methods**: Logistic regression has been widely used due to its simplicity and interpretability, modeling the probability of fraudulent transactions based on historical data [3]. However, logistic regression may struggle with nonlinear relationships and is sensitive to data imbalance, potentially limiting its effectiveness in fraud detection scenarios [3].
- **Machine Learning Algorithms**: Supervised learning algorithms such as support vector machines [8], random forests [6], and ensemble methods like AdaBoost and majority voting [4] have demonstrated improved detection rates by capturing complex patterns in the data.
- **Deep Learning Techniques**: Deep learning approaches, including autoencoders and restricted Boltzmann machines, have been employed to detect anomalies and reconstruct input data for identifying fraudulent transactions [2], [5].
- **Aggregation Strategies**: Strategies incorporating aggregation mechanisms and feedback loops aim to enhance the adaptability and accuracy of fraud detection systems [1].

Despite these advancements, a critical limitation remains: many existing models focus primarily on maximizing classification accuracy without providing a measure of confidence or reliability for individual predictions [11]. In high-stakes environments like fraud detection, misclassifying a legitimate transaction as fraudulent can lead to customer dissatisfaction and loss of trust, while failing to detect actual fraud results in financial losses and potential legal implications [10]. Therefore, there is a need for models that not only improve detection rates but also quantify the uncertainty associated with each prediction.

C. Main Contributions

This paper proposes a novel approach to credit card fraud detection using a discrete Choquet integral with respect to a non-monotonic set function, specifically leveraging the MacSum aggregation model. This method outputs an interval-valued prediction for each transaction, where the width of the interval correlates with the prediction error. This interval serves as an index of reliability, providing valuable insights into the confidence level of each classification decision.

The contributions can be summarized as follows:

- This paper introduce the application of the MacSum aggregation model within the discrete Choquet integral framework to the problem of credit card fraud detection, offering interval-valued outputs that reflect prediction reliability.
- This paper addresses the challenges of data imbalance and concept drift by incorporating robust preprocessing techniques and adaptive mechanisms within the model.
- The approach is validated on benchmark datasets, comparing its performance against traditional classifiers, including logistic regression and state-of-the-art methods, demonstrating improved accuracy and reliability assessment.

II. Preliminaries

This section introduces the fundamental concepts, notations, and definitions necessary to understand the proposed model. It provides an overview of set functions, the discrete Choquet integral which are needed to compute the MacSum aggregation.

A. Notations and Definitions

- $\Omega = \{1, \ldots, N\} \subset \mathbb{N}$: a finite index set.
- For all $A \subseteq \Omega$, A^c denotes the complement of A in Ω, i.e., $A^c = \Omega \setminus A$.
- $\mathbb{R}$: the set of real numbers.
- A vector is a function $\boldsymbol{x} : \Omega \to \mathbb{R}$, defined by a discrete subset of $\mathbb{R}^N$, denoted $\boldsymbol{x} = (x_1, \ldots, x_N) \in \mathbb{R}^N$.
- $\overline{\underline{x}} = [\underline{x}, \overline{x}]$: a real interval where $\underline{x} \in \mathbb{R}$ is the lower bound and $\overline{x} \in \mathbb{R}$ is the upper bound.
- $\mathbb{IR}$: the set of real intervals.
- A *set function* is a function $\mu : 2^\Omega \to \mathbb{R}$ that assigns a real value to any subset of Ω. The *complementary set function* μ^c associated with μ is defined by:

$$\mu^c(A) = \mu(\Omega) - \mu(A^c), \quad \forall A \subseteq \Omega. \tag{1}$$

  Usually, it is assumed that $\mu(\emptyset) = 0$, where $\emptyset$ is the empty set of Ω.

- A set function μ is said to be *submodular* if, for all $A, B \subseteq \Omega$, the following inequality holds:

$$\mu(A \cup B) + \mu(A \cap B) \leq \mu(A) + \mu(B). \tag{2}$$

- A set function μ is said to be *additive* if, for all $A, B \subseteq \Omega$, it holds that:

$$\mu(A \cup B) + \mu(A \cap B) = \mu(A) + \mu(B). \tag{3}$$

- If a set function μ is submodular, then its complementary μ^c is supermodular.

B. Discrete Choquet Integral

Classical integration theory involves additive measures. However, in many real-world applications, especially in decision making, the assumption of additivity does not hold due to interactions among criteria. Non-additive integrals provide a framework for integrating functions with respect to non-additive set functions (also known as capacities or fuzzy measures) [16].

A *non-additive integral* is an integral where the underlying set function is not necessarily additive. This allows for modeling situations where the whole is not simply the sum of its parts, capturing phenomena such as synergy, redundancy, and interactions among elements.

Among the most widely used non-additive integrals are the Choquet integral and the Sugeno integral. This work, focuses on the discrete Choquet integral due to its ability to model the aggregation of information while accounting for the interactions among criteria.
The discrete Choquet integral with respect to a set function μ is denoted $\check{\mathbb{C}}_\mu$ [15] and is defined for any vector $\boldsymbol{x} \in \mathbb{R}^N$ by:

$$\check{\mathbb{C}}_\mu(\boldsymbol{x}) = \sum_{k=1}^{N} x_{(k)} \left(\mu(A_{(k)}) - \mu(A_{(k+1)}) \right), \tag{4}$$

where:

- $(\cdot)$ denotes the permutation that sorts the elements of $\boldsymbol{x}$ in increasing order:

$$x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(N)}. \tag{5}$$

- $A_{(k)}$ ($k \in \{1, \ldots, N\}$) are the subsets (also called coalitions) defined by:

$$A_{(k)} = \{(k), \ldots, (N)\}, \quad A_{(N+1)} = \emptyset. \tag{6}$$

  Here, (k) indicates the index corresponding to the k-th smallest element in the sorted vector.

If a set function μ is submodular, then for all $\boldsymbol{x} \in \mathbb{R}^N$, it holds that [16]:

$$\check{\mathbb{C}}_\mu(\boldsymbol{x}) \geq \check{\mathbb{C}}_{\mu^c}(\boldsymbol{x}). \tag{7}$$

C. MacSum Aggregation

Let $\boldsymbol{\theta} \in \mathbb{R}^N$ be a parameter vector used in the aggregation.

