UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

An engine to simulate insurance fraud network data

Campo, Bavo D.C.; Antonio, K.

DOI
10.1007/s13385-024-00399-z
Publication date
2025
Document Version
Final published version
Published in
European Actuarial Journal

License
Article 25fa Dutch Copyright Act (https://www.openaccess.nl/en/policies/open-access-in-dutch-copyright-law-taverne-amendment)
Link to publication

Citation for published version (APA):
Campo, B. D. C., & Antonio, K. (2025). An engine to simulate insurance fraud network data. *European Actuarial Journal*, *15*(1), 255–295. https://doi.org/10.1007/s13385-024-00399-z

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:15 Nov 2026

European Actuarial Journal (2025) 15:255–295
https://doi.org/10.1007/s13385-024-00399-z

ORIGINAL RESEARCH PAPER

An engine to simulate insurance fraud network data

Bavo D. C. Campo[1] · **Katrien Antonio**[1,2,3,4]

Received: 28 October 2023 / Revised: 9 August 2024 / Accepted: 15 August 2024 /
Published online: 18 December 2024
© The Author(s), under exclusive licence to European Actuarial Journal Association 2024

Abstract

Traditionally, the detection of fraudulent insurance claims relies on business rules and expert judgement which makes it a time-consuming and expensive process (Óskarsdóttir et al. in Risk Anal 42(8):1872–1890, 2022). Consequently, researchers have been examining ways to develop efficient and accurate analytic strategies to flag suspicious claims. Feeding learning methods with features engineered from the social network of parties involved in a claim is a particularly promising strategy (see for example Óskarsdóttir et al. in Risk Anal 42(8):1872–1890, 2022; Van Vlasselaer et al. in Manag Sci 63(9):3090–3110, 2016; Tumminello et al. in J Risk Insur 90(2), 381–419, 2023). When developing a fraud detection model, however, we are confronted with several challenges. The uncommon nature of fraud, for example, creates a high class imbalance which complicates the development of well performing analytic classification models. In addition, only a small number of claims are investigated and get a label, which results in a large corpus of unlabeled data. Yet another challenge is the lack of publicly available data. This hinders not only the development of new methods, but also the validation of existing techniques. We therefore design a simulation machine that is engineered to create synthetic data with a network structure and available covariates similar to the real life insurance fraud data set analyzed in Óskarsdóttir et al. (Risk Anal 42(8):1872–1890, 2022). Further, the user has control over several data-generating mechanisms. We can specify the total number of policyholders and parties, the desired level of imbalance and the (effect size of the) features in the fraud generating model. As such, the simulation engine enables researchers and practitioners to examine several methodological challenges as well as to test their (development strategy of) insurance fraud detection models in a range of different settings. Moreover, large synthetic data sets can be generated to evaluate the predictive performance of (advanced) machine learning techniques.

✉ Bavo D. C. Campo
bavo.campo@kuleuven.be

1 Faculty of Economics and Business, KU Leuven, Leuven, Belgium

2 Faculty of Economics and Business, University of Amsterdam, Amsterdam, The Netherlands

3 LRisk, Leuven Research Center on Insurance and Financial Risk Analysis, KU Leuven, Leuven, Belgium

4 LStat, Leuven Statistics Research Center, KU Leuven, Leuven, Belgium

Keywords Social network data · Simulation machine · Insurance fraud detection · Class imbalance · Unlabeled data

1 Introduction

Fraudulent activity in the insurance industry causes significant financial losses for both insurance companies and policyholders. In non-life insurance, the total yearly cost of fraudulent claims is estimated to be more than $40 billion in the United States [10]. For an average family, this leads to an increased yearly premium of $400 to $700 [10]. To detect and mitigate fraud, insurance companies implement various anti-fraud measures. Traditionally, insurance companies rely on a combination of business rules and expert judgment to identify fraudulent claims [37]. The business rules flag suspicious claims, which are then sent to experts who determine whether the claim is fraudulent or not [56]. These in-depth investigations, however, are time-intensive and costly. Researchers have therefore developed insurance fraud detection models combining business rules and analytic techniques to flag the most suspicious claims, which are sent to the experts for further investigation. As such, experts can focus solely on the claims with a high likelihood of fraud and avoid spending precious resources on examining non-fraudulent claims. Insurers predominately rely on analytics to prevent fraud [9]. Moreover, fraud detection is considered an area for more intense use of big data and analytics in the insurance industry.

Within fraud analytics, researchers rely on a wide range of statistical and machine learning techniques (see Ngai et al. [33] and Albashrawi [2] for an overview). In the literature, we find examples of both supervised and unsupervised (machine learning) techniques to construct fraud detection models. Vosseler [55], for example, developed an unsupervised anomaly detection technique to identify fraudulent insurance claims. Nur Prasasti et al. [34] constructed fraud detection models using neural networks and tree-based machine learning techniques. The accuracy of such models, however, greatly depends on the input. We typically use traditional claim characteristics, such as the claim amount, as features in a fraud detection model [6]. These characteristics are static whereas the typical features of fraudsters tend to be dynamic [17, 37, 51]. According to Jensen [22], fraudsters adapt their tactics in response to fraud detection systems and hence, the typical fraudster profile evolves over time.

One particularly promising approach to capture the characteristics of fraudsters is through social network analytics [37, 51, 54]. In an insurance context, social network data captures the relationship between claims on the one hand and policyholders and other involved parties (e.g., garage, broker, expert) on the other hand. By analyzing the social network structure of reported claims, insurers can unravel patterns and relationships among policyholders and claims that are indicative of fraud. Moreover, this approach potentially uncovers organized schemes of collaborating fraudsters who are trying to hide their tracks. Criminals often commit crimes in groups to increase rewards and to decrease the risk of detection [3, 42]. Furthermore, in organized crime, such as fraud, social connections play a crucial role since these connections are based on trust and provide access to co-offenders and opportunities [53]. As such, social

network analytics can assist in identifying enduring relationships between fraudsters even as overt characteristics undergo changes.

Nonetheless, publicly available data on insurance fraud is scarce, in particular data with a social network structure. This makes it difficult for researchers to test, validate and improve existing fraud detection methods. Moreover, the lack of data hinders the reproducibility of research findings and the discovery of novel methodologies [5]. The main reason for the limited availability is the sensitive nature of the data [28]. Insurance data sets contain confidential information about the insurance company and its policyholders. In consequence, the data used to develop and validate fraud detection methods and models is almost never shared. Researchers in fraud analytics are confronted with several inherent methodological challenges when developing analytic models for fraud detection [5]. Investigating suspicious claims is a time-intensive and expensive process, which results in only few claims being labeled (i.e., whether the claim is fraudulent or non-fraudulent) [56]. In addition, due to fraud being uncommon, data sets are often characterized by a severe class imbalance (see, for example [17, 37, 47]). Another challenge is the continuous development of machine learning techniques [5]. To assess whether these perform better or on par with existing techniques, we need to evaluate competing models or techniques in similar conditions. That is, using the same (type of) data set.

One way to address the scarcity of publicly available data, is by using a simulation engine to generate synthetic data that mimics the structure of the original data set [28]. Simulation engines enable researchers to perform benchmark studies to investigate the properties and performance of various statistical and machine learning techniques [25, 31]. Additionally, synthetic data facilitates the development of new methods and stimulates the reproducibility of research. Recently, simulation engines have gained considerable attention in actuarial science. Gabrielli and Wüthrich [13] developed a simulation engine to generate individual claim histories of non-life insurance claims and So et al. [44] devised an engine to generate synthetic telematics data. However, both engines employ neural networks trained on a single real insurance data set to generate synthetic data that closely mirrors the original data. Conversely, the simulation machine of Avanzi et al. [4] does not emulate a single data set when generating individual non-life insurance claims. Instead, it allows users to generate a diverse set of scenarios which vary in complexity.

In this paper, we design a simulation machine that generates synthetic insurance fraud network data. The simulation engine is inspired by and mimics the structure and properties of the non-life motor insurance data used in Óskarsdóttir et al. [37], which contains both traditional claim and policy(holders) characteristics as well as social network features. When generating the synthetic data, the user has control over several data-generating mechanisms. Both policyholder, contract-specific and claim characteristics as well as the dependence between them can be adjusted. Further, when simulating the number of claims, the individual claim costs and the claim labels (i.e., fraudulent or non-fraudulent), the user can specify the (effect size of the) features that are used in the data-generating model. Specific characteristics of the social network structure and fraud investigation process can be adjusted as well. In addition, the size of the resulting data set and the required level of class imbalance can be set by the user. Hereby, the simulation engine provides researchers with a powerful and

valuable tool for evaluating and improving the performance of fraud detection methods across various scenarios. The simulation engine's ability to produce large data sets makes it ideal for machine learning techniques that require large amounts of data. Furthermore, researchers and practitioners can use the engine to test their (development strategy of) insurance fraud detection models and to investigate the performance of a wide range of analytic methods in tackling the challenges inherent to fraud data sets, e.g., the severe class imbalance and large number of missing labels. Additionally, by examining a specific method or model in these scenarios, researchers can gain a better understanding of its strengths and limitations.

This paper is structured as follows. In Sect. 2, we discuss the fraud cycle, existing analytic fraud detection strategies and provide a comprehensive overview of social network analytics for fraud detection strategies. Further, we address several challenges that are an integral part of fraud detection research. Section 3 delves into the design of the simulation engine and explains how a synthetic data set is generated. Section 4 showcases the simulation engine's capabilities by generating and exploring two different types of synthetic data sets. In the first type, we introduce a social network effect when simulating the claim labels and in the second type, we omit the social network effect. Additionally, using the artificial data, we provide a practical demonstration of the development and evaluation of a fraud detection model. We conclude the paper with Sect. 5.

2 Fighting fraud with data analytics: strategies, techniques and challenges

In this section, we provide an overview of some conventional and emerging approaches to fraud detection. We highlight the role of analytics in uncovering fraudulent activities and discuss the various challenges that researchers and practitioners face in this domain.

2.1 Uncovering fraud: traditional and analytic approaches

In insurance, policyholders file a claim to request a financial compensation for a covered loss. The insurance company retains all relevant information on past and current claims in a database, typically stored in a tabular format. The data set encompasses claim, policyholder, and contract-specific attributes, collectively referred to as traditional claim characteristics. Certain claims, however, are illegitimate and detecting fraudulent claims is essential for insurance companies to prevent financial losses and to protect their policyholders. Hereto, insurers adopt either a traditional, a data-driven or a combined strategy to flag suspicious claims.

Expert-based fraud detection

The traditional approach to detect fraud is expert-based [6]. This approach is two-fold. First, the insurance company flags certain claims as suspicious. To flag suspicious

claims, companies rely on a set of business rules that are based on insights from previous investigations. Second, once a claim is flagged as suspicious, the claim is passed to an expert, who conducts an in-depth investigation to determine whether the claim is fraudulent or not [56]. Hereafter, newly obtained insights from the investigation are used to adjust the procedure for identifying suspicious claims. This is known as the fraud detection cycle.

The expert-based approach, however, has some notable shortcomings [6]. It is highly dependent on the manual input and expertise of the expert. In addition, investigating claims is a time-intensive and expensive process. Moreover, the dynamic nature of fraud requires that the rule base to flag suspicious claims needs to be continuously monitored, improved and updated. To address these drawbacks, researchers have developed alternative approaches to detect fraud in a more automated manner [2, 6, 7, 33]. Notwithstanding, even with the alternative approaches, the inclusion of expert knowledge and input is critical to the success of the fraud detection system.

Fraud analytics

The limitations of the expert-based approach prompted researchers to develop data-driven methodologies to detect fraud. In the literature, either supervised or unsupervised learning techniques or a combination of both are employed. Within fraud detection, we use unsupervised learning techniques to identify anomalous behavior [6]. There is a plethora of anomaly detection techniques available and Hilal et al. [21] provides an extensive overview of anomaly detection techniques to detect financial fraud. In our paper, we focus on supervised learning methods which learn from historical, labeled data. There are numerous supervised techniques available that can be utilized to construct a fraud detection model, ranging from logistic regression models to neural networks. A comprehensive literature review of the supervised techniques applied in financial fraud detection can be found in [2, 7, 33].

One way to tackle insurance fraud detection is by treating it as a binary classification problem. Here, our response variable Y_i can take on only two values: 0 (non-fraud) or 1 (fraud). Further, each claim i has a corresponding covariate vector $\boldsymbol{x}_i$. The values herein correspond to a set of features that provide information on the policyholder, contract, claim, network and any other relevant features that can assist in identifying fraudulent claims. The general equation of a predictive classification model is

$$P[Y_i = 1|\boldsymbol{x}_i] = f(\boldsymbol{x}_i) \tag{1}$$

where $P[Y_i = 1|\boldsymbol{x}_i]$ denotes the probability that claim i is fraudulent given covariate vector $\boldsymbol{x}_i$ and $f(\boldsymbol{x}_i)$ denotes the predictive model. Logistic regression is one of the most popular models to estimate (1) [2, 6, 7, 33]. Other commonly employed techniques include tree-based learners [24, 43] and neural networks [15, 45].

To develop a fraud detection model, we rely on historical, labeled data of past observed fraud behavior [6]. This historical data is commonly derived from the expert judgment of previously investigated claims and serves as the foundation for constructing an effective fraud detection model. By training the fraud detection model on labeled claims, we aim to find hidden patterns that allow us to identify new fraudulent claims.

However, only a fraction of the historical claims are examined by fraud investigators and receive a label. To use both labeled and unlabeled claims, we can employ the approach of Óskarsdóttir et al. [37]. Here, we construct two different binary outcome variables to be used in a classification model. The first outcome variable distinguishes between labeled (i.e., fraudulent or non-fraudulent) and unlabeled (i.e., uninvestigated) claims. The second outcome variable is defined as fraudulent claims versus non-fraudulent and uninvestigated claims. Alternatively, we may rely on semi-supervised learning techniques to incorporate both labeled and unlabeled claims.

2.2 Enriching traditional claim characteristics with social network data

Both the expert-based and fraud analytics approach commonly rely on traditional claim characteristics stored in a tabular data set. To go beyond this tabular structure, we can rely on social network analytics which extracts information from the relational structure in the data set. As such, we augment the database with supplementary information on the social network structure of the claim and the involved parties. The involved parties are typically the policyholder and the experts involved in the claim [37]. Certain contracts involve the active participation of brokers, hereby incorporating them into the network structure. Further, depending on the type of insurance, other parties may be present as well. In motor insurance, for example, we commonly also have the auto repair shop that repaired the vehicle (hereafter referred to as the garage). In constructing the network structure, Óskarsdóttir et al. [37] take a holistic view by integrating information across multiple lines of business. In this paper, we focus exclusively on the social network structure within one specific insurance product.

