Computer Science and Information Systems 23(1):657–685 https://doi.org/10.2298/CSIS250405084D

A Knowledge Graph based Approach for Credit Card Fraud Detection*

George Konstantinos Dimou and Georgia Koloniari

University of Macedonia, Thessaloniki, Greece
{dai18069,gkoloniari}@uom.edu.gr

Abstract. Credit Card Fraud Detection remains an evolving and difficult challenge due to its complexity, class imbalance, and scale of transactional data. In this work, we propose a novel graph-based approach that constructs a Knowledge Graph (KG) from transactional data to model relationships between different entities. We calculate a variety of centrality measures, both in an unweighted and in a weighted KG, where edge weights represent transaction-specific features like amount, in order to capture structural importance. These centrality measures are then used to enrich the feature space for multiple Machine Learning (ML) models. Our experimental evaluation assesses the performance of the proposed approach with respect to both accuracy and efficiency. Various experiments are conducted, comparing multiple centrality measures, feature combinations and resampling, showcasing how the addition of the centrality measures as classification features significantly improves the performance of our classification models compared to relying only on the original transactional attributes.

Keywords: Knowledge Graphs, Centrality Measures, Machine Learning, Weighted Graph, Credit Card Fraud Detection.

1. Introduction

Fraud, defined as deliberate deception with the intention of obtaining personal or financial gain, poses a diachronic problem which nowadays emerges as more prevalent than ever. Focusing specifically on fraud for financial gain, its effects amount to substantial losses both to individuals and businesses. With the widespread of card-based payments *Credit Card Fraud* becomes a particularly prominent type of fraud, and comprises the most common type of identity theft, while the attempts at fraudulent credit card transactions rise by 46% every year with a projected $43 billion of losses by 2026 [34]. Credit card fraud can be distinguished between card-present and card-not-present fraud. The latter type accounts for 65% of all credit card losses [34] due to in part the ever-growing number of online transactions. While technology offers avenues for improved preventive measures (such as secure embedded microprocessor chips) these rely mainly on mistakes on the fraudsters part, and most often new means to circumvent these precautions keep emerging.

From the above, it becomes clear, that preventive measures are not sufficient, and of particular importance becomes the challenge of timely detection of a fraud attempt so as to minimize any potential losses. Therefore, *Fraud Detection*, defined as identifying

* This paper is an extended version of the work presented in [13].

activities suspicious of being fraudulent, is one of the fundamentals of fraud risk management along with prevention and investigation. *Credit Card Fraud Detection* (CCFD) is a subcategory of fraud detection that deals with fraudulent activities involving the use of credit cards, i.e., targeting credit card fraud, particularly.

Towards the development of effective fraud detection approaches both data analysis relying on statistics and AI-based methods, and in particular, Machine Learning (ML) methods have been widely deployed [2,21,31,39,41]. All such methods rely on feature engineering, which is not a trivial task, requiring to select appropriate features that will enable accurate and effective detection of suspicious transactions. However, though the advances in AI and ML offer significant promise for further improvements, one dimension that must not be ignored is the correlations that fraudulent transactions often have. Traditional approaches mostly ignore such hidden relationships, considering each transaction in isolation, and thus disregarding important information that can be garnered by identifying and exploiting the interconnections between fraudsters and fraudulent transactions.

To address this issue, in our work, we propose utilizing *Knowledge Graphs* (KGs) for modeling transactional data and exploiting KG-based features for CCFD. Knowledge Graphs are used to represent entities and their relationships and are particularly appropriate when we are interested in studying the inter-workings of any complex system such as a transaction network. Instead of only studying each transaction separately, modeling the data through a KG, enables us to embed it in a network that facilitates the revelation of hidden patterns and relationships that otherwise would be very difficult, if not impossible, to detect. The goal of this work is to design an appropriate KG for modeling transactional data that can be utilized to extract enriched features that, in turn, can be leveraged in ML approaches to achieve accurate fraud detection.