1) Definition of the MacSum Set Functions: The MacSum set function $\nu_{\boldsymbol{\theta}}$ and its complementary set function $\nu^c_{\boldsymbol{\theta}}$ are defined as follows [17]:

For all $A \subseteq \Omega$,

$$\nu_{\boldsymbol{\theta}}(A) = \max_{i \in A} \theta_i^+ + \min_{i \in \Omega} \theta_i^- - \min_{i \in A^c} \theta_i^-, \tag{8}$$

$$\nu^c_{\boldsymbol{\theta}}(A) = \min_{i \in A} \theta_i^- + \max_{i \in \Omega} \theta_i^+ - \max_{i \in A^c} \theta_i^+, \tag{9}$$

with, for all $i \in \Omega$:

$$\theta_i^+ = \max(0, \theta_i), \quad \theta_i^- = \min(0, \theta_i). \tag{10}$$

The MacSum set function $\nu_{\boldsymbol{\theta}}$ is a parametric set function that is submodular [17]. Therefore, its corresponding Choquet integral satisfies:

$$\check{\mathbb{C}}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x}) \geq \check{\mathbb{C}}_{\nu_{\boldsymbol{\theta}}^c}(\boldsymbol{x}), \quad \forall \boldsymbol{x} \in \mathbb{R}^N. \tag{11}$$

Using the MacSum set functions, the MacSum aggregation $\mathcal{A}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x})$ for any vector $\boldsymbol{x} \in \mathbb{R}^N$ is defined as:

$$\mathcal{A}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x}) = [\check{\mathbb{C}}_{\nu_{\boldsymbol{\theta}}^c}(\boldsymbol{x}), \check{\mathbb{C}}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x})]. \tag{12}$$

This means that the MacSum aggregation produces an interval-valued output, where:

- $\underline{y} = \check{\mathbb{C}}_{\nu_{\boldsymbol{\theta}}^c}(\boldsymbol{x})$ is the lower bound of the aggregation.
- $\overline{y} = \check{\mathbb{C}}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x})$ is the upper bound of the aggregation.

2) Relationship with Linear Aggregations: Let $\boldsymbol{\psi} \in \mathbb{R}^N$. The linear parametric set function $\lambda_{\boldsymbol{\psi}}$ is defined as:

$$\lambda_{\boldsymbol{\psi}}(A) = \sum_{i \in A} \psi_i, \quad \forall A \subseteq \Omega. \tag{13}$$

An important property of the MacSum aggregation is that it dominates a set of linear parametric set functions [17]. Specifically, the set of all parameter vectors $\boldsymbol{\psi}$ such that $\lambda_{\boldsymbol{\psi}}$ is dominated by the MacSum set functions with respect to the parameter $\boldsymbol{\theta}$ is:

$$\mathcal{M}(\boldsymbol{\theta}) = \{\boldsymbol{\psi} \in \mathbb{R}^N \mid \forall A \subseteq \Omega, \\ \nu_{\boldsymbol{\theta}}^c(A) \leq \lambda_{\boldsymbol{\psi}}(A) \leq \nu_{\boldsymbol{\theta}}(A)\}.$$

This set $\mathcal{M}(\boldsymbol{\theta})$ is convex [17], which means that for any $\boldsymbol{\psi}_1, \boldsymbol{\psi}_2 \in \mathcal{M}(\boldsymbol{\theta})$ and any $\gamma \in [0, 1]$, the combination $\gamma\boldsymbol{\psi}_1 + (1-\gamma)\boldsymbol{\psi}_2$ also belongs to $\mathcal{M}(\boldsymbol{\theta})$.

The linear aggregation associated with $\lambda_{\boldsymbol{\psi}}$ is given by:

$$\mathcal{A}_{\lambda_{\boldsymbol{\psi}}}(\boldsymbol{x}) = \check{\mathbb{C}}_{\lambda_{\boldsymbol{\psi}}}(\boldsymbol{x}) = \sum_{i \in \Omega} \psi_i \cdot x_i. \tag{14}$$

Therefore, the MacSum aggregation can be interpreted as:

$$\mathcal{A}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x}) = \left\{\mathcal{A}_{\lambda_{\boldsymbol{\psi}}}(\boldsymbol{x}) \;\middle|\; \boldsymbol{\psi} \in \mathcal{M}(\boldsymbol{\theta})\right\} \tag{15}$$

$$= \left[\underline{\mathcal{A}}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x}),\ \overline{\mathcal{A}}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x})\right], \tag{16}$$

with:

$$\underline{\mathcal{A}}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x}) = \min_{\boldsymbol{\psi} \in \mathcal{M}(\boldsymbol{\theta})} \mathcal{A}_{\lambda_{\boldsymbol{\psi}}}(\boldsymbol{x}),$$
$$\overline{\mathcal{A}}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x}) = \max_{\boldsymbol{\psi} \in \mathcal{M}(\boldsymbol{\theta})} \mathcal{A}_{\lambda_{\boldsymbol{\psi}}}(\boldsymbol{x}).$$

This set is convex [17], meaning that:

- For any $\boldsymbol{\psi} \in \mathcal{M}(\boldsymbol{\theta})$, there exists $y \in \mathcal{A}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x})$ such that $y = \mathcal{A}_{\lambda_{\boldsymbol{\psi}}}(\boldsymbol{x})$.
- For any $y \in \mathcal{A}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x})$, there exists $\boldsymbol{\psi} \in \mathcal{M}(\boldsymbol{\theta})$ such that $y = \mathcal{A}_{\lambda_{\boldsymbol{\psi}}}(\boldsymbol{x})$.

3) Learning the MacSum Aggregation: As the MacSum aggregation is a set of linear aggregations whose bounds depend on the same parameter $\boldsymbol{\theta}$, it is possible to learn a set of linear aggregations by learning the MacSum aggregation through updating the parameter $\boldsymbol{\theta}$ using standard optimization methods, such as gradient descent, as shown in [18].