Figure 1a depicts a toy example of a social network consisting of seven claims and seven involved parties. In this figure, the edges symbolize the connections between the claims and the parties. The claims and parties are represented by circles and the claims in the network are color-coded. Green claims correspond to non-fraudulent claims and fraudulent claims are colored red. There is a notable cluster of fraudulent claims (i.e., c_5, c_6 and c_7) that are strongly interconnected. Party p_7 is connected to all three fraudulent claims and might be the central figure in the criminal network. Via the claims, p_7 is connected to the fraudsters p_5 and p_6. In this example, all fraudsters are connected to each other via one or more fraudulent claims.

To obtain a mathematical representation of the network data, we follow the approach proposed in Óskarsdóttir et al. [37] and use a bipartite network of nodes $C \cup P$ and edges E. C denotes the set of all claims and P the set of all parties in the network. E is the set of edges that connect the nodes in C to the nodes in P. The bipartite network $G = (C \cup P, E)$ is undirected (i.e., there is no direction in the edges). We use c_i to denote an individual claim, where $i \in (1, \ldots, n_C)$ and n_C is the total number of nodes in C. p_j denotes an individual party. Here, $j \in (1, \ldots, n_P)$ where n_P is the total number of nodes in P. The network's edges are represented in a weight matrix $\boldsymbol{W}$ of dimension $n_P \times n_C$. Each individual edge carries a certain weight w_{ij} that reflects the strength of the relationship between claim i and party j. If $w_{ij} > 0$, claim c_i is connected to party p_j. We have an unweighted network when all nonzero values in $\boldsymbol{W}$ are equal to one.

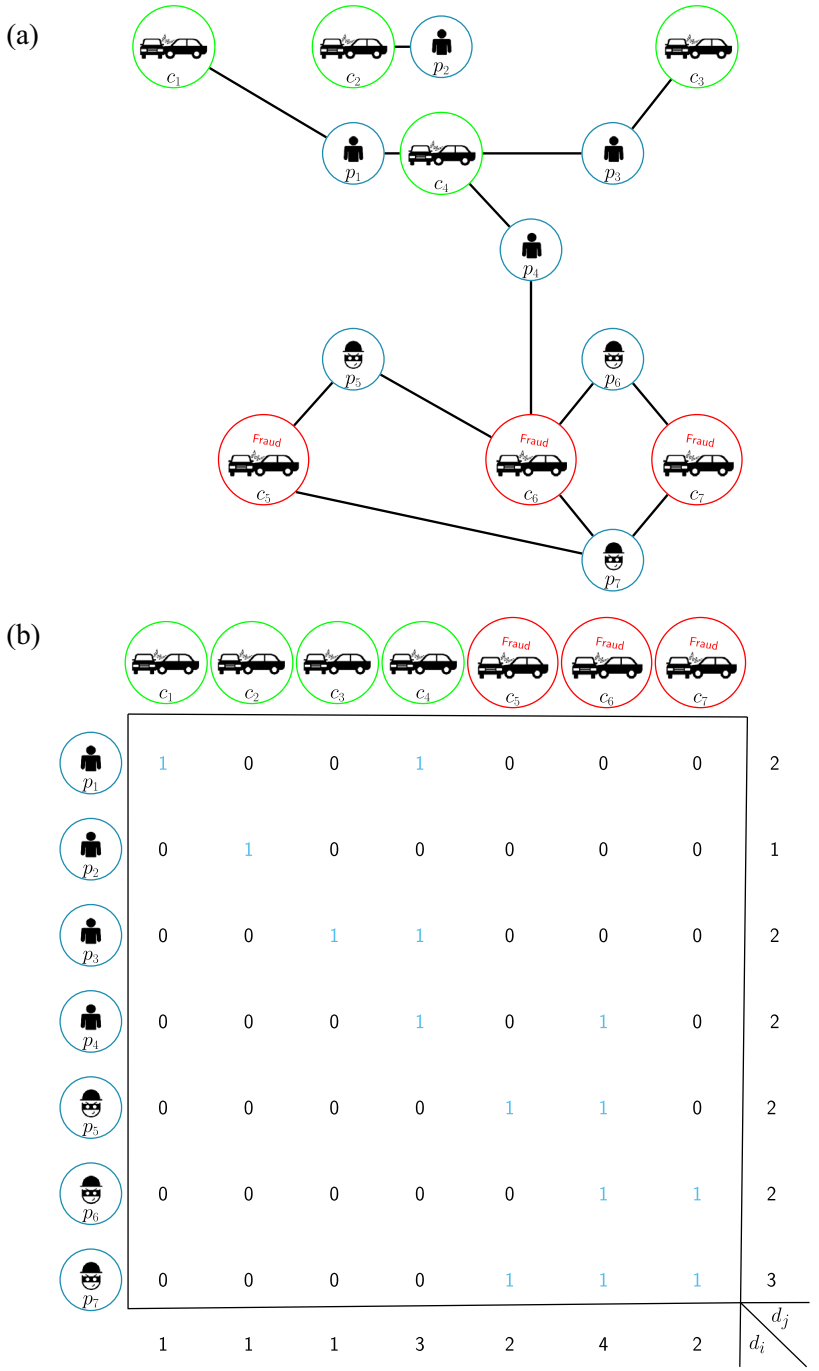

Fig. 1 A toy example of a social network in an insurance context

We refer to the set of nodes, connected to node c_i via a path of exactly k edges, as the k^{th} order neighborhood of c_i and we denote it as $\mathcal{N}_{c_i}^k$. Hence, the first-order neighborhood of a claim c_i consists of all involved parties

$$\mathcal{N}_{c_i}^1 = \{p_j | w_{ij} \neq 0\} \tag{2}$$

and the second-order neighborhood of c_i

$$\mathcal{N}_{c_i}^2 = \{c_k | p_j \in \mathcal{N}_{c_k}^1 \wedge w_{kj} \neq 0\} \setminus c_i. \tag{3}$$

are all the claims connected to the parties in $\mathcal{N}_{c_i}^1$. In an unweighted network, we refer to the number of nodes in a node's first order neighborhood as the degree of the node. For claim c_i, we denote this as d_i and d_j refers to the degree of party p_j. The degree of all claims is summarized in a $n_C \times n_C$ diagonal matrix $\boldsymbol{D}_C$, where $(\boldsymbol{D}_C)_{ii} = d_i \, \forall \, i$. Similarly, the $n_P \times n_P$ diagonal matrix $\boldsymbol{D}_P$ contains the degrees of all parties.

In this toy example of a social network, we use an unweighted network. Figure 1b represents the corresponding $\boldsymbol{W}$ that captures the connections in the example. The strong interconnectedness between fraudsters and fraudulent claims is also reflected in $\boldsymbol{W}$. In the lower right corner of $\boldsymbol{W}$ we notice a distinct cluster, which predominantly consists of fraudulent nodes and which reveals a web of fraudsters. We observe another group of connected claims in the upper left corner of $\boldsymbol{W}$. This cluster consists mostly of legitimate claims. Furthermore, there are only a few links connecting fraudulent and non-fraudulent nodes, indicating limited relationships between the two groups.

Homophily One of the fundamental concepts in a network-based fraud detection approach is the concept of homophily [6, 37]. This refers to the tendency of people to form social connections with individuals that are similar to themselves in some way [32]. Translated to an insurance context, this means that fraudulent claims are predominantly linked to other fraudulent claims, while non-fraudulent claims tend to be connected to other non-fraudulent claims. Moreover, fraudulent and non-fraudulent claims exhibit a weaker degree of connection with each other.

In criminology, it is a well-established fact that two-thirds of criminals work together to enhance their benefits and reduce the risk of detection [3, 42]. This phenomenon is particularly pronounced in organized crime, such as fraud, where social connections play a pivotal role. These connections provide access to accomplices and lucrative opportunities, while also fostering the trust necessary for effective collaboration [53].

By conceptualizing fraud as a social phenomenon, we can detect its presence and dynamics within a network. Assessing homophily in a network gives us a first indication whether fraudulent claims tend to cluster together. To assess whether there are patterns of homophily present in the network, we compute the dyadicity and heterophilicity of the network [6, 39]. Dyadicity measures the connectedness between nodes with the same label. The higher the dyadicity, the more densely connected the same-label nodes are, compared to what is expected based on a random network configuration. Conversely, heterophilicity assesses the degree of interconnection between nodes with different labels. Networks exhibit high heterophilicity when nodes with different labels show higher interconnectedness compared to what is expected by chance.

In a fraud context, the investigated claims can be labeled as fraudulent (1) or non-fraudulent (0). Claims that are uninvestigated have no label and are referred to as unlabeled. Hence, there are three different labels present in the data set. If our focus is on the identification of dense networks of fraudulent claims, we can adopt a one-versus-all classification strategy and group the unlabeled with the non-fraudulent claims. We denote the total number of fraudulent claims as ${}_1n_C$ and ${}_0n_C$ denotes the total number of non-fraudulent and unlabeled claims. Further, $n_C = {}_1n_C + {}_0n_C$. In this example, we have three types of relationships between claims or so-called dyads. That is: fraudulent claims connected to other fraudulent claims (1–1); fraudulent claims linked to non-fraudulent, unlabeled claims (1–0) and non-fraudulent, unlabeled claims connected to non-fraudulent, unlabeled claims (0–0). We use m_{11}, m_{10} and m_{00} to refer to the total number of dyads of each kind present in the network and $|E| = m_{11} + m_{10} + m_{00}$, where $|E|$ denotes the number of edges. ρ denotes the probability that two nodes are connected and is empirically calculated in the network as

$$\rho = \frac{2|E|}{n_C(n_C - 1)}. \tag{4}$$

If nodes are randomly connected to other nodes irrespective of their labels, the expected values of m_{11} and m_{10} equal [6]

$$\overline{m}_{11} = \frac{{}_1n_C({}_1n_C - 1)\rho}{2} \quad \text{and} \quad \overline{m}_{10} \quad = {}_1n_C(n_C - {}_1n_C)\rho. \tag{5}$$

We then calculate the dyadicity $\mathscr{D}$ and heterophilicity $\mathscr{H}$ of the network as

$$\mathscr{D} = \frac{m_{11}}{\overline{m}_{11}}, \tag{6}$$

$$\mathscr{H} = \frac{m_{10}}{\overline{m}_{10}}. \tag{7}$$

When $\mathscr{D} > 1$, the network is dyadic and fraudulent nodes are more densely connected to each other compared to what we expect by chance and $\mathscr{D} \approx 1$ corresponds to a random network configuration. Here, $\approx$ denotes approximately equal to. We have a heterophobic network if $\mathscr{H} < 1$, indicating that fraudulent claims have fewer connections to non-fraudulent claims than what is expected by chance. In a random network configuration, $\mathscr{H} \approx 1$. In our fictive example depicted in Fig. 1, the network is dyadic ($\mathscr{D} = 2.5$) and heterophobic ($\mathscr{H} = 0.28$). Consequently, we can infer that our network exhibits homophily as it displays both dyadicity ($\mathscr{D} > 1$) and heterophobia ($\mathscr{H} < 1$). In this toy example, engineering features from the social network potentially enables us to identify collaborating fraudsters that try to hide their tracks.

BiRank algorithm In a homophilic network (i.e., where $\mathscr{D} > 1$ and $\mathscr{H} < 1$), we can potentially uncover fraud by inspecting claims that are closer and more densely connected to known fraudulent claims. To evaluate the proximity to fraudulent claims, a suitable metric is needed. One effective approach is to employ the BiRank algorithm [19], as proposed in Óskarsdóttir et al. [37] This algorithm is an extension of the personalized PageRank algorithm [38] and is specifically designed for bipartite networks.

Using BiRank, we rank claims with respect to their exposure to known fraudulent claims [37].

The scores of nodes c_i and p_j are calculated iteratively as

$$c_i = \sum_{j=1}^{n_P} w_{ij} p_j \quad \text{and} \quad p_j = \sum_{i=1}^{n_C} w_{ij} c_i$$

where c_i and p_j denote the scores of nodes c_i and p_j, respectively. To ensure convergence and stability, the scores are normalized using

$$c_i = \sum_{j=1}^{n_P} \frac{w_{ij}}{\sqrt{d_i}\sqrt{d_j}} p_j \quad \text{and} \quad p_j = \sum_{i=1}^{n_C} \frac{w_{ij}}{\sqrt{d_i}\sqrt{d_j}} c_i. \tag{8}$$

This normalization lessens the contribution of high-degree nodes and gives better quality results [19]. To steer the scoring towards fraudulent claims, we incorporate query vectors in the scoring process. These query vectors encode our prior belief on the nodes' importance. We use $\boldsymbol{c}^0$ and $\boldsymbol{p}^0$ to denote the query vectors for the claims and parties, respectively. Further, c_i^0 and p_j^0 represent the individual entries in the vectors $\boldsymbol{c}^0$ and $\boldsymbol{p}^0$. We adjust (8) to

$$c_i = \alpha \sum_{j=1}^{n_P} \frac{w_{ij}}{\sqrt{d_i}\sqrt{d_j}} p_j + (1-\alpha) c_i^0 \quad \text{and} \quad p_j = \beta \sum_{i=1}^{n_C} \frac{w_{ij}}{\sqrt{d_i}\sqrt{d_j}} c_i + (1-\beta) p_j^0. \tag{9}$$

where $\alpha \in [0, 1]$ and $\beta \in [0, 1]$ are adjustable parameters. The values for α and β regulate the relative emphasis given to the network structure and the query vector. We rewrite (9) in matrix form

$$\boldsymbol{c} = \alpha \boldsymbol{S}\boldsymbol{p} + (1-\alpha)\boldsymbol{c}^0 \quad \text{and} \quad \boldsymbol{p} = \beta \boldsymbol{S}^T \boldsymbol{c} + (1-\beta)\boldsymbol{p}^0. \tag{10}$$

Here, $\boldsymbol{S} = \boldsymbol{D}_C^{-\frac{1}{2}} \boldsymbol{W} \boldsymbol{D}_P^{-\frac{1}{2}}$ denotes the symmetrically normalized weight matrix. We start the algorithm by randomly initializing the ranking vectors $\boldsymbol{c}$ and $\boldsymbol{p}$. Hereafter, we iteratively compute the node scores until convergence.

We encode information about known fraudulent claims into the query vector $\boldsymbol{c}^0$ to rank the nodes' scores towards fraudulent claims. When the claim is fraudulent, we set $c_i^0 = 1$ and $c_i^0 = 0$ otherwise. We define $\boldsymbol{p}^0 \equiv \boldsymbol{0}$, since only claims can be fraudulent and not parties. We set $\beta = 1$, since we do not include prior information on the parties. We adjust (10) to

$$\boldsymbol{c} = \alpha S \boldsymbol{p} + (1-\alpha)\boldsymbol{c}^0 \quad \text{and} \quad \boldsymbol{p} = S^T \boldsymbol{c}.$$

The iterative procedure to compute the fraud scores is summarized in Algorithm 1. The BiRank algorithm stops when the difference between two successive iterations is below a certain threshold or when we exceed the maximum number of iterations.