In particular, we explore two alternative KG models, one unweighted and one weighted, that exploits transactional properties to define appropriate edge weights. Network analysis is then utilized to extract additional network features that can act in a way complimentary to the traditional tabular data that conventional ML methods usually employ. As network features, we consider network *centrality measures*, that is measures that are used to quantify the importance of different nodes in a network. Centrality measures consider different aspects of network structure and thus identify nodes with different roles of importance in the network, acting as central figures, appearing in many network paths and so on. To address efficiency, we implement our KG models in a graph database and make use of its network analysis libraries to extract network features. We model CCFD as a binary classification problem and experimentally study the performance of various ML classifiers and various combinations of centralities with respect to both detection accuracy and efficiency.

The work presented here extends previous work that used KGs for CCFD presented in [13]. In this work, we consider a second weighted KG in addition to an unweighted KG model, and deploy additional centrality measures as well as weighted versions of them that exploit the corresponding weighted KG model. Furthermore, we have conducted a very thorough experimental evaluation comparing different combinations of centrality-based features and traditional features to determine the most successful combinations along with more classifiers. We have also conducted new experiments showcasing the scalability and time efficiency of the proposed approach.

Our approach is presented in the next sections as follows. Section 2 briefly summarizes work related to both CCFD and FD and KG-based approaches. Section 3 introduces our proposed KG data modeling, presenting both the unweighted and weighted KG model. Section 4 defines the CCFD classification problem, explains our ML pipeline, and focuses on the different centrality measures that we study. Section 5 includes our experimental evaluation, and Section 6 offers conclusions and directions for future work.

2. Related Work

Advancements in Artificial Intelligence (AI) and ML, have made them irreplaceable in CCFD. A plethora of studies [10,19,22,24,35] showcase how even the simplest ML models can perform noticeably better than conventional rule-based systems. Most such approaches utilize the tabular transactional data as their learning features, while most recently there are some attempts that explore the use of KGs to improve detection performance.

2.1. Fraud Detection Using Tabular Data Features

A characteristic that limits the performance of AI and ML-based methods is the high class imbalance that FD in general and CCFD as well exhibit. Most transactions are legitimate transactions and only a very small percentage consists of fraudulent transactions, making their classification as such extremely difficult. Therefore, many works particularly target this class imbalance issue by deploying different sampling schemes such as SMOTE [8], towards improving FD and CCFD accuracy. Varmedja et al. [40] employed a pipeline utilizing a feature selection tool, SMOTE and normalization to address the issue. Similarly, Dhankhad [12] used multiple algorithms and ensemble learning in tandem with SMOTE, observing the clear superiority of the stacked models. In another study, Rahman et al. [33] examined the Chinese stock market and financial research databases and utilized boosting algorithms that incorporated random under-sampling and clustering based under-sampling, called RUSBoost and CUSBoost, respectively. Those models clearly outperformed other popular boosting algorithms such as AdaBoost and XGBoost in terms of accuracy, showcasing the need of those specially-engineered algorithms. Even though, resampling can be beneficial and effectively reduce class imbalance, it frequently results in significant computing overhead and possible model bias. Furthermore, there are also instances where using resampling techniques even worsens performance, as also seen later in our work, offering no real value. Thus, alternative approaches are also explored, such as in [15], where instead of sampling, they opted to use multiple classifiers for different user groups and proposed clustering but along with substantial preprocessing increasing overhead as well.

Another direction towards dealing with class imbalance is methods based on fuzzy theory. In their stacking architecture, Hussein et al. [36] combined fuzzy set theory, rough set-based classification and SVM with fuzzy Rough Nearest Neighbor and Sequential Minimal Optimization. Although this approach showcased potential in handling imbalanced datasets, it suffered from significant computational overhead, rendering it impractical for larger datasets, as their evaluation was limited to fewer than 1000 samples. Similarly, Charizanos et al. [7] presented an online fuzzy FD system that used reasoning to accurately identify questionable credit card transactions. However, their method lacked generalization ability due to its focus on a single type of fraud - transactional fraud - and the need of fuzzy number types fine-tuning required for every different fraud type. Due to their computational requirements and limited applicability, fuzzy methods were judged inappropriate for our study's goals.