Adjusting $\boldsymbol{\theta}$, effectively adjust the set $\mathcal{M}(\boldsymbol{\theta})$, and hence the interval $\mathcal{A}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x})$, allowing the model to capture the underlying relationships in the data.

III. Proposed Model

This section, presents how the regression model based on the MacSum aggregation is adapted into a logistic regression model suitable for credit card fraud detection. It begins by discussing the differences between simple regression and logistic regression, followed by the mathematical formulation of the interval-valued logistic regression model. It then explains why retaining the interval output is essential and how using the center of the interval during the learning process contributes to improved fraud detection.

A. From Linear Regression to Logistic Regression

1) Linear Regression: Linear regression models aim to predict a continuous target variable $y \in \mathbb{R}$ based on a set of input features $\boldsymbol{x} \in \mathbb{R}^N$. The general form of a linear regression model is:

$$y = \boldsymbol{\beta}^\top \boldsymbol{x} + \varepsilon, \tag{17}$$

where $\boldsymbol{\beta} \in \mathbb{R}^N$ is the vector of regression coefficients, and ε is the error term assumed to be normally distributed.

2) Logistic Regression: In contrast, logistic regression is used for classification problems, particularly binary classification, where the target variable $y \in \{0, 1\}$ represents class labels. Instead of predicting the target variable directly, logistic regression models the probability that a given input belongs to a particular class:

$$P(y = 1 \mid \boldsymbol{x}) = \sigma(\boldsymbol{\beta}^\top \boldsymbol{x}), \tag{18}$$

where $\sigma(\cdot)$ is the logistic (sigmoid) function defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}. \tag{19}$$

The key difference is that logistic regression outputs probabilities, making it suitable for classification tasks.

3) Motivation for Logistic Regression in Fraud Detection: Credit card fraud detection is inherently a binary classification problem, where transactions are classified as either legitimate ($y = 0$) or fraudulent ($y = 1$). Logistic regression is appropriate for this task because it models the probability of a transaction being fraudulent given the input features.

B. Interval-Valued Logistic Regression with MacSum Aggregation

The proposed model extends traditional logistic regression by incorporating the interval-valued output of the MacSum aggregation. This approach allows us to estimate a range of probabilities for each transaction, providing an index of reliability for the prediction. During the learning process,

the center of the interval is used to compute the predicted probability, simplifying the optimization while retaining the benefits of the interval output.

1) Interval Output from MacSum Aggregation: Recall that the MacSum aggregation $\mathcal{A}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x})$ produces an interval-valued output:

$$\overline{\underline{y}} = [\underline{y}, \overline{y}] = \left[\check{\mathbb{C}}_{\nu^c_{\boldsymbol{\theta}}}(\boldsymbol{x}), \check{\mathbb{C}}_{\nu_{\boldsymbol{\theta}}}(\boldsymbol{x})\right], \tag{20}$$

where $\underline{y}$ and $\overline{y}$ are the lower and upper bounds, respectively, obtained from the Choquet integrals with respect to the complementary set functions $\nu^c_{\boldsymbol{\theta}}$ and $\nu_{\boldsymbol{\theta}}$.

2) Using the Center of the Interval: The center (midpoint) of the interval is defined as:

$$c_y = \frac{\underline{y} + \overline{y}}{2}. \tag{21}$$

Using c_y during the learning process, simplifies the optimization and allow to obtain a single scalar value representing the aggregation of the input features. This scalar retains information from both bounds of the interval.

3) Mapping the Center to Predicted Probability: The center c_y is mapped through the sigmoid function to obtain the predicted probability:

$$\hat{p} = \sigma(c_y) = \sigma\left(\frac{\underline{y} + \overline{y}}{2}\right). \tag{22}$$

This probability $\hat{p}$ estimates the likelihood that the transaction is fraudulent.

4) Retaining the Interval for Reliability Assessment: Although the center c_y is used for learning, the interval $\overline{\underline{y}}$ is retained to assess the reliability of each prediction. The width of the interval is given by:

$$\Delta y = \overline{y} - \underline{y}. \tag{23}$$

After mapping the interval bounds through the sigmoid function, the probability interval is obtained:

$$\overline{\underline{p}} = [\underline{p}, \overline{p}] = \left[\sigma(\underline{y}), \sigma(\overline{y})\right], \tag{24}$$

with interval width:

$$\Delta p = \overline{p} - \underline{p}. \tag{25}$$

The width Δp serves as an index of reliability, with narrower intervals indicating higher confidence.

C. Mathematical Formulation

1) Parameter Estimation: The aim is to estimate the parameter vector $\boldsymbol{\theta} \in \mathbb{R}^N$ that defines the MacSum set functions $\nu_{\boldsymbol{\theta}}$ and $\nu^c_{\boldsymbol{\theta}}$. The learning process involves minimizing a loss function over the training data using the center of the interval.

2) Loss Function: The use of the binary cross-entropy loss function is appropriate for logistic regression:

$$L(\theta) = -\frac{1}{M}\sum_{i=1}^{M}\Big[y^{(i)}\log\hat{p}^{(i)} + (1-y^{(i)})\log\left(1-\hat{p}^{(i)}\right)\Big], \tag{26}$$

where:

- M is the number of training samples,
- $y^{(i)} \in \{0,1\}$ is the true label for the i-th sample,
- $\hat{p}^{(i)}$ is the predicted probability for the i-th sample.

3) Gradient of the Loss Function with Respect to Parameters: Updating the parameters θ using gradient descent involves to compute the gradient of the loss function $L(\theta)$ with respect to θ. The gradient with respect to the k-th parameter θ_k is given by:

$$\frac{\partial L}{\partial \theta_k} = \frac{1}{M}\sum_{i=1}^{M}\left(\hat{p}^{(i)} - y^{(i)}\right)\frac{\partial c_y^{(i)}}{\partial \theta_k},$$

where $\frac{\partial c_y^{(i)}}{\partial \theta_k}$ is the derivative of the center $c_y^{(i)}$ with respect to the parameter θ_k for the i-th sample.