Algorithm 1: BiRank algorithm for computing fraud scores in a network of insurance claims and parties [37]. Adapted from Algorithm 1 in He et al. [19]. We omit the query vector $\boldsymbol{p}^0$ and set $\beta = 1$.

Input: Weight matrix $\boldsymbol{W}$, query vector $\boldsymbol{c}^0$ and hyperparameter $\alpha = 0.85$;
Output: Ranking vectors $\boldsymbol{c}$ and $\boldsymbol{p}$;

Symmetrically normalize $\boldsymbol{W}$: $\boldsymbol{S} = \boldsymbol{D}_P^{-\frac{1}{2}} \boldsymbol{W} \boldsymbol{D}_C^{-\frac{1}{2}}$;
Randomly initialize $\boldsymbol{c}$ and $\boldsymbol{p}$;
while *stopping criteria is not met* **do**
  $\boldsymbol{c} \leftarrow \alpha S \boldsymbol{p} + (1 - \alpha)\boldsymbol{c}^0$;
  $\boldsymbol{p} \leftarrow S^T \boldsymbol{c}$;
return $\boldsymbol{c}$ *and* $\boldsymbol{p}$;

Network featurization

Next, we engineer several social network features from the network structure, the labels and the scores resulting from the BiRank algorithm. These features capture the information that is embedded in the network of the claims and can be integrated into a tabular dataset alongside the individual characteristics of each claim. The social network features then represent an additional source of information that we can use in our fraud analytics models (see Sect. 2.1). We divide the network features into two groups, the fraud-score based features and the neighborhood based features (see Table 1). The results from the BiRank algorithm are used to compute the fraud-score based features. Here, we look at the claim's fraud score and the distribution of the fraud scores in, for instance, its first and second order neighborhood. To summarize these distributions, we can rely on (robust) central tendency measures such as the median or midmean[1] [50]. Further, we engineer neighborhood based features that capture the surrounding network structure of each claim. Here, we can compute the size of a claim's first and second neighborhood for instance.

2.3 Challenges within fraud analytics

When developing an analytic model for fraud detection, we encounter several challenges that are inherent to research on fraud. One of the main challenges is the infrequent nature of fraud, which leads to highly imbalanced data sets [6, 22, 49, 57]. This imbalance creates a bias towards the majority class for certain analytic techniques, resulting in compromised fraud detection performance. Within fraud research, we

[1] The midmean or interquartile mean is the average of the observations between the 25th and 75th percentiles.

Table 1 Fraud-score and neighborhood-based features, partially based on the feature engineering process from [37]

	Name	Order	Description
Fraud-score	`scores0`	0	The node's fraud score as determined via BiRank
	`n1.q1`	1	The first quartile of the empirical distribution of the fraud scores in the node's first order neighborhood
	`n1.med`	1	The median of the empirical distribution of the fraud scores in the node's first order neighborhood
	`n1.midmean`	1	The midmean (or interquartile mean) of the empirical distribution of the fraud scores in the node's first order neighborhood
	`n2.q1`	2	The first quartile of the empirical distribution of the fraud scores in the node's second order neighborhood
	`n2.med`	2	The median of the empirical distribution of the fraud scores in the node's second order neighborhood
	`n2.midmean`	2	The midmean (or interquartile mean) of the empirical distribution of the fraud scores in the node's second order neighborhood
Neighborhood	`n1.size`	1	The number of nodes in the node's first order neighborhood
	`n2.size`	2	The number of nodes in the node's second order neighborhood
	`n2.ratioFraud`	2	The number of known fraudulent claims in the node's second order neighborhood divided by `n2.size`
	`n2.ratioNonFraud`	2	The number of known non-fraudulent claims in the node's second order neighborhood divided by `n2.size`
	`n2.binFraud`	2	A binary value indicating whether there is a known fraudulent claim in the node's second order neighborhood

commonly organize the data into a tabular format, as this structure is typically required for predictive analytics [6]. This enables us to tackle the class imbalance problem by employing resampling techniques such as under- and over-sampling, SMOTE or ROSE [5, 37, 47, 48, 54]. A second challenge is the dynamic nature of fraud [5, 6, 57]. To remain undetected, fraudsters continuously adapt their behavior and tactics. Consequently, it is essential to detect fraud as soon as possible and to tweak or rebuild the fraud detection model when necessary. Hereto related is the computational efficiency of the fraud detection model [6, 57] which represents yet another challenge. With the rapid advancement of machine learning, new methods are constantly emerging, adding to the ever-growing repertoire of techniques available. Further, given the high cost of fraud, it is crucial to detect fraudulent activities instantly. Existing analytic methods for fraud detection are predominantly evaluated based on their accuracy, often overlooking the aspect of computational efficiency. For an accurate and reliable comparison of competing methods, it is essential to evaluate their performance on a set of benchmark data sets. Most studies, however, do not share their data sets due to

their sensitive nature. The lack of publicly available data hinders the reproducibility of research [5] and gives rise to a fifth challenge [40, 57]. The sixth and final challenge arises from the disproportionate misclassification cost and the specific performance criteria to evaluate the model [5, 57]. Analytic fraud models are commonly assessed using performance measures that evaluate the predictive performance. Notwithstanding, wrongly classifying claims has financial implications. Depending on the allocated budget for the fraud investigation process, wrongly classifying a fraudulent claim as legitimate can be considerably more expensive than the reverse. Consequently, it might be more appropriate to compare models in terms of their monetary performance [5].

3 Simulation engine

Our proposed simulation engine addresses one of the key challenges within fraud research. That is, the limited availability of publicly accessible data. In this section, we provide an in-depth overview of the data generation process and the architecture of the simulation engine. We outline the sequential steps taken to generate a realistic and representative insurance fraud network data set.

The resulting synthetic data set is structured in a tabular format, with each row representing a unique claim and its corresponding attributes. The columns in the data set capture items such as policyholder characteristics, traditional claim features, involved parties and social network features. These are the attributes typically used for fraud analytics. Further, for each claim we have two different types of labels. The first is the true label of the claim, indicating if it is fraudulent or not. This label is determined by the fraud generating model. The second type of label is the outcome of the fraud investigation, which can either take on the value fraudulent, non-fraudulent or uninvestigated. This variable is typically the one we have available in insurance fraud data sets.

Implementation

The simulation engine is available as open-source R package on Github at https://github.com/BavoDC/iFraudSimulator. An extensive overview of the default configuration is described in Appendix A as well as in the package's documentation.

Architecture

We generate a synthetic, tabular data set in seven consecutive steps (see Fig. 2). We start by generating the policyholder characteristics (step 1). Hereafter, we simulate the contract-specific attributes per policyholder (step 2). We use the policyholder and contract-specific features as input for our data-generating claim frequency model and generate the number of claims per contract (step 3). Next, we simulate the individual claim amounts using a data-generating claim severity model (step 4). Similarly to step 3, we use the policyholder and contract-specific characteristics as input for the data-generating model. Subsequently, we combine all simulated claims and their

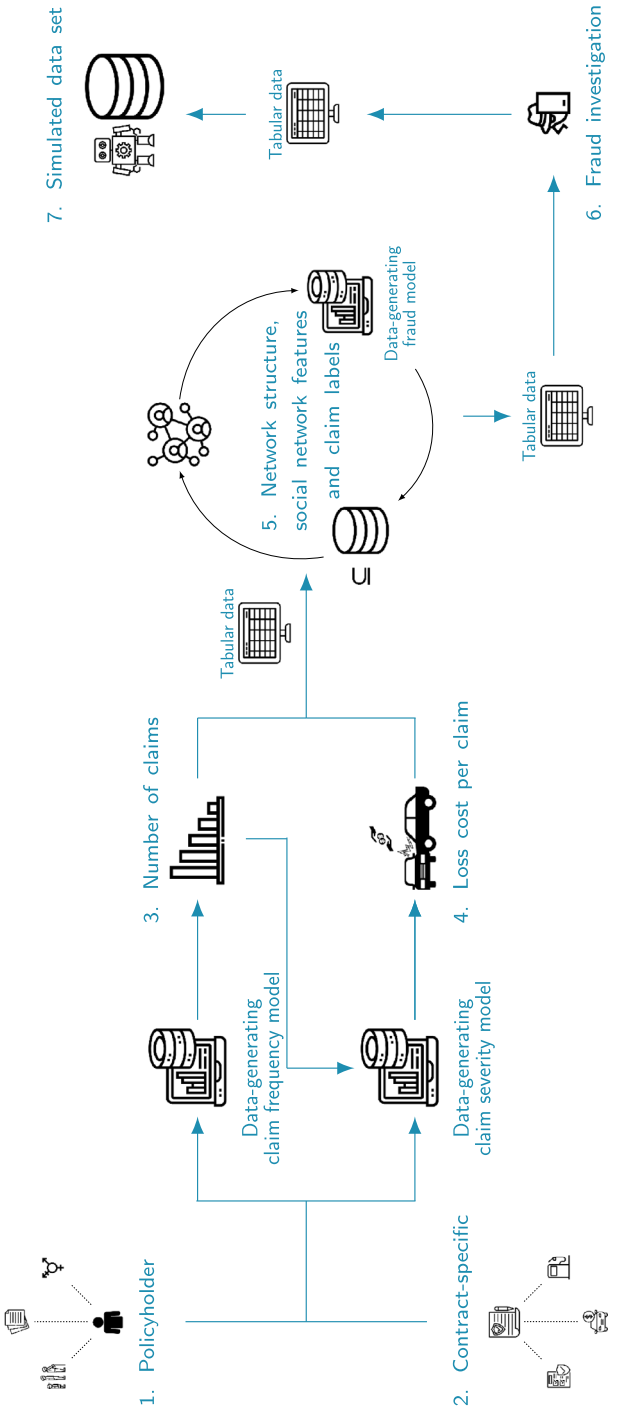

Fig. 2 Roadmap of the simulation engine

characteristics into a tabular data set and we proceed with the observations that have at least one claim. In step 5, we generate the network structure of the claims by connecting each claim to different types of parties. Next, we engineer the social network features and generate the claim labels which represent the ground truth of the claim (i.e., fraudulent or non-fraudulent). We replicate the fraud investigation process in step 6. Hereby, we generate the label that is commonly available in fraud data sets (see Sect. 2). This label expresses whether the claim has been investigated for fraud and what the outcome of that investigation was. We conclude the synthetic data generation with step 7 where we merge all simulated data.

The simulation engine offers a range of customizable features. In all seven steps, several parameters can be adjusted. Moreover, it allows for dependencies between certain features. For example, we can specify a negative dependence between the age and the value of the car. Consequently, older cars will be characterized by a lower value. By allowing for dependencies between features, the synthetic data captures more realistic and nuanced relationships among variables and more closely mirrors real-world data sets. Following, we discuss the seven consecutive steps in detail.

3.1 Policyholder and contract-specific characteristics

Prior to the synthetic data generation, the user can adjust several parameters that determine the characteristics of the synthetic data (see Appendix A for the default configuration). Hence, the user has full control over the data-generating mechanics. One of the adjustable parameters is the number of policyholders n_{ph}, which determines the size of the resulting data set. By default, $n_{ph} = 10\,000$. We use $i = 1, \ldots, n_{ph}$ as an index for the policyholders. In addition to n_{ph}, the user can also specify the total number of experts n_{exp}, garages n_{gar}, brokers n_{bro} and other person(s) involved in the claim n_{per}. These parameter values will govern the size of the social network.

Prior to generating the synthetic data set, we set the seed to ensure reproducibility. This guarantees that all random number generators produce identical results when we use the same seed. Once the seed is set, we start the synthetic data generation by simulating the policyholder characteristics for n_{ph} policyholders (step 1 in Fig. 2). Here, we generate features such as the age and gender of the policyholder and the number of contracts. We use $\texttt{NrContractsPH}_i$ to denote the number of contracts of policyholder i and use $j = 1, \ldots, \texttt{NrContractsPH}_i$ as an index for the contracts. Per policyholder, we also generate the number of years since the inception of the first contract and refer hereto as the exposure w_i. By default, we set the average exposure to five years and the maximum to 20 years. Additionally, we simulate the contract-specific exposure w_{ij}. In the synthetic data set, we consolidate the multiple years of coverage into a single contract to simplify the data structure and analysis. That is, by default we allow $w_{ij} > 1$. Table 2 gives an overview of the different attributes that are generated. The first column in this table depicts the variable name, the second the variable type and the third column contains the feature description. The last column of Table 2 specifies which generator is used to simulate the feature values. For certain features, the user can specify the range of the feature values. When generating the data, we ensure that all simulated values fall within this prespecified range (see Appendix B).

Table 2 The policyholder and contract-specific characteristics, along with the generator used to simulate the feature values This table shows the default settings. Parameters and parameter values that can be adjusted by the user are underlined. For certain variables, denoted with an uppercase star preceding the variable name, we adjust the range to prevent generating implausible values (see Appendix B)

	Variable	Type	Description	Generator
Policyholder	IDPH	Continuous	Unique ID to identify the policyholder (index $i = 1, \ldots, \underline{n_{ph}}$)	
	*AgePH	Continuous	Age of the policyholder in years. Default range is [18, 80]	$\mathcal{N}(\underline{40}, \underline{15})$
	GenderPH	Categorical	Gender of the policyholder: female ($u_i \leq \underline{0.28}$), male ($u_i > \underline{0.29}$) or non-binary ($\underline{0.28} \leq u_i \leq \underline{0.29}$)	$u_i \sim U(0, 1)$
	*ExpPH	Continuous	Time since inception of the first contract of the policyholder, in years	$\mathcal{N}(\underline{5}, \underline{1.5})$
	RateNrContracts	Continuous	Rate parameter λ_i for generating the number of contracts	$\lambda_i = 0.25(1.05 - 2.5 \times 10^{-6} \times \texttt{AgePH}_i + 0.0025 \times \texttt{AgePH}_i^2 - 2.65 \times 10^{-5} \times \texttt{AgePH}_i^3)$
	*NrContractsPH	Ordinal	Number of contracts. Default range is [1, 5]	$\text{Poi}(\lambda_i)$
Contract-specific	ContractID	Continuous	Unique ID to identify the contract (index $j = 1, \ldots, \texttt{NrContractsPH}_i$)	
	ExpPHContracts	Continuous	Duration or exposure of the contract, in years	if $\texttt{NrContractsPH}_i > 1$ $\texttt{ExpPH}_i - \text{U}(0, \texttt{ExpPH}_i/2)$ else $\texttt{ExpPHContracts}_{ij} = \texttt{ExpPH}_i$
	AgeCar	Continuous	Age of the vehicle in years	$\max(\mathcal{N}(7.5, \sqrt{5}), \texttt{ExpPHContracts}_{ij})$
	OrigValueCar	Continuous	Original value of the vehicle	$\text{Exp}(\lambda_i / \texttt{NrContractsPH}_i)\upsilon$
	ValueCar	Continuous	Current value of the car	$\texttt{OrigValueCar}_{ij}(1 - \delta)^{\texttt{AgeCar}_{ij}}$
	Coverage	Categorical	Type of coverage provided by the insurance company	$\text{Multinomial}(1, \pi_{\text{TPL}}, \pi_{\text{PO}}, \pi_{\text{FO}})$