2.2. Knowledge Graphs for Fraud Detection

Although the idea of KGs dates back to the 1980s [38], it was not until DBpedia was launched in 2007 and Google's use in 2012 that they gained the popularity they have today. Despite their potential, KGs are rarely used in FD. A number of works represent the transactional data with graphs where nodes represent transactions, and focusing on unsupervised learning, model the problem of fraud detection as anomaly detection, aiming to identify anomalous nodes, aka fraudulent transactions [1,14,20]. Another direction of research more closely related to our approach focuses on the construction of domain-specific Knowledge Graphs and the use of graph embeddings or network measures and supervised ML for fraud detection. Zhang et al. [44] used an Auto Insurance Knowledge Graph to model transaction correlations. Their method included creating an ontology using ML-based feature selection and having domain experts confirm it. Using KG embeddings, they managed to enhance gang FD, a type of fraud that involves links between claimants and repair shops by integrating geographic data with string similarity. They evaluated using XGBoost, SVM and Neural Networks, showing notable performance gains in detecting individual fraud cases. Employing four centrality measures - Degree, Betweenness, Closeness and Eigenvector each providing insights into corporate networks - Wen et al. [42] built a KG centered on managers and their associations with firms. They evaluated their KG using SVN, KNN, Decision Tree (DT) and Random Forest (RF) trained on data from Chinese A-share market from 2011 to 2017, with PCA and SMOTE for preprocessing. However, they did not explore combinations of centralities potentially missing valuable contextual information.

More recently, Graph Neural Networks (GNNs) [37] have also been deployed to address fraud detection along with the use of graph-based data representations. Cherif et al. [9] presented an encoder-decoder GNN model, that effectively captures complex interactions between customer and merchants by framing FD as an edge prediction task. Their method achieves strong performance in a large dataset by integrating geographical data and engineering transaction features into a heterogeneous network structure. The model's reliance on distance-based features may however restrict its usage in environments with restricted access in real-time geolocation. Also, in [26], fraud detection in blockchain is addressed using Variational Graph Autoencoders based on a KG that is build based on the blockchain transactions. These methods also struggle with class imbalance and efforts to address the issue in this context as well have appeared, such as GraphSMOTE [45]. GraphSMOTE applies over-sampling in an embedding space it creates, while it trains both the GNN and the over-sampling module to guarantee the integration of the generated data into the learning process.

3. Knowledge Graph Based Modeling

Traditional state-of-the-art ML approaches primarily focus on individual entities when trying to detect fraudulent transactions. Typically, they examine transactions independently, ignoring any possible connections between financial transactions. For instance, the model would assess a new transaction executed by a user with a rich fraudulent record similarly to someone who solely engages in lawful activities. Recognizing that, KGs can be used to efficiently tackle this limitation, due to their natural treatment of transactions as interconnected entities.

By organizing transactions as linked entities rather than discrete events, KGs offer a practical approach, with a more comprehensive modeling of financial activities and the different relationships between transactions, the actors performing the transactions and other related properties. In this section, we introduce two distinct KG-based architectures: an *unweighted* model that captures relationships without any additional numerical characteristics and a *weighted* model that incorporates financial disparities as edge weights to improve FD.

3.1. Unweighted KG Model

They key for the creation of a KG comes down to effectively describing data that has been extracted from a data source, such as a relational database. It is crucial to select the most appropriate entities and the relationships that connect them. These characteristics would be representative of the various aspects present in a transaction, such as its temporal dimensions, transactional data and finally related actor-user information necessary to construct our KG. Since KGs by definition mandate the existence of multiple entities [16,43], a method to produce them is vital.

Following the *property graph model*, we represent the transactional data by discerning between nodes with different labels representing semantically related entities, directed, named edges that represent the different types of relationships existing between the defined entities, and finally properties that are used to describe both nodes (entities) and edges (relationships).