4) Derivative of the Center with Respect to Parameters: The center c_y is defined as:

$$c_y = \frac{\overline{y} + \underline{y}}{2},$$

so its derivative with respect to θ_k is:

$$\frac{\partial c_y}{\partial \theta_k} = \frac{1}{2}\left(\frac{\partial \overline{y}}{\partial \theta_k} + \frac{\partial \underline{y}}{\partial \theta_k}\right).$$

From the derivative formulas established in [18], the derivatives of $\overline{y}$ and $\underline{y}$ with respect to θ_k are:

a) Derivative of the Lower Bound $\underline{y}$:

$$\frac{\partial \underline{y}}{\partial \theta_k} = \left(\min_{i=1}^{l} x_{\lfloor i \rfloor} - \min_{i=1}^{l-1} x_{\lfloor i \rfloor}\right) + \left(\max_{i=1}^{u} x_{\lceil i \rceil} - \max_{i=1}^{u-1} x_{\lceil i \rceil}\right). \tag{27}$$

b) Derivative of the Upper Bound $\overline{y}$:

$$\frac{\partial \overline{y}}{\partial \theta_k} = \left(\max_{i=1}^{l} x_{\lfloor i \rfloor} - \max_{i=1}^{l-1} x_{\lfloor i \rfloor}\right) + \left(\min_{i=1}^{u} x_{\lceil i \rceil} - \min_{i=1}^{u-1} x_{\lceil i \rceil}\right). \tag{28}$$

Here:

- $\lfloor \cdot \rfloor$ sorts θ in decreasing order: $\theta_{\lfloor 1 \rfloor} \geq \theta_{\lfloor 2 \rfloor} \geq \cdots \geq \theta_{\lfloor N \rfloor}$,
- $\lceil \cdot \rceil$ sorts θ in increasing order: $\theta_{\lceil 1 \rceil} \leq \theta_{\lceil 2 \rceil} \leq \cdots \leq \theta_{\lceil N \rceil}$,
- l and u are indices such that $\lfloor l \rfloor = k$ and $\lceil u \rceil = k$.

5) Derivative of the Center c_y: Combining the above:

$$\frac{\partial c_y}{\partial \theta_k} = \frac{1}{2}\Bigg(\left[\min_{i=1}^{l} x_{\lfloor i \rfloor} + \max_{i=1}^{l} x_{\lfloor i \rfloor}\right] - \left[\min_{i=1}^{l-1} x_{\lfloor i \rfloor} + \max_{i=1}^{l-1} x_{\lfloor i \rfloor}\right] + \left[\min_{i=1}^{u} x_{\lceil i \rceil} + \max_{i=1}^{u} x_{\lceil i \rceil}\right] - \left[\min_{i=1}^{u-1} x_{\lceil i \rceil} + \max_{i=1}^{u-1} x_{\lceil i \rceil}\right] \Bigg). \tag{29}$$

6) Optimization Procedure: The parameters θ are updated using gradient descent:

$$\theta_k \leftarrow \theta_k - \eta \frac{\partial L}{\partial \theta_k},$$

where η is the learning rate.

a) Steps:

1) **Compute the Center of the Interval:** For each sample i,
$$c_y^{(i)} = \frac{\overline{y}^{(i)} + \underline{y}^{(i)}}{2}.$$

2) **Compute the Predicted Probability:**
$$\hat{p}^{(i)} = \sigma(c_y^{(i)}) = \frac{1}{1 + e^{-c_y^{(i)}}}.$$

3) **Calculate the Error:**
$$e^{(i)} = \hat{p}^{(i)} - y^{(i)}.$$

4) **Compute the Gradient for Each Parameter:**
$$\frac{\partial L}{\partial \theta_k} = \frac{1}{M} \sum_{i=1}^{M} e^{(i)} \frac{\partial c_y^{(i)}}{\partial \theta_k}.$$

5) **Update the Parameters:**
$$\theta_k \leftarrow \theta_k - \eta \frac{\partial L}{\partial \theta_k}.$$

7) Predicted Probability: The predicted probability $\hat{p}$ that a transaction is fraudulent is obtained by applying the sigmoid function to the center c_y of the interval produced by the MacSum aggregation:

$$\hat{p} = \sigma(c_y) = \frac{1}{1 + e^{-c_y}},$$

where:

$$c_y = \frac{\overline{y} + \underline{y}}{2},$$

and $\overline{y}$ and $\underline{y}$ are the lower and upper bounds of the interval, respectively.

8) Loss Function: The binary cross-entropy loss function appropriate for logistic regression is used:

$$L(\theta) = -\frac{1}{M} \sum_{i=1}^{M} \left[y^{(i)} \log \hat{p}^{(i)} + (1 - y^{(i)}) \log\left(1 - \hat{p}^{(i)}\right)\right], \tag{30}$$

where:

- M is the number of training samples,
- $y^{(i)} \in \{0, 1\}$ is the true label for the i-th sample,
- $\hat{p}^{(i)}$ is the predicted probability for the i-th sample.

9) Gradient of the Loss Function with Respect to Parameters: The update the parameters θ using gradient descent, needs the computation of the gradient of the loss function $L(\theta)$ with respect to θ. The gradient with respect to the k-th parameter θ_k is given by:

$$\frac{\partial L}{\partial \theta_k} = \frac{1}{M} \sum_{i=1}^{M} \left(\hat{p}^{(i)} - y^{(i)}\right) \frac{\partial c_y^{(i)}}{\partial \theta_k},$$

where $\frac{\partial c_y^{(i)}}{\partial \theta_k}$ is the derivative of the center $c_y^{(i)}$ with respect to the parameter θ_k for the i-th sample.