Table 2 continued

	Variable	Type	Description	Generator
			`TPL` = only third party liability `PO` = partial omnium = TPL + limited material damage `FO` = full omnium = TPL + comprehensive material damage	(see Appendix C)
	`Fuel`	Categorical	Type of fuel of the vehicle: `Gasoline/LPG/Other` (0) or `Diesel` (1)	Bernoulli(0.3)
	`BonusMalus`	Ordinal	Level occupied in bonus-malus scale of the insurance company	$\lfloor \mathcal{G}(1, 1/3) \rfloor$
Claim	`ClaimAge`	Integer	Number of months from beginning of contract to the date of the incident	$\min(\lfloor \text{Exp}(0.25) \rfloor, \lfloor \texttt{ExpPHContracts}_{ij} * 12 \rfloor)$
	`ClaimDate`	Continuous	Number of years between the start of the contract and the claim's filing date	$\max(U(0, \texttt{ExpPHContracts}_{ij}), \texttt{ClaimAge}_{ijk}/12)$
	`Police`	Categorical	Whether police was called when the incident happened: `no` (0) or `yes` (1)	Bernoulli(0.25)
	`nPersons`	Integer	Number of other persons involved in the claim (see Sect. 3.3). Range is [0, 5]	$S \xleftarrow{\pi_p} x$

$\mathcal{N}(\mu, \sigma)$ denotes the normal distribution with mean μ and standard deviation σ, U the uniform distribution, Poi the Poisson distribution, $\mathcal{G}$ the gamma distribution and Exp denotes the exponential distribution. For the gamma distribution $\mathcal{G}(\alpha, \tau)$, we use the parameterization with the density function $f(x) = \tau^\alpha x^{\alpha-1} \exp(-\tau x)/\Gamma(\alpha)$. $\upsilon = 25 \times 10^3$ if $\texttt{GenderPH}_i = \texttt{male}$, $\upsilon = 20 \times 10^3$ if $\texttt{GenderPH}_i = \texttt{female}$ and $\upsilon = 22.5 \times 10^3$ if $\texttt{GenderPH}_i = \texttt{non-binary}$. δ is the depreciation rate (see Table 3). We use $\lfloor \cdot \rfloor$ to depict the floor function. $S \xleftarrow{\pi_p} x$ denotes that a random sample x is drawn from the set $S = (0, 1, 2, 3, 4, 5)$ with corresponding probability $\pi_p = (0.02, 0.57, 0.19, 0.10, 0.10, 0.02)$

Table 3 Overview of the dependencies between the variables

Variables	Dependency
`AgePH` and `GenderPH`	Weak negative dependence, introduced using AMH copula with $\theta = -0.15$
`AgePH` and `ExpPH`	Weak positive dependence, introduced using AMH copula with $\theta = 0.15$
`AgePH` and `NrContracts`	Convex function (see Table 2) and positive dependence between `AgePH` and `NrContracts`, introduced using AMH copula with $\theta = 0.95$
`AgeCar` and `OrigValueCar`	Negative dependence, introduced using Frank copula with $\theta = -25$
`OrigValueCar` and `ValueCar`	The depreciation rate $\delta = 0.15$ for cars whose original value $< 30\,000$ and $\delta = 0.075$ when the original value $\geq 30\,000$ (see Table 2 and [46])
`Coverage` and `ValueCar`, `AgeCar`, `AgePH`	A dependency is introduced using a multinomial logistic regression model (see Appendix C)

Hereby, we avoid generating implausible or invalid values. For the policyholder's age, for example, the default range is [18, 80]. Consequently, none of the policyholders will be younger than 18 (the legal driving age in many countries) or older than 80. Once all policyholder characteristics are generated, we proceed to simulate the contract-specific features such as the age of the car and the type of coverage (see Table 2).

Dependence structure

The simulation engine allows to specify a dependency between different features. To generate the dependency, we rely on copulas [8] and we use θ to denote the parameter that controls the dependence. In our simulation engine, we restrict ourselves to the bivariate Ali–Mikhail–Haq (AMH) [26] and Frank copula [8]. We choose the aforementioned copulas for their ability to model both positive and negative dependence. As such, they serve the requirements of our simulation engine, to generate a dependency between features. If other copula classes are desired, the user has to modify the source code.

Table 3 presents an overview of the dependencies and the method used to incorporate them. Within insurance, we commonly have variables that are correlated [16]. Consequently, by allowing for dependencies, we can create a more realistic data set. For example, in real life data sets we commonly observe that older cars are worth less compared to newer cars. In our engine, we incorporate this negative dependency between the age of the car and its value by using a Frank copula with $\theta = -25$.

3.2 Claim frequency and claim severity

Next, we proceed to simulating the number of claims N_{ij} for policyholder i on contract j and the individual claim costs L_{ijk}. We use $k = (1, \ldots, N_{ij})$ as an index for

the claims. Hereto, we employ the frequency-severity approach [12, 35] where we model the claim frequency and claim severity separately. To simulate the number of claims and the claim costs as a function of the policyholder and contract-specific characteristics, we rely on the generalized linear model (GLM) framework [30]. We use a Poisson GLM with log link as the data-generating model for the number of claims [35, 41]

$$N_{ij} \sim \text{Poi}(w_{ij} \exp({}_{cf}\boldsymbol{x}_{ij}^{\top}\boldsymbol{\beta}_{cf})). \tag{11}$$

To generate N_{ij}, we take a random draw from $\text{Poi}(w_{ij} \exp({}_{cf}\boldsymbol{x}_{ij}^{\top}\boldsymbol{\beta}_{cf}))$, with ${}_{cf}\boldsymbol{x}_{ij}$ the covariate vector and $\boldsymbol{\beta}_{cf}$ is the corresponding parameter vector. The exposure w_{ij} of the contract is included as an offset term and the subscript cf stands for claim frequency. In our simulation engine, the user can specify which features should be included in ${}_{cf}\boldsymbol{x}_{ij}$ and the features' effect size can be adjusted via $\boldsymbol{\beta}_{cf}$. As such, the user can control the relation between the N_{ij}'s and the policyholder and contract-specific characteristics (see Appendix D for the default specification of the claim frequency model).

Hereafter, we generate the claim-specific characteristics (see Table 2). For example, we simulate the duration in months since the beginning of the contract until the date of the incident. Hereby, we create the claim-specific information that is typically available in fraud insurance data sets and that is used in fraud detection models (see, for example, Óskarsdóttir et al. [37]).

Next, we proceed with generating the cost L_{ijk} of claim k under contract j of policyholder i. The data-generating model for the claim amounts L_{ijk} is driven by a gamma GLM with log link

$$L_{ijk} \sim \mathcal{G}(\alpha, \alpha/\exp({}_{cs}\boldsymbol{x}_{ij}^{\top}\boldsymbol{\beta}_{cs} + N_{ij}\zeta)), \tag{12}$$

where $\mathcal{G}$ denotes the gamma[2] distribution and α the shape parameter, which we set to 0.25. The subscript cs stands for claim severity and the parameter ζ controls the dependency between the claim frequency and claim severity [11, 14]. Within the frequency-severity approach, we commonly assume that $\zeta = 0$ (i.e., the claim frequency and claim severity are independent). We specify 50 as a lower limit for L_{ijk} to prevent generating implausible low claim amounts. Consequently, when $L_{ijk} < 50$ we replace it with a randomly drawn value from $U(50, 150)$ to ensure that a low yet realistic claim amount is generated. As with the data-generating claim frequency model (see Eq. 11), we can specify which features are included in ${}_{cs}\boldsymbol{x}_{ij}$ as well as their effect size via $\boldsymbol{\beta}_{cs}$ (see Appendix D for the default model specification).

3.3 Constructing the social network structure and simulating fraudulent claims

Our next objective is to generate the social network structure that links claims to parties and to other claims. Within motor insurance each claim is linked to a policyholder and a garage. Other parties involved in the claim may include brokers and persons other

[2] For the gamma distribution, we use the parameterization with the density function $f(x) = \tau^{\alpha}x^{\alpha-1}\exp(-\tau x)/\Gamma(\alpha)$ where $\tau_i = \alpha/\exp({}_{cs}\boldsymbol{x}_{ij}^{\top}\boldsymbol{\beta}_{cs} + N_{ij}\zeta)$ and $\Gamma(\cdot)$ denotes the gamma function.

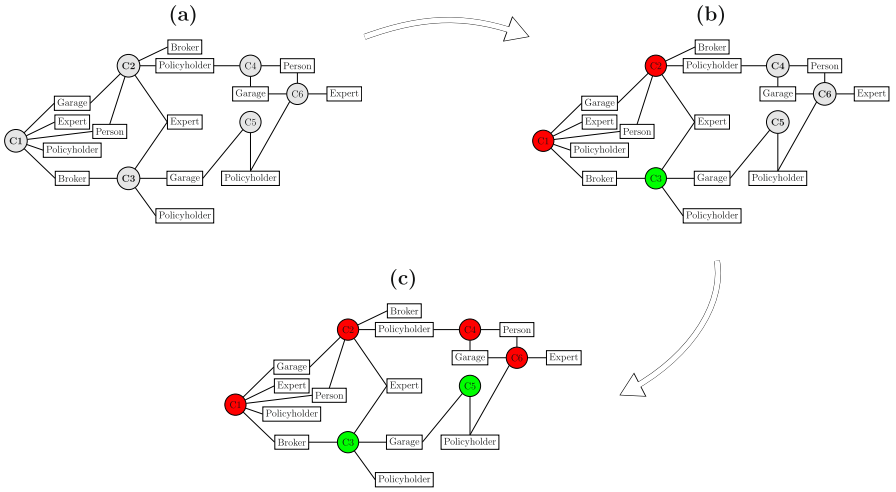

Fig. 3 Example of a social network structure, which illustrates the desired connectivity we want to obtain in our synthetic data set. Each claim is linked to specific parties, and as a result, claims that share the same party are connected to each other in the network. The rectangles depict the parties and the circles depict the claim. Red claims are fraudulent claims, green claims are legitimate claims and the gray claims represent unlabeled claims

than the policyholder. Experts are involved in the process only when the claim amount exceeds a certain threshold [23]. Insurance companies commonly handle minor losses without the involvement of an expert to inspect the damage or injury. Our goal is to enhance the simulated claims with a network structure similar to the one depicted in Fig. 3a. In this figure, the circles depict claims and the rectangles parties.

To accomplish this, we create a set A_p for each party type, such as the set of garages A_g, brokers A_b, experts A_e, and other persons A_o. These sets represent the specific parties of each type within the larger set of all possible parties, denoted as $P = A_g \cup A_b \cup A_e \cup A_o$. The size of a specific set A_p is determined by the corresponding user-specified parameter (see Sect. 3.1 and Appendix A). For example, if the number of garages n_{gar} is set to 150, the simulation engine will create a set A_g of 150 unique garages. For each claim, we then randomly select one (or multiple when we connect the claim to other persons, see Table 2) member from A_p to link the claim to a specific party. By repeating this procedure for each type of party, we create a social network structure where every claim is connected to different (types of) parties (see Fig. 3a). As a rule, we do not link the claim to an expert when $L_{ijk} < 250$ [23].

Next, we proceed with generating the claim label. The data-generating fraud model is a logistic regression model

$$Y_{ijk} \sim \text{Bern}\left(\pi_{ijk}\right) \quad \text{and} \quad \pi_{ijk} = \frac{e^{0\beta_f + f\boldsymbol{x}_{ijk}^\top \boldsymbol{\beta}_f}}{1 + e^{0\beta_f + f\boldsymbol{x}_{ijk}^\top \boldsymbol{\beta}_f}}. \tag{13}$$

Here, Bern denotes a Bernoulli distribution and Y_{ijk} is a binary variable indicating if the k^{th} claim of the j^{th} contract of policyholder i is fraudulent ($Y_{ijk} = 1$) or not ($Y_{ijk} = 0$). The subscript f stands for fraud and ${}_0\beta_f$ is the intercept term. The relation between the fraud label and the features is adjusted via the covariate vector ${}_f\boldsymbol{x}_{ijk}$ and the parameter vector $\boldsymbol{\beta}_f$ (see Appendix E for the default specification).

We generate the claim labels in an iterative manner and this process is visualized in Fig. 3. In this figure, the label of the claims is color-coded. Gray stands for unlabeled, red for fraudulent and green for non-fraudulent. Panel a represents the network at initialization, when all claims are unlabeled. We have no fraud-related information at this point and hence, no values for the social network features that rely on this information (e.g., the ratio of known fraudulent claims in the second order neighborhood). Consequently, at initialization, we remove all fraud-score and neighborhood based features (see Table 1) from ${}_f\boldsymbol{x}_{ijk}$ and $\boldsymbol{\beta}_f$ in (13). Next, we take a random subset, equal in size to 1% for example, of all simulated claims. By deliberately taking a small subset of the data, we ensure that only a limited proportion of the claim labels is generated without effect of the fraud-score and neighborhood based features. In Fig. 3a, this subset consists of claims C1, C2 and C3. To generate the claim labels, we take a random draw from Bern(π_{ijk}) (see (13)). As such, we generate our first set of labeled claims (see Fig. 3b). This enables us to compute the values for the fraud-score and neighborhood based features. Consequently, from here on out, we include these features in ${}_f\boldsymbol{x}_{ijk}$ and $\boldsymbol{\beta}_f$. To generate the labels of the remaining claims, we again take a random subset of unlabeled claims. In Fig. 3b, this subset corresponds to claims C5 and C6. The size of this random subset can be set by the user. By default, this is equal to 10% of all simulated claims. We combine this subset with the unlabeled claims in the 2^{nd} order neighborhood of the fraudulent claims in the previous iteration (i.e., C4 in Fig. 3b). In doing so, we ensure that every subset includes unlabeled claims that are connected to fraudulent claims and that we propagate fraud through the network. We engineer the social network features for all claims in the subset and we take random draws from Bernoulli(π_{ijk}) to generate the claim labels (Fig. 3c).

Algorithm 2 is a generalization of the process illustrated in Fig. 3. We use this iterative algorithm to simulate the claim labels Y_{ijk} and each iteration consists of three steps. First, we take a random subset of the unlabeled claims and we combine this subset with all unlabeled claims in the 2^{nd} order neighborhood of the fraudulent claims in the previous iteration. Second, we engineer the social network features for all claims in the subset. Third, we take random draws from Bernoulli(π_{ijk}) to generate the claim labels (Fig. 3c). This concludes one iteration and we repeat the algorithm until all claims are labeled.