This is accomplished by representing each *Transaction* as a separate entity and establishing additional entities for *Users* and *Time*. The sender (Originator) and receiver (Destination) of a Transaction, are mapped using User nodes, which ensure the appropriate representation of all financial actors. A Time entity is employed to incorporate the temporal dimension, preserving the Transactions chronological order. Finally, *amount* and *type* are two distinct examples of Transaction-specific attributes, which are maintained as properties directly connected to the Transaction entity.

Maintaining the integrity of the graph structure requires the establishment of relationships that are understandable and straightforward. For instance, a *MADE_A* relationship can be used to link the Originator User to the Transaction entity and similarly a *WITH_* relationship can link the Transaction to the Destination User. Furthermore, a temporal *AT* relationship links the Transaction to the corresponding Time entity.

Careful consideration must be given to attributes which are closely tied to account balances, for both User nodes involved, before and after the transaction transpires. Based on the realistic assumption that a User can be directly involved into multiple transactions happening simultaneously, attributing such features to User nodes could lead to redundancy. A more efficient alternative handles those features as relationship properties between Users and Transactions. By preserving similar relationship properties, not only unnecessary duplication is avoided, but also the usage of weighted centrality measurements is enabled.

Figure 1 presents a potential structure, with minimal number of features as described previously, that demonstrates how transactions and their associated entities are cohesively represented within the knowledge graph.

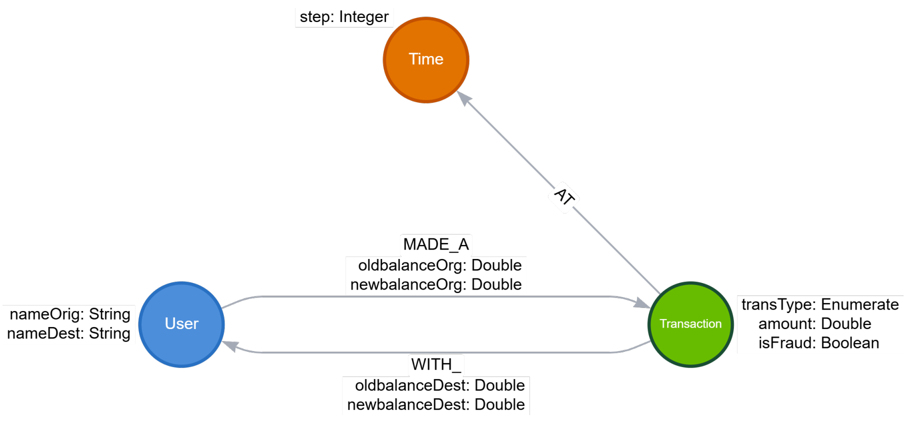

Fig. 1. Visualization of a potential unweighted Knowledge Graph structure following the property graph model. We discern between nodes labeled as: User, Transaction and Time along with their associated relationships

The naming conventions are consistent with our particular dataset, however depending on the application context they can be adapted, replaced or in some occasions omitted. By using the property graph model, our KG can be easily extended to incorporate additional properties, and even relationships or entities if these are available.

As stated, an unweighted graph is a type of graph which contains no numerical values assigned to the relationships between entities. In the context of FD, this could represent relationships between Users, Transactions and Time solely based on their presence alone. Due to the fact, that it eliminates the possibility of biases introduced from numerical properties, this kind of representation proves essential for applying structural analysis techniques such as centrality measures.

3.2. Weighted KG Model

In contrast to unweighted graphs, a weighted graph assigns numerical values, weights, to edges, so as to differentiate between relationships of varying strength or intensity. As such, a weighted graph may offer a more accurate depiction of transactional interactions by incorporating numerical values into the relationships between the entities that quantify the strength of the corresponding relationships. In other words, providing edges with meaningful values, relationship weights provide another layer of information by capturing patterns beyond simple connectivity.