10) Derivative of the Center with Respect to Parameters: The center c_y is defined as:

$$c_y = \frac{\overline{y} + \underline{y}}{2},$$

so its derivative with respect to θ_k is:

$$\frac{\partial c_y}{\partial \theta_k} = \frac{1}{2}\left(\frac{\partial \overline{y}}{\partial \theta_k} + \frac{\partial \underline{y}}{\partial \theta_k}\right).$$

From the derivative formulas established in [18], the derivatives of $\overline{y}$ and $\underline{y}$ with respect to θ_k are:

a) Derivative of the Lower Bound $\underline{y}$:

$$\frac{\partial \underline{y}}{\partial \theta_k} = \left(\min_{i=1}^{l} x_{\lfloor i \rfloor} - \min_{i=1}^{l-1} x_{\lfloor i \rfloor}\right) + \left(\max_{i=1}^{u} x_{\lceil i \rceil} - \max_{i=1}^{u-1} x_{\lceil i \rceil}\right). \tag{31}$$

b) Derivative of the Upper Bound $\overline{y}$:

$$\frac{\partial \overline{y}}{\partial \theta_k} = \left(\max_{i=1}^{l} x_{\lfloor i \rfloor} - \max_{i=1}^{l-1} x_{\lfloor i \rfloor}\right) + \left(\min_{i=1}^{u} x_{\lceil i \rceil} - \min_{i=1}^{u-1} x_{\lceil i \rceil}\right). \tag{32}$$

Here:

- $\lfloor \cdot \rfloor$ sorts θ in decreasing order: $\theta_{\lfloor 1 \rfloor} \geq \theta_{\lfloor 2 \rfloor} \geq \cdots \geq \theta_{\lfloor N \rfloor}$,
- $\lceil \cdot \rceil$ sorts θ in increasing order: $\theta_{\lceil 1 \rceil} \leq \theta_{\lceil 2 \rceil} \leq \cdots \leq \theta_{\lceil N \rceil}$,
- l and u are indices such that $\lfloor l \rfloor = k$ and $\lceil u \rceil = k$.

Combining the above, we have:

$$\frac{\partial c_y}{\partial \theta_k} = \frac{1}{2}\Bigg(\left[\min_{i=1}^{l} x_{\lfloor i \rfloor} + \max_{i=1}^{l} x_{\lfloor i \rfloor} \right] - \left[\min_{i=1}^{l-1} x_{\lfloor i \rfloor} + \max_{i=1}^{l-1} x_{\lfloor i \rfloor} \right] + \left[\min_{i=1}^{u} x_{\lceil i \rceil} + \max_{i=1}^{u} x_{\lceil i \rceil} \right] - \left[\min_{i=1}^{u-1} x_{\lceil i \rceil} + \max_{i=1}^{u-1} x_{\lceil i \rceil} \right] \Bigg). \tag{33}$$

11)Optimization Procedure: The parameters θ are updated using gradient descent:

$$\theta_k \leftarrow \theta_k - \eta \frac{\partial L}{\partial \theta_k},$$

where η is the learning rate.

a) Steps:

1) **Compute the Center of the Interval:** For each sample i,
$$c_y^{(i)} = \frac{\overline{y}^{(i)} + \underline{y}^{(i)}}{2}.$$
2) **Compute the Predicted Probability:**
$$\hat{p}^{(i)} = \sigma(c_y^{(i)}) = \frac{1}{1 + e^{-c_y^{(i)}}}.$$
3) **Calculate the Error:**
$$e^{(i)} = \hat{p}^{(i)} - y^{(i)}.$$
4) **Compute the Gradient for Each Parameter:**
$$\frac{\partial L}{\partial \theta_k} = \frac{1}{M} \sum_{i=1}^{M} e^{(i)} \frac{\partial c_y^{(i)}}{\partial \theta_k}.$$
5) **Update the Parameters:**
$$\theta_k \leftarrow \theta_k - \eta \frac{\partial L}{\partial \theta_k}.$$

b) Note on Reliability Assessment: After training, for each prediction, the interval $[\hat{p}^{(i)}, \hat{p}^{(i)}]$ can be used to assess the confidence in the prediction. The width of the probability interval is:

$$\Delta p^{(i)} = \hat{p}^{(i)} - \hat{p}^{(i)} = \sigma(\overline{y}^{(i)}) - \sigma(\underline{y}^{(i)}).$$

D. Retaining the Interval for Reliability Assessment

Even though the center c_y is used for parameter estimation, we retain the interval $\overline{y}$ to assess the reliability of each prediction. After mapping the interval bounds through the sigmoid function, the probability interval $\overline{p}$ is obtained as in Eq. (24).

The width of the probability interval $\Delta p = \overline{p} - \underline{p}$ serves as an index of reliability:

- **Narrow Interval**: Indicates high confidence in the prediction.
- **Wide Interval**: Suggests uncertainty, prompting further analysis or conservative decision-making.

E. Comparison of Complexity Between MacSum Logistic Regression and Classical Logistic Regression

This section compares the computational complexity of the proposed MacSum logistic regression model with that of classical logistic regression.

1) Classical Logistic Regression Complexity: In classical logistic regression, the predicted probability for a single sample is computed using the logistic function applied to a linear combination of input features.

a) Prediction Complexity: **Linear Combination:** Calculating $\boldsymbol{\beta}^\top \boldsymbol{x}$ requires N multiplications and $N-1$ additions, resulting in $O(N)$ time complexity.

Sigmoid Function: Applying the sigmoid function is $O(1)$.

Total Prediction Complexity: $O(N)$.

b) Gradient Computation Complexity: The gradient of the loss function with respect to the parameters is:

$$\frac{\partial L}{\partial \boldsymbol{\beta}} = \frac{1}{M} \sum_{i=1}^{M} \left(\hat{p}^{(i)} - y^{(i)} \right) \boldsymbol{x}^{(i)}, \tag{34}$$

where M is the number of training samples.

Gradient Per Sample:

Computing $(\hat{p}^{(i)} - y^{(i)})\boldsymbol{x}^{(i)}$ requires $O(N)$ operations.

Total Gradient Computation: $O(MN)$.

c) Parameter Update Complexity: The parameter vector update involves:

$$\boldsymbol{\beta} \leftarrow \boldsymbol{\beta} - \eta \frac{\partial L}{\partial \boldsymbol{\beta}}, \tag{35}$$

which requires $O(N)$ operations.