The user can specify which features are included in ${}_f\boldsymbol{x}_{ijk}$ and determine their effect size in $\boldsymbol{\beta}_f$. Consequently, the user has the flexibility to activate or deactivate specific feature effects and to control their strength. By including social network features in ${}_f\boldsymbol{x}_{ijk}$ and via the specified effect size in $\boldsymbol{\beta}_f$, for example, we determine to which extent the network exhibits patterns of homophily. The greater the corresponding effect size in $\boldsymbol{\beta}_f$, the more densely connected fraudulent claims will be. Conversely, we can turn off the social network effect by omitting the social network features from ${}_f\boldsymbol{x}_{ijk}$. Further, we can set the desired level of class imbalance. Hereto, our simulation engine

Algorithm 2: Iterative algorithm to simulate the claim labels

Model: $Y_{ijk} \sim \text{Bern}(\pi_{ijk})$
Initialization: Remove fraud-score and neighborhood based features from $({}_f\boldsymbol{x}_{ijk}, \boldsymbol{\beta}_f)$ in the first iteration and generate the initial claim labels using (13)
repeat
1 Take a subset of the simulated database: a random sample of unlabeled claims combined with the unlabeled claims in 2^{nd} order neighborhood of fraudulent claims in the previous iteration;
2 Construct the social network features for the claims in this subset;
3 Generate the claim label using the data-generating logistic regression model in (13) ;
until *all claims are labeled*;

determines which value for ${}_0\beta_f$ results in the target class imbalance (see Appendix E for detailed information).

3.4 Replicating the expert-based fraud detection approach

In a real life fraud data set, we typically have historical, labeled data that is the result of an investigation by a fraud expert (see Sect. 2.1). In our simulation engine, we replicate the two steps of this fraud detection approach to obtain these labels. First, we flag claims as suspicious based on a set of business rules which can be defined by the user. Hence, an alert will be triggered for claims that meet the criteria outlined in the business rules. By default, we flag claim k under contract j of policyholder i as suspicious if it satisfies one of the following criteria: a) the claim is filed within one year of the most recent claim (i.e., $\texttt{ClaimDate}_{ijk} - \texttt{ClaimDate}_{ij(k-1)} \leq 1$); b) the individual claim amount $L_{ijk} > 75\%$ of $\texttt{ValueCar}_{ijk}$ or c) the cumulative claim amount $\sum_{l=1}^{k} L_{ijl} > 200\%$ of $\texttt{ValueCar}_{ijk}$. In reality, these claims are passed to an expert who performs an in-depth investigation. Following the investigation, the expert judgement determines whether the claim is legitimate or not. We simulate the expert judgement in the second step as follows. For a claim that is flagged by the business rules in step one, we first look at its ground truth label Y_{ijk}. If $Y_{ijk} =$ `non-fraudulent`, we take a random draw from $Y_{ijk}^{\texttt{expert}} \sim \text{Bern}(0.01)$. Hence, when the claim is legitimate, we have a 99% probability that the expert will label the claim as non-fraudulent. Conversely, if $Y_{ijk} =$ `fraudulent`, we randomly draw from $Y_{ijk}^{\texttt{expert}} \sim \text{Bern}(0.99)$. Thus, for fraudulent claims, there is a 99% probability that the expert will classify them as fraudulent as well. Further, claims that are not flagged by the business rules obtain the label `uninvestigated`. Hereby, we create all three labels that are typically available in an insurance fraud data set: `non-fraudulent`, `fraudulent` or `uninvestigated`. By following the procedure as outlined above, we acknowledge and reflect the inherent missing information and uncertainties that exist in real-life data. That is, the expert-based approach is not entirely infallible [6]. Claims that are judged to be non-fraudulent by the investigation may in reality be fraudulent and vice versa. In addition, we acknowledge and replicate the phenomenon of having a substantial proportion of uninvestigated claims that are unlabeled.

4 Generating synthetic fraud network data: illustrations

In this section, we illustrate the capabilities of our simulation engine. We specifically highlight the impact of social network features on the resulting synthetic data sets. Hereto, we generate and analyze two different types of data sets. In one data set we include a moderately strong social network effect and in another one we exclude it. Additionally, we provide an illustrative example of the construction and evaluation of a fraud detection model using a synthetically generated data set. We explore to which extent the constructed model is able to identify fraudulent claims that are not investigated and labeled by the expert.

4.1 The impact of social network features on the synthetic data

We generate two different types of data sets. In the first type of data set we introduce a moderately strong social network effect in the claim label generation (see Sect. 3.3). We denote this type of data set as $\mathscr{D}^{Network}$. Table 4 depicts the specification of the effect sizes used in the simulation of $\mathscr{D}^{Network}$. We include various types of features to generate a realistic and representative synthetic data set. These features encompass the policyholder, claim-specific, and social network characteristics. In order to replicate the social dynamics of fraud, we assign a strong effect size for the social network features `n1.size`, `n2.size` and `n2.ratioFraud`. Hereby, we create a synthetic data set where the network structure exhibits patterns of homophily. Conversely, in the second type of data set $\mathscr{D}^{Non-network}$, we exclude all network-related features from the data-generating fraud model (see Table 4). As such, we create a data set where fraud is not influenced by social interactions or network dynamics. The claim label generation is solely driven by policyholder and claim-specific characteristics. The selected set of policyholder and claim-specific features is identical in $\mathscr{D}^{Network}$ and $\mathscr{D}^{Non-network}$, as well as the effect sizes of these features (see Table 4).

For both types of data sets, the number of policyholders is set to 200,000 and the target class imbalance (i.e., the ratio of the number of fraudulent claims to the total number of claims) to 2%. All other settings remain at their default values (see Appendix A). We generate 100 data sets of each type and explore the distribution of the claim labels across these simulated data sets. Hereto, we calculate the frequency and relative frequency of the Y_{ijk} categories (i.e., `fraudulent` and `non-fraudulent`) and $Y_{ijk}^{\texttt{expert}}$ (i.e., `fraudulent`, `non-fraudulent`, and `uninvestigated`) in each data set. The average, minimum, and maximum values for both the frequency and relative frequency are computed and presented in Table 5. In all synthetic data sets, the empirical class imbalance is nearly identical to the target class imbalance. The minimum class imbalance in $\mathscr{D}^{Network}$ is 1.97% and the maximum 2.06%. In $\mathscr{D}^{Non-network}$, the minimum is 1.98% and the maximum is 2.04%. The class imbalance is also reflected in the expert judgement. Only a small fraction of claims are subject to investigation (approximately 9%), and among those investigated, only a minority are found to be fraudulent. Further, the empirical distributions of both Y_{ijk} and $Y_{ijk}^{\texttt{expert}}$ are similar across all simulated data sets.

Table 4 Specification of the data-generating fraud model in $\mathscr{D}^{Network}$ and $\mathscr{D}^{Non-network}$ We generate the claim label Y_{ijk} by taking a random draw from Bern (π_{ijk}) where $\pi_{ijk} = \exp({}_0\beta_f + {}_f\boldsymbol{x}_{ijk}^{\top}\boldsymbol{\beta}_f)(1 + \exp({}_0\beta_f + {}_f\boldsymbol{x}_{ijk}^{\top}\boldsymbol{\beta}_f))^{-1}$

Feature	β_f	
	$\mathscr{D}^{Network}$	$\mathscr{D}^{Non-network}$
Policyholder:		
`AgePH`	−2.00	−2.00
`NrContractsPH`	−1.50	−1.50
Claim-specific:		
`ClaimAmount`	0.20	0.20
`ClaimAge`	−0.35	−0.35
Social network:		
`n1.size`	2.00	0
`n2.size`	−2.00	0
`n2.ratioFraud`	3.00	0

Table 5 The average, minimum and maximum frequency and relative frequency (%) of the ground truth and expert-based claim labels across the 100 synthetic data sets

		Frequency (%)		
		Average	Minimum	Maximum
$\mathscr{D}^{Network}$	Y_{ijk} :			
	Fraudulent	2 175.37 (2.01%)	2 121.00 (1.97%)	2 219.00 (2.06%)
	Non-fraudulent	105 937.66 (97.99%)	105 103.00 (97.94%)	106 655.00 (98.03%)
	$Y_{ijk}^{\texttt{expert}}$:			
	Fraudulent	284.28 (0.26%)	260.00 (0.24%)	318.00 (0.30%)
	Non-fraudulent	9 149.38 (8.46%)	8 952.00 (8.30%)	9 431.00 (8.67%)
	Uninvestigated	98 679.37 (91.27%)	98 021.00 (91.08%)	99 428.00 (91.45%)
$\mathscr{D}^{Non-network}$	Y_{ijk} :			
	Fraudulent	2 175.84 (2.01%)	2 139.00 (1.98%)	2 209.00 (2.04%)
	Non-fraudulent	105 921.40 (97.99%)	104 978.00 (97.96%)	106 789.00 (98.02%)
	$Y_{ijk}^{\texttt{expert}}$:			
	Fraudulent	293.64 (0.27%)	254.00 (0.24%)	335.00 (0.31%)
	Non-fraudulent	9 137.37 (8.45%)	8 924.00 (8.29%)	9 454.00 (8.72%)
	Uninvestigated	98 666.23 (91.28%)	97 826.00 (91.01%)	99 366.00 (91.45%)

Empirical distribution of the features

Figures 4 and 5 present the empirical distribution of the features included in the data-generating fraud models (see Table 4) of one synthetically generated data set. Figure 4 displays the features' empirical distribution in a simulated data set $\mathscr{D}^{Network}$. Figure 5 shows this in a synthetic data set $\mathscr{D}^{Non-network}$. The empirical distribution

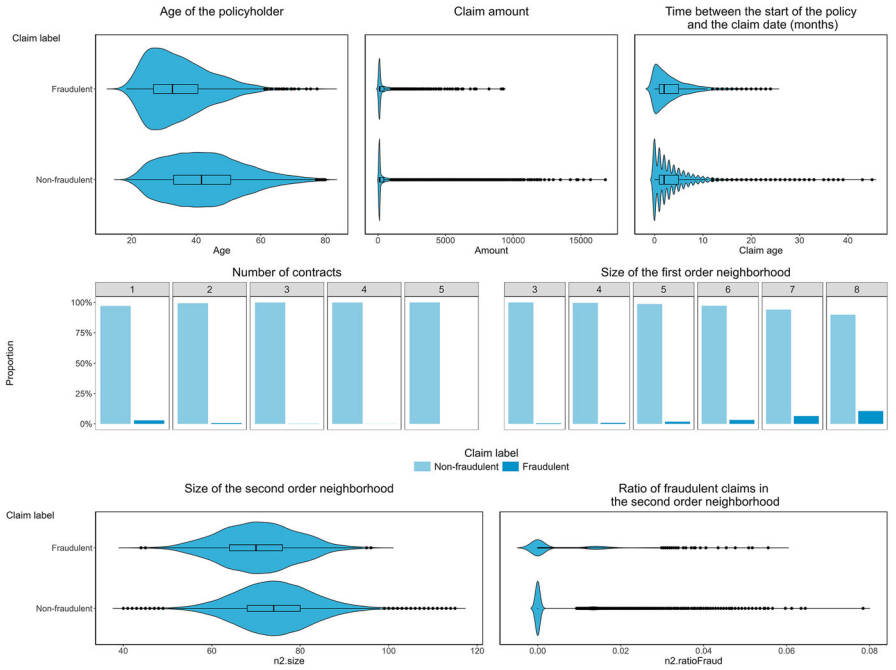

Fig. 4 Illustration of the features' empirical distribution in a synthetically generated $\mathscr{D}^{Network}$

of policyholder and claim-specific features is different across fraudulent and non-fraudulent claims in both types of data sets. For example, fraudulent claims are mostly associated with younger policyholders (top left plot on Figs. 4 and 5). Further, in $\mathscr{D}^{Network}$, the difference in the empirical distribution between fraudulent and non-fraudulent claims is also present for the social network features `n1.size`, `n2.size` and `n2.ratioFraud`. This suggests that the claim labels are linked to these features in $\mathscr{D}^{Network}$. In contrast, the empirical distributions of the social network features do not differ in $\mathscr{D}^{Non-network}$, indicating that there is no association between these features and the claim label.

Homophily

Figure 6 illustrates the dyadicity $\mathscr{D}$ and heterophilicity $\mathscr{H}$ observed in the simulated data sets (see Sect. 2.2). In $\mathscr{D}^{Network}$, the fraudulent claims are more densely connected to each other compared to what we expect by chance ($\mathscr{D} > 1$). In addition, fraudulent claims have fewer connections to non-fraudulent claims relative to what we expect by chance ($\mathscr{H} < 1$). In comparison, we observe no patterns of homophily in $\mathscr{D}^{Non-network}$. Both the dyadicity ($\mathscr{D} \approx 1$) and heterophilicity ($\mathscr{H} \approx 1$) correspond to values indicative of a random network configuration.

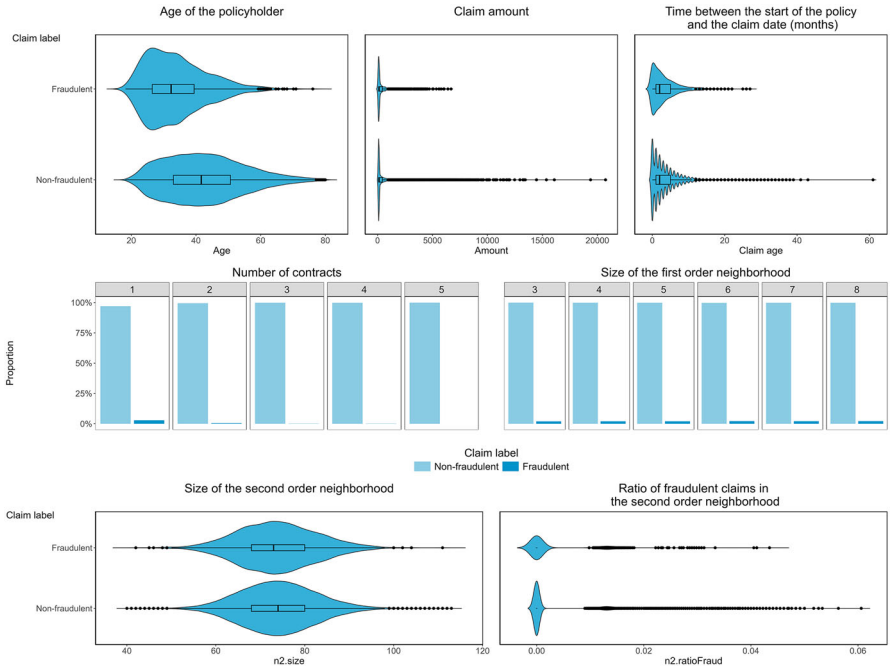

Fig. 5 Illustration of the features' empirical distribution in a synthetically generated $\mathscr{D}^{Non-network}$