To this end, we propose the use of a weighted KG model in which we introduce three different edge weights, specific to the edge type of the specific edge they characterize. In particular, we define the *sender weight* w_{sender} representing the change in balance for the originator of the transaction, the *receiver weight* $w_{receiver}$ representing the change in balance for the recipient of the transaction, and the time weight w_{time} which captures the net transfer, as formally defined in Eq. (1), Eq. (2) and Eq. (3), respectively.

$$w_{sender(i)} = |newbalanceOrg_i - oldbalanceOrg_i| \tag{1}$$

$$w_{receiver(j)} = |newbalanceDest_j - oldbalanceDest_j| \tag{2}$$

$$w_{time(i,j)} = |w_{receiver(j)} - w_{sender(i)}| \tag{3}$$

where:

$newbalanceOrg_i$	= final balance of the Originator after the transaction.
$oldbalanceOrg_i$	= initial balance of the Originator before the transaction.
$newbalanceDest_j$	= final balance of the Destination after the transaction.
$oldbalanceDest_j$	= initial balance of the Destination before the transaction.

While in most cases one would expect the receiver and sender weights to be equal, especially in cases of fraudulent transactions this is not always the case, which is what motivated us to define the weights based on these differences instead of the transaction amount.

In Fig. 2, we present an instance of a weighted KG. Bold edges indicate a greater difference in balances between users following a transaction, while edges with normal width represent smaller balance variations. All other properties are omitted from the figure to avoid redundancy, retaining only the weights with toy values for clarity.

4. Our Approach

The application of conventional ML techniques for the problem of CCFD has long been a topic of interest for researchers, resulting in a plethora of works being published annually. These initiatives employ a variety of techniques, ranging from unsupervised learning intended for real-time FD scenarios to supervised approaches utilizing publicly available datasets. In this paper, we focus on supervised learning and propose leveraging features derived from an appropriately constructed KG to improve its performance.

4.1. Problem Definition

Adopting supervised learning to address the problem of fraud detection, we model CCFD as a problem of binary classification where the goal is to predict whether a new transaction is legitimate or fraudulent. Any ML classification method relies on known labeled historical data, called the training set, in order to learn a function (or model) that will be able to accurately predict the label of new, unknown data. A significant step in this process is feature engineering and/or selection, that is the process of constructing and/or selecting the

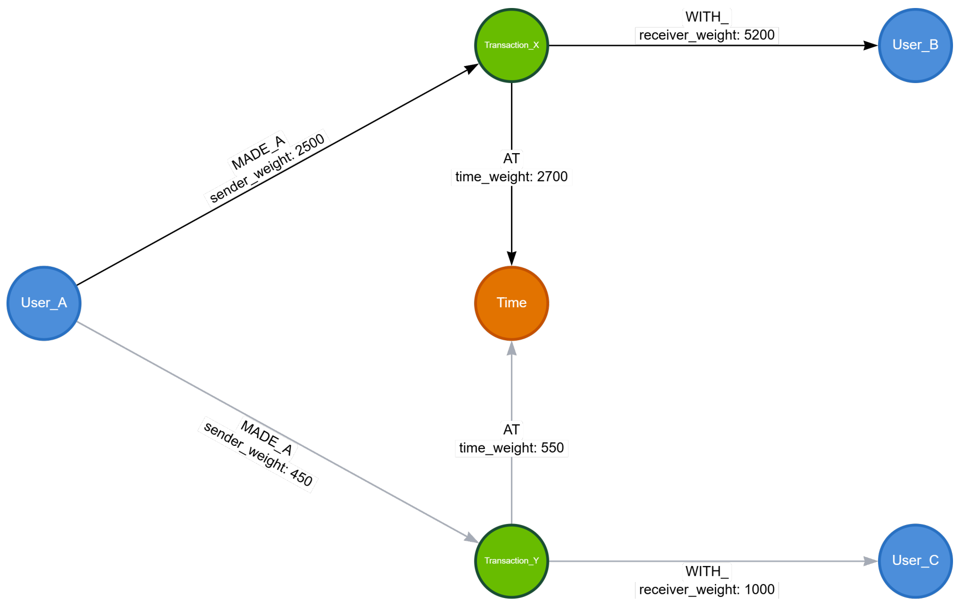

Fig. 2. Weighted KG representation. User_A has engaged in two transactions, with Transaction_X involving a larger balance difference, indicated by a bold relationship edge. Previously mentioned node and relationship features are removed to avoid redundancy

appropriate characteristics (features) of the data that will achieve the construction of an effective model with high predictive accuracy. In CCFD, this task is usually relegated to just utilizing all available transactional data attributes as input into various classification algorithms.