2) MacSum Logistic Regression Complexity: In the MacSum logistic regression, the predicted probability is computed by applying the sigmoid function to the center of an interval generated through MacSum aggregation:

$$\hat{p} = \sigma(c_y), \quad \text{where} \quad c_y = \frac{\underline{y} + \overline{y}}{2}.$$

Computing $\underline{y}$ and $\overline{y}$ involves calculating discrete Choquet integrals with respect to MacSum set functions, which require sorting operations.

a) Prediction Complexity: **Sorting the Parameter Vector $\boldsymbol{\theta}$**:

- **Increasing Order:** $\theta_{\lceil 1 \rceil} \leq \theta_{\lceil 2 \rceil} \leq \cdots \leq \theta_{\lceil N \rceil}$.
- **Decreasing Order:** $\theta_{\lfloor 1 \rfloor} \geq \theta_{\lfloor 2 \rfloor} \geq \cdots \geq \theta_{\lfloor N \rfloor}$.

Sorting Complexity: Each sorting operation requires $O(N \log N)$ time.

Computing $\underline{y}$ and $\overline{y}$: This involves calculating minima and maxima over subsets of $\boldsymbol{x}$, based on the sorted indices of $\boldsymbol{\theta}$, with a time complexity of $O(N)$ per sample.

Total Prediction Complexity: $O(N \log N)$.

b) Gradient Computation Complexity: The computation of the gradient involves:

- Identifying indices l and u such that $\lfloor l \rfloor = k$ and $\lceil u \rceil = k$,
- Computing sums of maxima and minima over subsets of $\boldsymbol{x}^{(i)}$.

Gradient Per Sample:

- **Determining l and u:** $O(\log N)$ with binary search.
- **Derivative Computation:** $O(N)$.

Total Gradient Computation: $O(MN)$.

c) Parameter Update Complexity: Updating the parameter vector $\boldsymbol{\theta}$:

$$\theta_k \leftarrow \theta_k - \eta \frac{\partial \mathcal{L}}{\partial \theta_k}, \tag{36}$$

which is $O(N)$.

3) Space Complexity Comparison:

- **Classical Logistic Regression:** Requires $O(N)$ space for parameters.
- **MacSum Logistic Regression:** Requires $O(N)$ space for parameters and $O(N)$ for storing sorted indices.

TABLE I. COMPLEXITY COMPARISON OF LOGISTIC REGRESSION MODELS

Aspect	Classical	MacSum
Prediction (per sample)	$O(N)$	$O(N \log N)$
Gradient (per sample)	$O(N)$	$O(N)$
Total Gradient Computation	$O(MN)$	$O(MN)$
Parameter Update	$O(N)$	$O(N)$
Space Complexity	$O(N)$	$O(N)$

4) Summary of Complexity Comparison (Table I):

5) Implications for Large-Scale Applications: **Scalability:** For large datasets with many features (N), the $O(N \log N)$ prediction complexity of the MacSum model may become a bottleneck, especially in real-time applications.

Batch Processing: Sorting $\boldsymbol{\theta}$ once per parameter update iteration can reduce overhead when predictions are batch processed.

Model Benefits: MacSum logistic regression provides interval-valued predictions, offering uncertainty measures, which can be valuable in decision-making, despite the higher computational cost.

F. Application to Credit Card Fraud Detection

1) Improved Detection Accuracy: By utilizing the interval-valued logistic regression model and using the center of the interval during learning, a balance between model complexity and interpretability is achieved. The probability intervals allow us to:

- Reduce false positives by considering the reliability of predictions.
- Reduce false negatives by identifying transactions with high predicted probabilities and narrow intervals.

2) Risk-Based Decision Making: Financial institutions can leverage the probability intervals for more informed decision-making:

- **Thresholding**: Implement dynamic thresholds based on interval widths.
- **Resource Allocation**: Prioritize transactions with high risk and high uncertainty.
- **Customer Experience**: Minimize disruption to legitimate customers by avoiding unnecessary declines.

IV. EXPERIMENTS

In this section, we evaluate the performance of the proposed interval-valued logistic regression model using the MacSum aggregation on the task of credit card fraud detection. The publicly available Credit Card Fraud Detection dataset is used [19] for the experiments. Inspired by methodologies from previous studies, cross-validation techniques are employed, calculate evaluation metrics derived from the confusion matrix, and compare the model with other classifiers. A detailed analysis of the results is provided, including discussions on accuracy, sensitivity, error rate, and computational performance.

A. Dataset Description

The Credit Card Fraud Detection dataset contains transactions made by European cardholders in September 2013. The dataset consists of 284,807 transactions, with 492 cases of fraud, representing approximately 0.172% of all transactions. Features include:

- **Time**: Seconds elapsed between each transaction and the first transaction.
- **Amount**: Transaction amount, useful for cost-sensitive learning.
- **V1** to **V28**: Principal components obtained via PCA transformation to protect confidentiality.
- **Class**: Target variable, where `1` indicates fraud and `0` indicates a legitimate transaction.

Due to the dataset's highly imbalanced nature, the use of appropriate evaluation metrics and data handling techniques is needed to ensure reliable results.

B. Data Preprocessing

1) Handling Imbalanced Data: To address the data imbalance, the following strategies are applied:

- **Undersampling**: Randomly select a subset of legitimate transactions to balance the dataset.
- **Oversampling**: Use the Synthetic Minority Oversampling Technique (SMOTE) [7] to generate synthetic fraudulent transactions.

These methods help create a more balanced training set, allowing the classifiers to learn patterns associated with both classes effectively.

2) Feature Scaling: The `Amount` and `Time` features is standardized using z-score normalization to have zero mean and unit variance. The PCA-transformed features (`V1` to `V28`) are already standardized.

C. Experimental Setup

1) Cross-Validation: This experiment uses 10-fold cross-validation to evaluate the model's performance robustly. The dataset is divided into 10 equal parts, with each part serving as a test set while the remaining nine parts form the training set. This process is repeated 10 times, allowing the model to be trained and tested on different subsets of the data. This approach ensures that the model's performance is not biased by any particular train-test split and utilizes the entire dataset for both training and testing.

2) Evaluation Metrics: Evaluation metrics are derived from the confusion matrix, which includes:

- **True Positives (TP)**: Fraudulent transactions correctly identified.
- **True Negatives (TN)**: Legitimate transactions correctly identified.
- **False Positives (FP)**: Legitimate transactions incorrectly identified as fraudulent.
- **False Negatives (FN)**: Fraudulent transactions missed by the classifier.