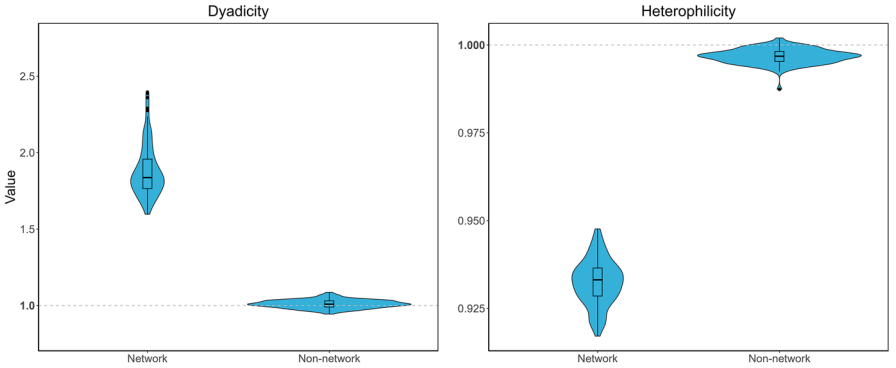

Fig. 6 The empirical distribution of the dyadicity and heterophilicity in the synthetically generated $\mathscr{D}^{Network}$ and $\mathscr{D}^{Non-network}$ data sets

Effect size of the features

Next, we estimate the coefficient vector $\boldsymbol{\beta}_f$ in each synthetic data set. We fit the following logistic regression model

$$\begin{aligned}\text{logit}(E[Y_{ijk}]) = {}& {}_0\beta_f + {}_1\beta_f \texttt{AgePH}_i + {}_2\beta_f \texttt{NrContractsPH}_{ij} \\ & + {}_3\beta_f \texttt{ClaimAmount}_{ijk} \\ & + {}_4\beta_f \texttt{ClaimAge}_{ijk} + {}_5\beta_f \texttt{n1.size}_{ijk} \\ & + {}_6\beta_f \texttt{n2.size}_{ijk} + {}_7\beta_f \texttt{n2.ratioFraud}_{ijk}.\end{aligned} \tag{14}$$

where i refers to the policyholder, j to the contract and k to the claim. This model is the same as the data-generating fraud model (see Table 4). Figure 7 depicts the empirical distribution of the estimated coefficient vector $\widehat{\boldsymbol{\beta}}$ across the 100 simulated data sets. Panel a shows the estimates obtained from $\mathscr{D}^{Network}$ and panel b from $\mathscr{D}^{Non-network}$. In both types of data sets, we observe some minor deviations from the specified effect size for most features, reflecting sampling variability. Further, the variability of the estimates is relatively small. Hence, we are able to accurately replicate the specified effect size for the different features, with only minor deviations due to sampling variability. The deviation from the specified effect size and variability, however, is substantially larger for the social network feature `n2.ratioFraud`. This feature represents the proportion of fraudulent claims in a claim's second order neighborhood. The estimated effect size of `n2.ratioFraud` is lower than its value as specified in $\boldsymbol{\beta}_f$. This is most likely attributable to the iterative growth in the number of fraudulent claims (see Algorithm 2), leading to a deviation in the estimated effect size from the originally specified value. In the first iterations, there are only a few instances of fraudulent claims. Hence, most of the unlabeled claims will have similar values for `n2.ratioFraud`. As the number of iterations increases, there will be a progressive increase in the proportion of fraudulent claims (see Appendix F). Accordingly, there will be more distinct feature values for `n2.ratioFraud`. Thus, the empirical distribution of `n2.ratioFraud` alters with each iteration.

4.2 Exploring the capabilities of the simulation engine: evaluating a fraud detection model's effectiveness

In this section, we illustrate the development and validation of a fraud detection model using a supervised learning technique (see Sect. 2). We proceed with the 100 synthetic data sets $\mathscr{D}^{Network}$. In each simulated data set we construct a fraud detection model by fitting a logistic regression model to the investigated claims (i.e., those that are investigated and labeled by the expert, see Sect. 3.4). We rely on logistic regression given its robustness to imbalanced class sizes [29, 36, 52]. Table 5 illustrates that said imbalance is present in each synthetic data set. To examine the added value of social network analytics when a network effect is present, we define two distinct model specifications. For model 1, we only include policyholder and claim-specific features

$$\begin{aligned}\text{logit}(E[Y_{ijk}]) = {}& {}_0\beta_f + {}_1\beta_f \texttt{AgePH}_i + {}_2\beta_f \texttt{NrContractsPH}_{ij} \\ & + {}_3\beta_f \texttt{ClaimAmount}_{ijk} + {}_4\beta_f \texttt{ClaimAge}_{ijk}.\end{aligned} \tag{15}$$

Next, we extend model 1 by incorporating social network features as well, resulting in model 2

$$\begin{aligned}\text{logit}(E[Y_{ijk}]) = {}& {}_0\beta_f + {}_1\beta_f \texttt{AgePH}_i + {}_2\beta_f \texttt{NrContractsPH}_{ij} \\ & + {}_3\beta_f \texttt{ClaimAmount}_{ijk} + {}_4\beta_f \texttt{ClaimAge}_{ijk} \\ & + {}_5\beta_f \texttt{n1.size}_{ijk} + {}_6\beta_f \texttt{n2.size}_{ijk} \\ & + {}_7\beta_f \texttt{n2.ratioFraud}_{ijk}.\end{aligned} \tag{16}$$

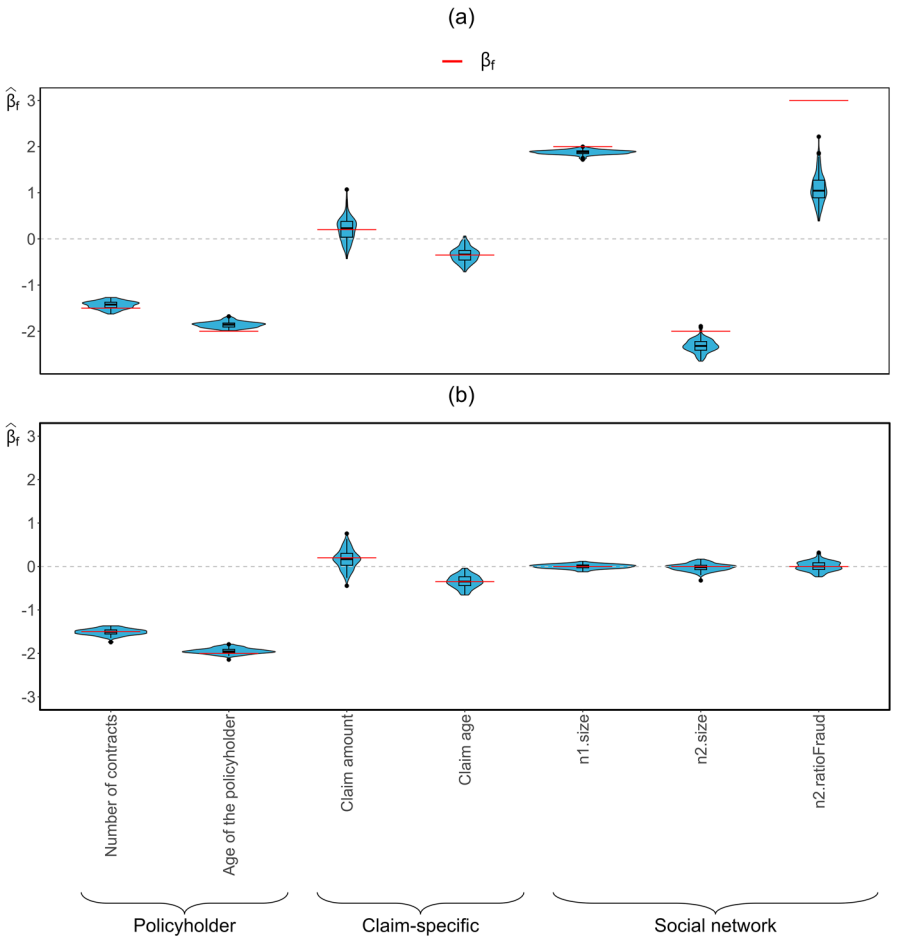

Fig. 7 Empirical distribution of the coefficient estimates across the (**a**) 100 simulated data sets $\mathscr{D}^{Network}$ and (**b**) 100 simulated data sets $\mathscr{D}^{Non-network}$. The red lines on the plot depict the features' effect size as specified in $\boldsymbol{\beta}_f$

To assess the predictive performance of the fraud detection models, we rely on the area under the receiver operating characteristic curve (AUC) [18] and the top decile lift (TDL) [27]. The AUC measures how well the model differentiates between fraudulent and non-fraudulent claims. An AUC of 0.5 corresponds to a random model and a perfect model has an AUC of 1. The TDL measures the extent to which a model surpasses a random model in detecting fraudulent claims. We calculate the TDL by dividing the proportion of fraudulent claims among the top 10% of claims with the highest predicted probability by the relative frequency of fraudulent claims in the data set. The TDL of a random model is equal 1. The higher the TDL, the better the model performance.

In each synthetic data set, we examine the in- and out-of-sample predictive performance of the fitted model. Hereto, we use the model fit to compute the probability of

fraud for claims in the in- and out-of-sample data set

$$\pi_{ijk} = \frac{e^{0\hat{\beta}_f + f\boldsymbol{x}_{ijk}^\top \widehat{\boldsymbol{\beta}}_f}}{1 + e^{0\hat{\beta}_f + f\boldsymbol{x}_{ijk}^\top \widehat{\boldsymbol{\beta}}_f}}. \tag{17}$$

The in-sample data set consists of the investigated claims (approximately 9% of all claims are investigated, see Table 5). Here, we use the labels of $Y_{ijk}^{\texttt{expert}}$ as outcome when computing the performance measures. The out-of-sample data set contains all uninvestigated claims. In our synthetic data set, we have the advantage of having access to the ground truth label Y_{ijk} of the uninvestigated claims, which is not available in real-life data sets. We calculate the out-of-sample AUC and TDL using Y_{ijk}. As such, we evaluate to which extent our model is able to generalize and detect fraud in the unlabeled claims.

Figure 8 depicts the in- and out-of-sample predictive performance of model 1 and model 2. In terms of AUC and TDL, model 2 consistently outperforms model 1 in both the in- and out-of-sample evaluations. Consequently, by incorporating social network features in addition to the traditional claim characteristics, we enhance the model's ability to identify fraudulent claims. Furthermore, the TDL of model 1 approaches one in all simulated data sets, indicating that the model performs no better than random chance in identifying fraudulent claims within the top 10% of predicted probabilities. In comparison, the TDL of model 2 is substantially larger than one. Model 2 also retains its predictive performance on the out-of-sample data sets. Hence, by training the model on the investigated claims, we can effectively capture the distinct patterns exhibited by fraudulent claims. One seemingly contradictory finding, however, is that the out-of-sample AUCs are higher than the in-sample AUCs. This discrepancy in performance is likely due to the different labels used for model evaluation. For the in-sample comparison, we rely on the expert judgment labels $Y_{ijk}^{\texttt{expert}}$. Conversely, for the out-of-sample comparison we use the ground-truth labels Y_{ijk}. This variation in label sources may contribute to the observed differences in model performance. In addition, the in-sample data sets are 10 times smaller than the the out-of-sample data sets. Consequently, the in-sample data sets exhibit more variability. Further, a small number of the investigated claims will be false positives or false negatives (see Sect. 3.4). When we fit the models with the ground truth-label Y_{ijk} instead of $Y_{ijk}^{\texttt{expert}}$, the in-sample performance is higher compared to the out-of-sample performance (see Appendix G).

5 Discussion

In this paper, we present a powerful and flexible toolbox to generate synthetic insurance fraud network data. The simulation engine consists of seven consecutive steps which enable us to generate a complete and complex data set. The engine generates policyholder characteristics, contract-specific features, the number of claims and individual claim costs, and the claim labels (fraudulent or non-fraudulent). To ensure that the simulated data accurately reflects the real-world scenario, the fraud investigation

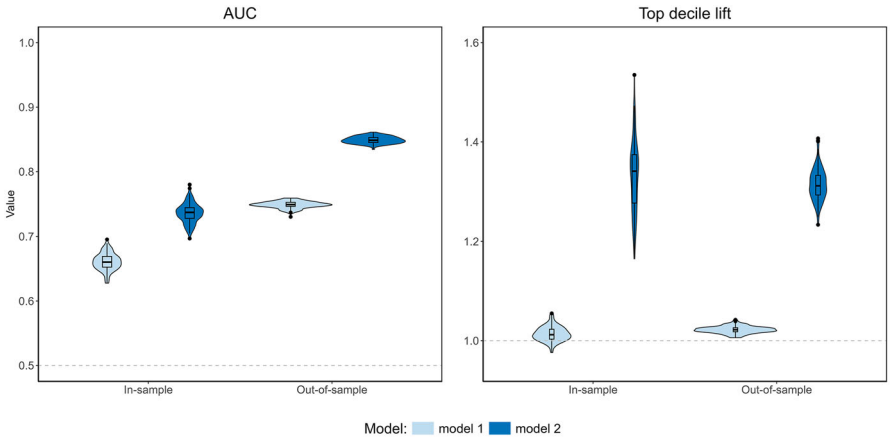

Fig. 8 Distribution of the performance measures across the 100 simulated data sets $\mathcal{D}^{Network}$. The grey line in the plots corresponds to the performance of a random model. The in-sample performance is evaluated using the labels of the investigated claims. The out-of-sample performance is assessed using the ground truth label Y_{ijk} of the uninvestigated claims

process is also replicated. The generated data from each step is combined to produce a final synthetic database that can be used for various purposes. In generating the synthetic data, the user has control over various data-generating mechanisms.

The simulation engine can produce diverse scenarios to meet different research needs. We showcase this ability by generating two distinct types of data sets, one where the social network effect is present during the claim label generation and one where it is absent. Our results highlight the toolbox's capability to simulate synthetic data according to the user-defined parameters. Our simulation engine accurately generates the desired class imbalance as well as the specified effect sizes of the covariates (including the social network features). As such, we are able to generate data sets that closely mirror real-life insurance fraud data sets in motor insurance.

The synthetic datasets mirror the network structure and include covariates similar to those present in the real-life motor insurance fraud dataset analyzed in [37]. Our simulation engine, however, does not generate more granular information about the filed claim which might be available to the insurer. For instance, in real-life datasets, insurers often have access to additional information such as a written report or textual description accompanying the claim. Additionally, our simulation engine is tailored specifically for simulating synthetic motor insurance data. Future research could explore methods to generate finer detailed information on the claims and expanding the capabilities of the engine to cover other (non-)life insurance products.

Researchers can utilize our simulation engine to conduct benchmark studies, aimed at addressing (methodological) challenges posed by insurance fraud. For instance, future research can focus on the evaluation of sampling techniques to handle the high class imbalance and the performance of learning methods in combination with said sampling techniques.

A Default configuration of the simulation engine

The default settings for the database, policyholder, contract-specific and claim characteristics are given in Table 6. The default data-generating claim frequency and claim severity models are given in Appendix D. In Appendix E, we specify the default data-generating fraud model.

We define the level of class imbalance as the ratio of the number of fraudulent claims to the total number of claims. Further, `ExcludeParties` is a parameter that allows to exclude certain types of parties from the network. We include all types of parties by default.