Problem 1 (Credit Card Fraud Detection as Binary Classification). Given a set of historical transaction records $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$, where each transaction t_i is represented by a feature vector $\mathbf{x}_i \in \mathbb{R}^d$ and an associated binary label $y_i \in \{0, 1\}$ indicating whether the transaction is legitimate (0) or fraudulent (1), the objective is to learn a classification function $f : \mathbb{R}^d \to \{0, 1\}$ that can accurately predict the label $\hat{y}$ of an unseen transaction $\mathbf{x}_{\text{new}}$.

The main objective of this work is to further enhance performance by incorporating new transactional features that frequently go undetected in the traditional aforementioned models. To that end, we propose leveraging the KG that can be derived from the same transactional data and use its structural properties as supplementary features for the ML classifiers. Let $\mathcal{G} = (V, E)$ represent our KG modeling entities such as users, transactions and temporal information as we have described in the previous section. Using network analysis, we derive additional structural features from $\mathcal{G}$ and augment the original feature vectors of our transactional data before using them as input for the ML classifiers. In particular, the feature vector $\mathbf{x}_i$ of each transaction t_i is extended as follows:

$$\mathbf{x}'_i = [\mathbf{x}_i \,\|\, \mathbf{c}_i],$$

where $\mathbf{c}_i$ denotes the vector of network features associated with t_i. As network features, we rely on the use of different node *centrality* measures, which we describe in detail in the next section.

Thus, given the set of transaction records $\mathcal{T} = \{t_1, t_2, \ldots, t_n\}$ and the derived KG, $\mathcal{G} = (V, E)$, constructed from $\mathcal{T}$, our goal is to learn a classification function $f' : \mathbb{R}^{d'} \to \{0, 1\}$ based on the enhanced feature vectors $\mathbf{x}'_i$ that will predict whether a new transaction is fraudulent (1) or not (0).

The entire data analysis pipeline is shown in Fig. 3. The raw data, after any preprocessing, such as cleaning and sampling, is additionally transformed into two KG representations (weighted and unweighted). Features are extracted both from the clean transaction data and the KGs using graph analysis that yields centrality-based features. Both types of features are combined and used as input to train the ML classifiers.

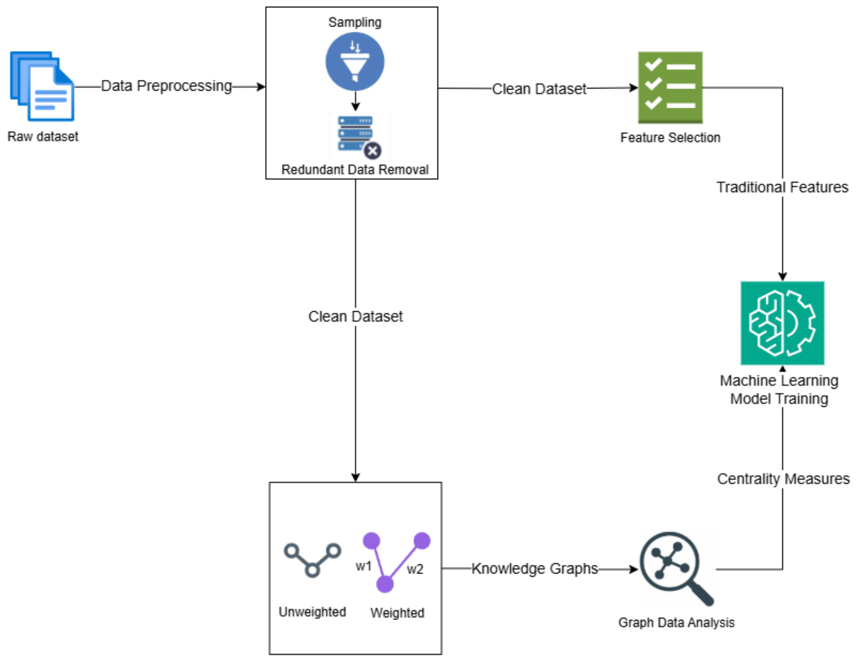

Fig. 3. Overview of the ML pipeline using graph-based features. The workflow begins with the preprocessing of the raw data. Two parallel paths are then followed: one for traditional feature extraction via feature selection and another for Knowledge Graphs construction, for both the unweighted and weighted variants. The graphs are then analyzed to extract centrality measures, which as a final step will be utilized alongside the original features to enhance machine learning model training