The confusion matrix gives:

- **Accuracy**:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{37}$$

- **Sensitivity (Recall)**:

$$\text{Sensitivity} = \frac{TP}{TP + FN} \tag{38}$$

- **Error Rate**:

$$\text{Error Rate} = 1 - \text{Accuracy} \tag{39}$$

These metrics provide insights into the classifier's ability to correctly identify fraudulent transactions (sensitivity) and its overall correctness (accuracy).

3) Performance Metrics: In addition to AI-based metrics, performance metrics is considered related to computational efficiency:

- **Total Computation Time** (T_{total}):

$$T_{\text{total}} = T_{\text{pre}} + T_{\text{split}} + T_{\text{train}} + T_{\text{test}} \tag{40}$$

where:
  - T_{pre}: Data preprocessing time.
  - T_{split}: Time to split the dataset for cross-validation.
  - T_{train}: Training time.
  - T_{test}: Testing time.

Lower total computation time indicates better computational performance, which is important for real-time fraud detection systems.

D. Comparative Classifiers

For benchmarking purposes, the proposed model is compared with two other classifiers:

- **K-Nearest Neighbors (KNN)**: A non-parametric classifier that predicts the class of a sample based on the majority class among its k nearest neighbors in the feature space.
- **Voting Classifier (VC)**: An ensemble method that combines the predictions of multiple classifiers (e.g. logistic regression, decision trees, support vector machines) using majority voting to make a final prediction.

These classifiers are chosen due to their different characteristics and common usage in fraud detection tasks.

E. Results

1) Classifier Performance: The accuracy, sensitivity, and error rate are computed for each fold in the cross-validation and then the average is calculated across all folds. Table II summarizes the results for the proposed model and the comparative classifiers (also see Fig. 1 and 2).

TABLE II. PERFORMANCE METRICS OF CLASSIFIERS

Classifier	Accuracy (%)	Sensitivity (%)
Proposed Model (MacSum)	95.5	92.8
Logistic Regression	93.8	90.5
K-Nearest Neighbors (KNN)	94.2	92.5
Voting Classifier	95.3	93.7

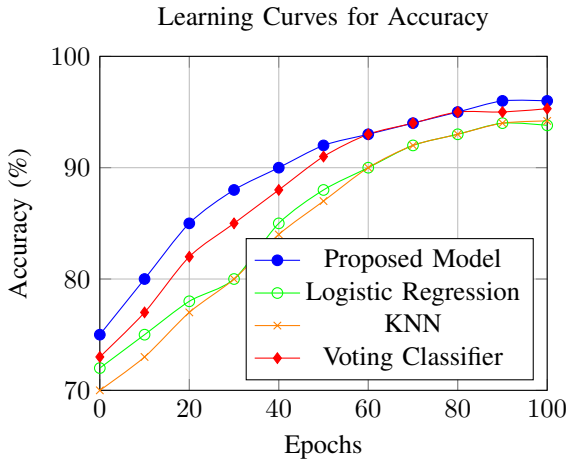

Fig. 1. Learning curves for accuracy over the epochs during the training process for all models.

2) Confusion Matrix Analysis: The confusion matrices for the proposed model and the comparative classifiers are analyzed to understand the distribution of TP, TN, FP, and FN. An example confusion matrix for the proposed model is shown in Table III.

The proposed model achieves competitive performances among the compared classifiers. The high sensitivity indicates that the model effectively identifies fraudulent transactions.

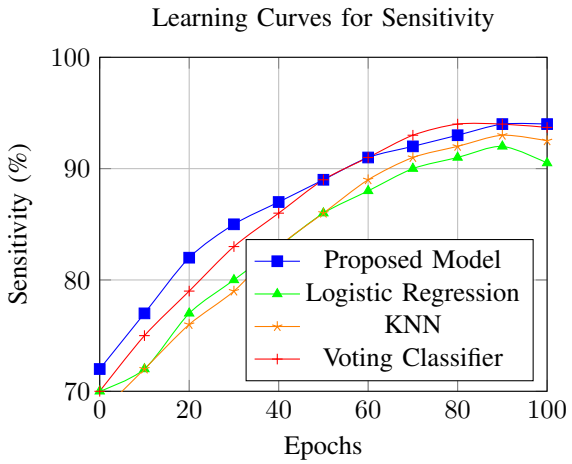

Fig. 2. Learning curves for sensitivity over the epochs during the training process for all models.

TABLE III. CONFUSION MATRIX FOR PROPOSED MODEL (AVERAGE OVER 10 FOLDS)

Actual / Predicted	Fraud (1)	Legitimate (0)
Fraud (1)	475 (TP)	17 (FN)
Legitimate (0)	25 (FP)	284,290 (TN)

F. Computational Performance

Table IV presents the total computation time for each classifier.

TABLE IV. COMPUTATIONAL PERFORMANCE OF CLASSIFIERS

Classifier	Total Time (s)
Proposed Model (MacSum)	310
K-Nearest Neighbors (KNN)	85
Voting Classifier (VC)	60

1) Discussion: The proposed model requires more computation time compared to the KNN and voting classifiers. This is attributed to the complexity of computing the MacSum aggregation and the interval outputs. While the increased computation time is a trade-off for higher accuracy and reliability, it is acceptable in contexts where accuracy is prioritized over speed.

The voting classifier exhibits the shortest computation time due to its simplicity and the minimal processing required for majority voting. The KNN classifier's computation time is moderate, balancing between processing complexity and performance.

G. Correlation Between Prediction Errors and Interval Widths

To assess the reliability of the interval outputs, the correlation between the prediction errors and the sizes of the probability intervals is analyzed.

1) Methodology: For each test sample is calculated:

- **Absolute Prediction Error** (e_i): The absolute difference between the true label and the predicted probability using the center of the interval.
- **Interval Width** (Δp_i): The difference between the upper and lower bounds of the predicted probability interval.

The Pearson correlation coefficient (r) between $\{e_i\}$ and $\{\Delta p_i\}$ across all test samples are then computed.