B Limiting the range of feature values

For `AgePH`, the user can define a range of acceptable values $[\min_{\texttt{AgePH}}, \max_{\texttt{AgePH}}]$ where $\min_{\texttt{AgePH}}$ denotes the lower bound and $\max_{\texttt{AgePH}}$ the upper bound. The default range is [18, 80]. Using the generator, we simulate $\texttt{AgePH}_i$ for $i = (1, \ldots, n_{ph})$ (see Tables 2 and 6). Hereafter, we calculate the relative frequency for all values $\lceil \texttt{AgePH}_i \rceil$ that fall within the predefined range $[\min_{\texttt{AgePH}}, \max_{\texttt{AgePH}}]$. The ceiling function $\lceil \cdot \rceil$ is used to round values to the nearest integer. Values outside of the range $[\min_{\texttt{AgePH}}, \max_{\texttt{AgePH}}]$ are redistributed among the integer values falling within this interval, proportional to the relative frequency of the values within this range. Hereafter, we take a random draw from $U(0, 1)$ and add this value to the integer to obtain a numeric value.

The default range $[\min_{\texttt{ExpPH}}, \max_{\texttt{ExpPH}}]$ for `ExpPH` is set to [0, 20]. Furthermore, $\texttt{AgePH}_i$ - $\texttt{ExpPH}_i$ cannot be smaller than the user-specified $\min_{\texttt{AgePH}}$. This would imply that the contract started before the policyholder reached the legal age of driving. Hereto, we define $\texttt{MaxExp}_i = \texttt{AgePH}_i$ - $\min_{\texttt{AgePH}}$. For values outside the prespecified range, we redraw a value from $U(\min_{\texttt{ExpPH}}, \texttt{MaxExp}_i)$.

We define [1, 5] as default range for `NrContractsPH`. Values outside this range are rounded to the closest boundary.

C Simulating type of coverage

The type of coverage is a nominal variable with three levels. We rely on a multinomial regression model to generate the type of coverage as a function of `ValueCar`, `AgeCar` and `AgePH`. The general form of the multinomial logistic regression model [1] is

$$\log\left(\frac{\pi_j(\boldsymbol{x}_i)}{\pi_J(\boldsymbol{x}_i)}\right) = \boldsymbol{x}_i^\top \boldsymbol{\beta}_j, \quad j = (1, \ldots, J-1),$$

where $\pi_j(\boldsymbol{x}_i) = P(\texttt{Coverage}_i = j|\boldsymbol{x}_i)$ denotes the probability that $\texttt{Coverage}_i$ equals category j. $\texttt{Coverage}_i$ denotes the response variable's value for observation i and here, we use $j = (1, \ldots, J)$ as an index for the categories of the response variable.

Table 6 Default configuration of the variables in the simulation engine

	Variable	Type	Description	Default value/generator
Data set	`TargetPrev`		Target level of class imbalance	`0.01`
	`NrPH`		Number of policyholder	`10,000`
	`NrExperts`		Number of unique experts	$\lfloor 0.01$`NrPH`$\rfloor$
	`NrBrokers`		Number of unique brokers	$\lfloor 0.01$`NrPH`$\rfloor$
	`NrGarages`		Number of unique garages	$\lfloor 0.03$`NrPH`$\rfloor$
	`NrPersons`		Number of unique persons	1.5`NrPH`
	`ExcludeParties`		Type of party to exclude	`Expert`
Policyholder	`AgePH`	Continuous	Age of the policyholder in years, default range is [18, 80]	$\mathcal{N}(40, 15)$
	`GenderPH`	Categorical	Gender of the policyholder, default settings are `female` if $u_i \leq 0.28$, `male` if $u_i > 0.29$ and `non-binary` otherwise	$u_i \sim U(0, 1)$
	`ExpPH`	Continuous	Exposure of the policyholder in years, default range is [0, 20]	$\mathcal{N}(5, 1.5)$
	`RateNrContracts`	Continuous	Rate parameter λ_i for generating `NrContractsPH`	$\lambda_i = 0.25(1.05 - 2.5 \times 10^{-6} \times \texttt{AgePH}_i + 0.0025 \times \texttt{AgePH}_i^2 - 2.65 \times 10^{-5} \times \texttt{AgePH}_i^3)$
	`NrContractsPH`	Ordinal	Number of contracts. Default range is [1, 5]	$\text{Poi}(\lambda_i)$

Table 6 continued

	Variable	Type	Description	Default value/generator
Contract-specific	ExpPHContracts	Continuous	Exposure corresponding to the contract	$\texttt{ExpPH}_i - \text{U}(0, \texttt{ExpPH}_i/2)$ if $\texttt{NrContractsPH}_i > 1$ else $\texttt{ExpPHContracts}_{ij} = \texttt{ExpPH}_i$
	AgeCar	Continuous	Age of the vehicle in years	$\max(\mathcal{N}(7.5, \sqrt{5}), \texttt{ExpPHContracts}_{ij})$
	OrigValueCar	Continuous	Original value of the vehicle	$\text{Exp}(\lambda_i / \text{NrContractsPH}_i)$
	ValueCar	Continuous	Current value of the car	$\texttt{OrigValueCar}_{ij}(1 - \delta^{\text{a}})^{\texttt{AgeCar}_{ij}}$
	Coverage	Categorical	Type of coverage provided by the insurance company (see Appendix C)	$\text{Multinomial}(1, \pi_{\text{TPL}}, \pi_{\text{PO}}, \pi_{\text{FO}})$
	Fuel	Categorical	Type of fuel of the vehicle	Bernoulli(0.3)
	BonusMalus	Ordinal	Level occupied in bonus-malus scale of the insurance company	$\min(\lfloor \mathcal{G}(1, 1/3) \rfloor, 22)$
Claim	ClaimAge	Integer	Number of months from the contract's inception to the date of the incident	$\lfloor \text{Exp}(0.25) \rfloor$
	ClaimDate	Continuous	Number of years between the start of the contract and the claim's filing date	$\max(U(0, \texttt{ExpPHContracts}_{ij}), \texttt{ClaimAge}_{ijk}/12)$
	Police	Categorical	Whether police was called when the incident happened	Bernoulli(0.25)
	nPersons	Integer	Number of people involved in the claim, range is [0, 5]	$S \xleftarrow{\pi_p} x, S = (0, 1, 2, 3, 4, 5)$ and $\pi_p = (0.02, 0.57, 0.19, 0.10, 0.10, 0.02)$

[a] If $\texttt{OrigValueCar} < 30000$, $\delta = 0.15$ and $\delta = 0.075$ otherwise. Hence, more expensive cars have a lower depreciation rate

We use category J as reference category. $\boldsymbol{x}_i$ denotes the covariate vector and $\boldsymbol{\beta}_j$ is the parameter vector for category j. For notational simplicity, we assume that $\boldsymbol{x}_i$ is fixed for all categories $j = (1, \ldots, J-1)$. Further, $\sum_{j=1}^{J} \pi_j(\boldsymbol{x}_i) = 1 \ \forall \ i \in (1, \ldots, N)$ where N denotes the total number of observations.

The category-specific probability for category $j = (1, \ldots, J-1)$ is calculated as

$$\pi_j(\boldsymbol{x}_i) = \frac{e^{\boldsymbol{x}_i^\top \boldsymbol{\beta}_j}}{1 + \sum_{h=1}^{J-1} e^{\boldsymbol{x}_i^\top \boldsymbol{\beta}_h}}$$

and we calculate this probability for reference category J as

$$\pi_J(\boldsymbol{x}_i) = \frac{1}{1 + \sum_{h=1}^{J-1} e^{\boldsymbol{x}_i^\top \boldsymbol{\beta}_h}}.$$

In our simulation engine, $\texttt{Coverage}_i \in (\texttt{TPL}, \texttt{PO}, \texttt{FO})$ and we define

$$\begin{aligned} \boldsymbol{\beta}_{\texttt{TPL}} &= (\log(0.50), \log(1.25), \log(0.25)), \\ \boldsymbol{\beta}_{\texttt{PO}} &= (\log(1.25), \log(0.75), \log(1.05)), \\ \boldsymbol{\beta}_{\texttt{FO}} &= (\log(1.50), \log(0.75), \log(1.25)). \end{aligned} \tag{18}$$

The covariate vector $\boldsymbol{x}_i$ consists of the normalized values for the value of the car, age of the car and age of the policyholder. Given a variable $a = (a_1, \ldots, a_i, \ldots, a_N)$, we normalize it to the range $[-1, 1]$ using

$$2\frac{(a - \min(a))}{\max(a) - \min(a)} - 1. \tag{19}$$

Hereafter, we calculate the probabilities $(\pi_{\text{TPL}}(\boldsymbol{x}_i), \pi_{\text{PO}}(\boldsymbol{x}_i), \pi_{\text{FO}}(\boldsymbol{x}_i))$ for all observations. We generate values for the type of coverage by taking random draws from Multinomial$(1, \pi_{\text{TPL}}, \pi_{\text{PO}}, \pi_{\text{FO}})$.

Using the default values (see Eq. 18), the probability of signing up for a full omnium is larger for expensive, relatively new cars and older policyholders. Similarly, the probability of taking out a partial omnium is higher for expensive, relatively new cars but here the effect of age is less strong. Young policyholders with an inexpensive, older car have a higher probability to take out a policy with only third party liability.

D Claim frequency and claim severity model

Both the claim frequency and claim severity model are based on the results in [20]. In this paper, the authors fit a claim frequency and claim severity model on a motor insurance portfolio from a Belgian insurer. Further, [20] used a data-driven method to bin the continuous variables `AgePH`, `AgeCar` and `BonusMalus` into categorical variables. These bins are given in Table 7. We denote these binned versions as

Table 7 Bins of the continuous variables as used in [20]

Variable	Bins
`AgePHBin`	[18, 26]; (26, 30]; (30, 36]; (36, 50]; (50, 60]; (60, 65]; (65, 70]; (70, 80]
`AgeCarBin`	[0, 5]; (5, 10]; (10, 20]; (20, max(`AgeCar`)]
`BonusMalusBin`	[0, 1); [1, 2); [2, 3); [3, 7); [7, 9); [9, 11); [11, 22]

`AgePHBin`, `AgeCarBin` and `BonusMalusBin`. By default, we use these binned versions in the data-generating claim frequency and claim severity model (see Table 8).

E Data generating fraud model and class imbalance

Table 9 depicts the default specification of the data-generating fraud model. The column names indicate which features are included, while the values represent the corresponding value in $\boldsymbol{\beta}_f$. Further, in the data-generating model we use the normalized version of the features `ClaimAmount`, `ClaimAge`, `n1.size`, `n2.size`, `AgePH` and `n2.ratioFraud` (see (19)). Hereby, we bring all features to the same scale. This ensures that the features' effect sizes, as specified in $\boldsymbol{\beta}_f$, are comparable. Furthermore, at the end of every iteration in Algorithm 2, we normalize both `n2.ratioFraud` and `n2.ratioNonFraud`. We do so since the network grows with every step in the algorithm. By normalizing these features, we aim to mitigate the influence of fluctuating values across iterations (also see Appendix F).

Further, the simulation engine allows us to specify the desired level of class imbalance p_t. We achieve this by employing the following approach in the third step of Algorithm 2, where we generate the claim labels

$$Y_{ijk} \sim \text{Bern}\left(\pi_{ijk}\right) \quad \text{and} \quad \pi_{ijk} = \frac{e^{{}_0\beta_f + {}_f\boldsymbol{x}_{ijk}^\top \boldsymbol{\beta}_f}}{1 + e^{{}_0\beta_f + {}_f\boldsymbol{x}_{ijk}^\top \boldsymbol{\beta}_f}}. \tag{20}$$

Before generating Y_{ijk}, we set a seed for the random number generator to ensure reproducibility. We achieve the desired level of imbalance p_t by optimizing

$$\min_{{}_0\beta_f} |p_t - {}_a p| \tag{21}$$

where the range of possible values for ${}_0\beta_f$ is set to [-10, 10] and where

$${}_a p = \frac{\sum_{i,j,k} I(Y_{ijk} = \texttt{fraudulent})}{\sum_{i,j,k} I(Y_{ijk} = \texttt{fraudulent OR } Y_{ijk} = \texttt{non-fraudulent})} \tag{22}$$

denotes the actual level of class imbalance in the synthetic data set (using all available claim labels). Here, $I(\cdot)$ represents the indicator function. However, for certain randomly drawn sets of Y_{ijk}, it is possible that there is no optimal ${}_0\beta_f$ (i.e., when ${}_0\beta_f$ is equal to -10 or 10) in a given iteration. In this scenario, we change the seed and

Table 8 Default specification of the claim frequency and claim severity model

Variable	β_{cf}	β_{cs}
(Intercept)	−2.18	6.06
`AgePH:`		
[18,26] (reference)		
(26,30]	log(0.85)	log(0.85)
(30,36]	log(0.75)	log(0.75)
(36,50]	log(0.70)	log(0.85)
(50,60]	log(0.60)	log(0.85)
(60,65]	log(0.55)	log(1.15)
(65,70]	log(0.60)	log(1.25)
(70, max(`AgePH`)]	log(0.70)	log(1.50)
`Coverage:`		
`TPL` (reference)		
`PO`	−0.12	−0.16
`FO`	−0.11	0.11
`AgeCarBin`		
(0, 5] (reference)		
(5,10]	log(0.90)	0
(10,20]	log(0.80)	0
(20, max(`AgeCar`)]	log(0.60)	0
`Fuel:`		
`Gasoline/LPG/Other` (reference)		
`Diesel`	log(1.19)	0
`BonusMalusBin:`		
[0,1) (reference)		
[1,2)	0.12	0.10
[2,3)	0.18	0.15
[3,7)	0.34	0.15
[7,9)	0.48	0.15
[9,11)	0.54	0.20
[11,22]	0.78	0.30

Table 9 Default specification of the data-generating fraud model

	`ClaimAmount`	`ClaimAge`	`n1.size`	`n2.size`	`NrContractsPH`	`AgePH`	`n2.ratioFraud`
$\boldsymbol{\beta}_f$	0.20	−0.35	2.00	−2.00	−1.50	−2.00	3.00

generate another set of Y_{ijk}. We return to step 1 of Algorithm 2 when no optimal ${}_0\beta_f$ is identified following 100 seed changes.

F Distribution values n2.ratioFraud

Figure 9 depicts the features values of `n2.ratioFraud` across the different iterations of Algorithm 2 when generating the claim labels in a synthetic data set. The horizontal axis depicts the iteration and the vertical axis the feature values. Per iteration, we show the empirical distribution of `n2.ratioFraud` in the random subset (see Algorithm 2). In the first iterations, we have a small number of fraudulent claims. As a consequence, most observations have similar feature values for `n2.ratioFraud`. The scarcity of distinct values is evident from the compactness of the violin plots. With each iteration, the number of fraudulent claims grows, resulting in an increase in distinct values for `n2.ratioFraud`. In Fig. 9, this is reflected by the increase in width of the violin plots.