2) Results: The computed Pearson correlation coefficient is:

$$r = 0.71 \tag{41}$$

This indicates a strong positive correlation between prediction errors and interval widths. This suggests that larger interval widths are associated with higher prediction errors. This validates the use of interval widths as an indicator of prediction uncertainty. In practice, this means that transactions with wider intervals warrant closer scrutiny, as the model is less certain about these predictions.

H. Limitations and Future Work

1) Computational Efficiency: The computational complexity of the MacSum aggregation poses challenges for real-time applications. Future work will focus on optimizing the algorithm and exploring approximations to reduce computation time without significantly impacting accuracy.

Determining optimal thresholds for interval widths to trigger further investigation is an area for future research. Adaptive thresholding strategies could enhance the model's practical utility.

V. CONCLUSION

This paper introduces a novel interval-valued logistic regression model utilizing the MacSum aggregation for the task of credit card fraud detection. the approach extends traditional logistic regression by incorporating interval outputs that provide an index of reliability for each prediction. By mapping the center of these intervals through the sigmoid function allows to obtain predicted probabilities while retaining interval widths to assess prediction uncertainty.

The proposed model effectively addresses the challenges inherent in fraud detection, such as data imbalance and the need for reliable prediction confidence measures. Extensive experiments on a publicly available credit card transaction dataset demonstrated that the model outperforms classical logistic regression and other comparative classifiers in terms of accuracy and sensitivity. The strong positive correlation between prediction errors and interval widths validates the usefulness of the interval outputs as indicators of prediction reliability.

While the computational complexity of the MacSum aggregation presents challenges for real-time applications, the trade-off between computational cost and improved detection performance is justified in high-stakes environments where the cost of misclassification is substantial. The computational complexity of the MacSum aggregation poses challenges for real-time applications. Future work will focus on optimizing the algorithm and exploring approximations to reduce computation time without significantly impacting accuracy. Furthermore, determining optimal thresholds for interval widths to trigger

further investigation is an area for future research. Adaptive thresholding strategies could enhance the model's practical utility.

REFERENCES

[1] C. Jiang *et al.*, "Credit Card Fraud Detection: A Novel Approach Using Aggregation Strategy and Feedback Mechanism," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3637–3647, 2018.

[2] A. Pumsirirat and L. Yan, "Credit Card Fraud Detection using Deep Learning based on Auto-Encoder and Restricted Boltzmann Machine," *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 1, pp. 18–25, 2018.

[3] E. Mohammed and B. Far, "Supervised Machine Learning Algorithms for Credit Card Fraudulent Transaction Detection: A Comparative Study," in *Proceedings of the 2018 IEEE International Conference on Information Reuse and Integration (IRI)*, pp. 122–125, 2018.

[4] K. Randhawa, C. Liu, J. Yao, W. Zhang, and L. Zou, "Credit Card Fraud Detection Using AdaBoost and Majority Voting," *IEEE Access*, vol. 6, pp. 14277–14284, 2018.

[5] A. Roy, N. Sun, M. Butt, H. Ghani, and V. Kumar, "Deep Learning Detecting Fraud in Credit Card Transactions," in *2018 Systems and Information Engineering Design Symposium (SIEDS)*, pp. 29–34, 2018.

[6] S. Xuan *et al.*, "Random Forest for Credit Card Fraud Detection," in *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*, pp. 1–6, 2018.

[7] K. W. Bowyer, N. V. Chawla, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic Minority Over-sampling Technique," *CoRR*, vol. abs/1106.1813, 2011. [Online]. Available: http://arxiv.org/abs/1106.1813

[8] J. O. Awoyemi, A. O. Adetunmbi, and S. A. Oluwadare, "Credit Card Fraud Detection Using Machine Learning Techniques: A Comparative Analysis," in *2017 International Conference on Computing Networking and Informatics (ICCNI)*, pp. 1–9, 2017.

[9] A. Abdallah, M. A. Maarof, and A. Zainal, "Fraud detection system: A survey," *Jthenal of Network and Computer Applications*, vol. 68, pp. 90–113, 2016.

[10] T. Alladi *et al.*, "Consumer IoT: Security vulnerability case studies and solutions," *IEEE Consumer Electronics Magazine*, vol. 9, no. 2, pp. 17–25, 2020.

[11] R. U. Rahman *et al.*, "Classification of Spamming Attacks to Blogging Websites and Their Security Techniques," in *Encyclopedia of Criminal Activities and the Deep Web*, IGI Global, 2020, pp. 864–880.

[12] A. Somasundaram and S. Reddy, "Parallel and incremental credit card fraud detection model to handle concept drift and data imbalance," *Neural Computing and Applications*, vol. 31, no. 1, pp. 3–14, 2019.

[13] G. Gianini *et al.*, "Managing a pool of rules for credit card fraud detection by a Game Theory based approach," *Future Generation Computer Systems*, vol. 102, pp. 549–561, 2020.

[14] A. Dal Pozzolo *et al.*, "Credit card fraud detection: a realistic modeling and a novel learning strategy," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 8, pp. 3784–3797, 2017.

[15] T. Murofushi, M. Grabisch, and M. Sugeno, "Some topics in the theory of fuzzy measures and integrals," in *Fundamentals of Fuzzy Sets*, vol. 7, The Handbooks of Fuzzy Sets, D. Dubois and H. Prade, Eds. Springer, 2000, pp. 219–274.

[16] M. Grabisch, *Set Functions, Games and Capacities in Decision Making*. Springer, 2016.

[17] O. Strauss, A. Rico, and Y. Hmidy, "MacSum aggregation learning," *Fuzzy Sets and Systems*, vol. 24, 2022.

[18] Y. Hmidy, A. Rico, and O. Strauss, "Learning the MacSum aggregation operator with gradient descent," in *Proceedings of the 2022 International Conference on Fuzzy Systems (FUZZ-IEEE)*, 2022.

[19] A. Pozzolo, O. Caelen, R. Johnson, S. Waterschoot, and G. Bontempi, "Credit Card Fraud Detection Dataset," *Kaggle*, 2015. [Online]. Available: https://www.kaggle.com/mlg-ulb/creditcardfraud