G Predictive performance in the synthetic data sets

To obtain insights into the results in Sect. 4, we first fit the following model to the investigated claims

$$\begin{aligned}\text{logit}(E[Y_{ijk}]) = {}&{}_0\beta_f + {}_1\beta_f \texttt{AgePH}_i + {}_2\beta_f \texttt{NrContractsPH}_{ij} \\ &+ {}_3\beta_f \texttt{ClaimAmount}_{ijk}\end{aligned}$$

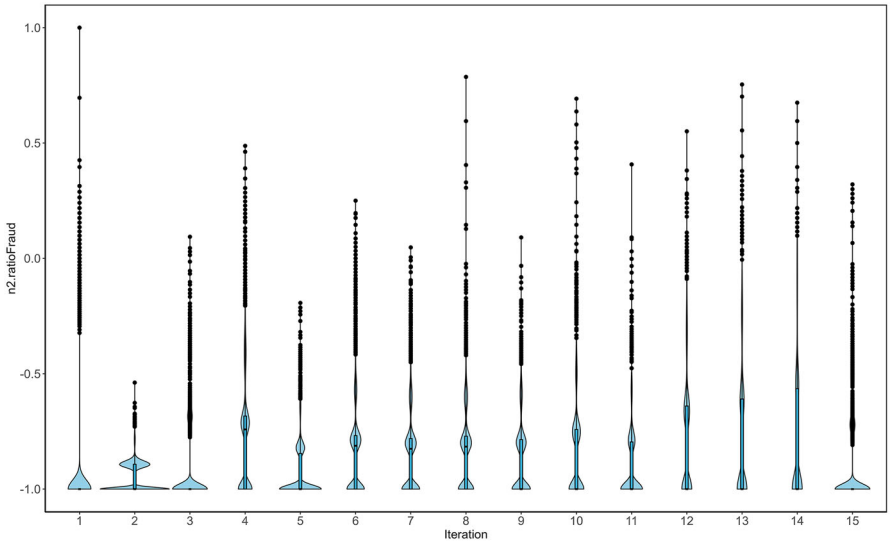

Fig. 9 Distribution of the values of `n2.ratioFraud` in each iteration. Per iteration, the violin plot depicts the density of `n2.ratioFraud` in the subset

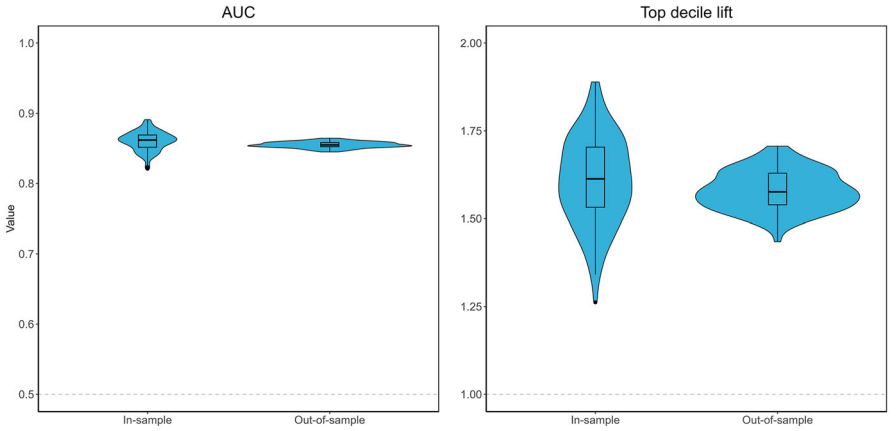

Fig. 10 Distribution of the performance measures in $\mathscr{D}^{Network}$. The grey line in the plots corresponds to the performance of a random model. The in-sample performance is evaluated using the ground truth label Y_{ijk} of the investigated claims. The out-of-sample performance is assessed using the ground truth label Y_{ijk} of the uninvestigated claims

$$+ {}_4\beta_f \texttt{ClaimAge}_{ijk} + {}_5\beta_f \texttt{n1.size}_{ijk} + {}_6\beta_f \texttt{n2.size}_{ijk} + {}_7\beta_f \texttt{n2.ratioFraud}_{ijk}. \quad (23)$$

Hence, here we use the ground-truth Y_{ijk} instead of the expert judgement $Y_{ijk}^{\texttt{expert}}$ as the response variable. Hereafter, we examine the performance on the investigated (i.e., the in-sample data set) and uninvestigated (i.e., the out-of-sample data set) claims. Figure 10 depicts the performance on the in- and out-of-sample data sets. Both the AUC and TDL indicate that the fitted models are substantially better than a random model. In addition, compared to the in-sample data sets, the performance is slightly lower on the out-of-sample data sets. Hence, the model is able to capture the underlying relationships between the predictors and the target variable.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s13385-024-00399-z.

Acknowledgements Katrien Antonio gratefully acknowledges funding from the FWO and Fonds de la Recherche Scientifique - FNRS (F.R.S.-FNRS) under the Excellence of Science (EOS) programme, project ASTeRISK Research Foundation Flanders [grant number 40007517], from the Chaire DIALog by CNP Assurances and the FWO network W001021N. The authors gratefully acknowledge support from the Ageas research chair on insurance analytics at KU Leuven. Additionally, the authors would like to thank Misja Langeler for the assistance provided during the initial stages of the simulation engine.

Data Availability The simulation engine used in this study is available as an open-source R package on GitHub: https://github.com/BavoDC/iFraudSimulator. Detailed configuration settings are described in Appendix A and the package documentation. The analysis can be replicated using the R script and simulated datasets provided in the supplementary material.

Declarations

Conflict of interest No potential Conflict of interest was reported by the authors.

References

1. Agresti A (2013) Categorical data analysis, 3rd edn. Wiley, Hoboken
2. Albashrawi M (2016) Detecting financial fraud using data mining techniques: a decade review from 2004 to 2015. J Data Sci 14(3):553–570
3. Andresen MA, Felson M (2009) The impact of co-offending. Br J Criminol 50(1):66–81
4. Avanzi B, Taylor G, Wang M, Wong B (2021) Synthetic: An individual insurance claim simulator with feature control. Insur Math Econ **100**(2):296–308
5. Baesens B (2023) Fraud analytics: a research agenda. J Chin Econ Bus Stud 21(1):137–141
6. Baesens B, Van Vlasselaer V, Verbeke W (2015) Fraud analytics using descriptive, predictive, and social network techniques: a guide to data science for fraud detection, 1st edn. Wiley, Hoboken
7. Barman S, Pal U, Sarfaraj MA, Biswas B, Mahata A, Mandal P (2016) A complete literature review on financial fraud detection applying data mining techniques. Int J Trust Manag Comput Commun 3(4):336–359
8. Denuit M, Dhaene J, Goovaerts M, Kaas R (2005) Actuarial theory for dependent risks: measures, orders and models. Wiley, West Sussex
9. European Insurance and Occupational Pensions Authority (2019) Big data analytics in motor and health insurance: a thematic review. Publications Office of the European Union, Luxembourg
10. FBI 2022 (2022) Investigating insurance fraud. https://www.fbi.gov/stats-services/publications/insurance-fraud. Accessed 08 July 2022
11. Frees EW, Gao J, Rosenberg MA (2011) Predicting the frequency and amount of health care expenditures. N Am Actuarial J 15(3):377–392
12. Frees E, Derrig R, Meyers G (2014) Predictive modeling applications in actuarial science, vol 1. Predictive modeling techniques. Cambridge University Press, New York
13. Gabrielli A, Wüthrich M (2018) An individual claims history simulation machine. Risks **6**(2)
14. Garrido J, Genest C, Schulz J (2016) Generalized linear models for dependent frequency and severity of insurance claims. Insur Math Econ 70:205–215
15. Ghobadi F, Rohani M (2016) Cost sensitive modeling of credit card fraud using neural network strategy. In 2016 2nd international conference of signal processing and intelligent systems (ICSPIS). IEEE, New York, pp 1–5
16. Goldburd M, Khare A, Tevet D, Guller D (2016) Generalized linear models for insurance rating. Casualty Actuarial Society, CAS Monographs Series, Arlington, p 5
17. Gomes C, Jin Z, Yang H (2021) Insurance fraud detection with unsupervised deep learning. J Risk Insur 88(3):591–624
18. Hanley J, McNeil B (1982) The meaning and use of the area under a receiver operating characteristic (roc) curve. Radiology 143(1):29–36
19. He X, Gao M, Kan M-Y, Wang D (2017) Birank: towards ranking on bipartite graphs. IEEE Trans Knowl Data Eng 29(1):57–71
20. Henckaerts R, Antonio K, Clijsters M, Verbelen R (2018) A data driven binning strategy for the construction of insurance tariff classes. Scand Actuar J 2018(8):681–705
21. Hilal W, Gadsden SA, Yawney J (2022) Financial fraud: A review of anomaly detection techniques and recent advances. Expert Syst Appl 193:116429
22. Jensen D (1997) Prospective assessment of AI technologies for fraud detection: a case study. In: AAAI workshop on AI approaches to fraud detection and risk management. Citeseer, pp 34–38
23. KBC Brussels (2023) Accident with no dispute, or claim on comprehensive insurance. https://www.kbcbrussels.be/retail/en/insurance/vehicle/damage-or-theft/accident-with-no-dispute-or-claim-on-comprehensive-insurance.html. Accessed 27 Jan 2023
24. Kho JRD, Vea LA (2017) Credit card fraud detection based on transaction behavior. In: TENCON 2017-2017 IEEE region 10 conference. IEEE, New York, pp 1880–1884
25. Khondoker M, Dobson R, Skirrow C, Simmons A, Stahl D (2016) A comparison of machine learning methods for classification using simulation with multiple real data examples from mental health studies. Stat Methods Med Res 25(5):1804–1823 (**PMID: 24047600**)
26. Kumar P (2010) Probability distributions and estimation of Ali–Mikhail–Haq copula. Appl Math Sci 4(14):657–666
27. Lemmens A, Croux C (2006) Bagging and boosting classification trees to predict churn. J Market Res 43(2):276–286

28. Lopez-Rojas EA, Gorton D, Axelsson S (2015) Using the RetSim simulator for fraud detection research. Int J Simul Process Model 10(2):144–155
29. Marques AI, Garcia V, Sanchez JS (2013) On the suitability of resampling techniques for the class imbalance problem in credit scoring. J Oper Res Soc 64(7):1060–1070
30. McCullagh P, Nelder JA (1999) Generalized linear models. Chapman and Hall, London
31. Morris TP, White IR, Crowther MJ (2019) Using simulation studies to evaluate statistical methods. Stat Med 38(11):2074–2102
32. Newman M (2010) Networks: an introduction. Oxford University Press, Oxford
33. Ngai E, Hu Y, Wong Y, Chen Y, Sun X (2011) The application of data mining techniques in financial fraud detection: a classification framework and an academic review of literature. Decis Supp Syst 50(3):559–569
34. Nur Prasasti IM, Dhini A, Laoh E (2020) Automobile insurance fraud detection using supervised classifiers. In: 2020 international workshop on big data and information security (IWBIS). IEEE, New York, pp 47–52
35. Ohlsson E, Johansson B (2010) Non-life insurance pricing with generalized linear models. Springer, Heidelberg
36. Oommen T, Baise LG, Vogel RM (2011) Sampling bias and class imbalance in maximum-likelihood logistic regression. Math Geosci 43(1):99–120
37. Óskarsdóttir M, Ahmed W, Antonio K, Baesens B, Dendievel R, Donas T, Reynkens T (2022) Social network analytics for supervised fraud detection in insurance. Risk Anal 42(8):1872–1890
38. Page L, Brin S, Motwani R, Winograd T (1999) The PageRank citation ranking: bringing order to the web. Technical report, Stanford InfoLab, Stanford
39. Park J, Barabási A-L (2007) Distribution of node characteristics in complex networks. Proc Natl Acad Sci PNAS 104(46):17916–17920
40. Pourhabibi T, Ong K-L, Kam BH, Boo YL (2020) Fraud detection: a systematic literature review of graph-based anomaly detection approaches. Decis Supp Syst 133:113303
41. Quijano Xacur OA, Garrido J (2015) Generalised linear models for aggregate claims: to Tweedie or not? Eur Actuarial J 5(1):181–202
42. Reiss AJ (1988) Co-offending and criminal careers. Crime Justice (Chicago, Ill.) **10**:117–170
43. Roy R, George KT (2017) Detecting insurance claims fraud using machine learning techniques. In: 2017 international conference on circuit, power and computing technologies (ICCPCT). IEEE, New York, pp 1–6
44. So B, Boucher J-P, Valdez EA (2021) Synthetic dataset generation of driver telematics. Risks **9**(4)
45. Srivastava A, Yadav M, Basu S, Salunkhe S, Shabad M (2016) Credit card fraud detection at merchant side using neural networks. In: 2016 3rd international conference on computing for sustainable global development (INDIACom). Bharati Vidyapeeth, New Delhi as the organizer of INDIACom—2016, pp 667–670
46. Storchmann K (2004) On the depreciation of automobiles: an international comparison. Transportation (Dordrecht) 31(4):371–408
47. Subudhi S, Panigrahi S (2020) Use of optimized fuzzy c-means clustering and supervised classifiers for automobile insurance fraud detection. J King Saud Univ Comput Inf Sci **32**(5):568–575
48. Sundarkumar GG, Ravi V (2015) A novel hybrid undersampling method for mining unbalanced datasets in banking and insurance. Engineering Applications of Artificial Intelligence 37:368–377
49. Thabtah F, Hammoud S, Kamalov F, Gonsalves A (2020) Data imbalance in classification: experimental evaluation. Inf Sci 513:429–441
50. Tukey JW (1977) Exploratory data analysis. Addison-Wesley, Reading
51. Tumminello M, Consiglio A, Vassallo P, Cesari R, Farabullini F (2023) Insurance fraud detection: a statistically validated network approach. J Risk Insur 90(2):381–419
52. van den Goorbergh R, van Smeden M, Timmerman D. Van, Calster B (2022) The harm of class imbalance corrections for risk prediction models: illustration and simulation using logistic regression. J. Am Med Inform Assoc 29(9):1525–1534
53. van Koppen MV, de Poot CJ, Kleemans ER, Nieuwbeerta P (2010) Criminal trajectories in organized crime. Br J Criminol 50(1):102–123
54. Van Vlasselaer V, Eliassi-Rad T, Akoglu L, Snoeck M, Baesens B (2016) GOTCHA! network-based fraud detection for social security fraud. Manag Sci 63(9):3090–3110
55. Vosseler A (2022) Unsupervised insurance fraud prediction based on anomaly detector ensembles. Risks (Basel) **10**(7):132

56. Warren DE, Schweitzer ME (2018) When lying does not pay: how experts detect insurance fraud. J Bus Ethics 150(3):711–726
57. West J, Bhattacharya M (2016) Intelligent financial fraud detection: a comprehensive review. Comput Secur 57:47–66

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.