4.2. Graph-Based Features

Incorporating a KG into an ML pipeline comes down to effectively providing extra structural information that improves the learning process. This is usually achieved either through the usage of KG *embeddings* or *centrality measures*. The focus of this study is on the latter. Centrality measures serve as quantitative indicators of a node's significance or impact within the graph, capturing topological properties which are normally inaccessible through conventional tabular data representations. Different centrality measures assess the importance of a node in the network's structure through different perspectives, and therefore the use of several such measures can provide additional information about the role a node plays in the overall network.

Thus, in our approach, after constructing the KG based on the cleaned transactional data, we evaluate several centrality measures using both the unweighted and weighted KG model, as most of these measures are defined for both unweighted and weighted graphs. This approach enables us to experimentally study how the presence of relationship weights affects the importance of each node when incorporated into the centrality measure by the weighted model.

Let $G = (V, E)$ be an unweighted graph, where V denotes the set of vertices and $E \subseteq V \times V$ represents the set of edges connecting the pair of nodes. All edges are regarded as equal in this formulation and no numerical values are assigned to any. A weighted graph extends upon this definition to $G = (V, E, w)$, where $w : E \to \mathbb{R}$ is a weight function that assigns each edge $e \in E$ a positive real-valued weight. Various characteristics of the relationships between nodes, such as intensity, frequency or as in our case, the difference of balances between the involved users can be represented by them so as to further enrich the information conveyed by each edge in the graph with additional semantic meaning.

Based on the two definitions above, we explore the use of the following centrality measures.

Degree centrality is a simple centrality that counts the number of direct connections a node has. It shows how frequently a node interacts with other nodes in the network and nodes with high Degree are regarded as influential. Degree centrality can be expanded to a weighted version by summing the weights of all edges incident to a given node, as shown in Eq.(4). Furthermore, in directed graphs it can be further separated into in-degree and out-degree centrality, which take into consideration the quantity of incoming and outgoing edges, respectively.

$$CD(v) = \begin{cases} \deg(v), & \text{if the graph is unweighted} \\ \sum_{u \in N(v)} w_{vu}, & \text{if the graph is weighted} \end{cases} \tag{4}$$

where:

$\deg(v)$	= number of edges connected to node v.
$N(v)$	= set of neighboring nodes of v.
w_{vu}	= weight of the edge between nodes v and u.

Eigenvector [3,23] centrality expands upon the idea of simply counting a node's direct links by taking into account the importance of the nodes to which said node is connected. A node linked to other highly influential nodes will attain a higher eigenvector score. The measure assumes unweighted relationships by default but this can be easily modified to consider relationship weights. In this case, the score of a node in the previous iteration sent to its neighbors is multiplied by the scaled relationship weight. Negative weights are also ignored in this computation. Detailed formulation of the unweighted and weighted approaches are presented in Eq.(5) and Eq.(6), respectively.

$$EC_i = \frac{1}{\lambda} \sum_{j \in N(i)} A_{ij}\, EC_j \tag{5}$$

$$EC_i = \frac{1}{\lambda} \sum_{j \in N(i)} w^*_{ij}\, EC_j \quad \text{with} \quad w^*_{ij} = \frac{\max(w_{ij}, 0)}{\sum_{k \in N(i)} \max(w_{ik}, 0)} \tag{6}$$

where:

λ	$=$ largest eigenvalue of the adjacency matrix.
A_{ij}	$= (i, j)$ element of the adjacency matrix.
$N(i)$	$=$ set of neighbors of node i.
w_{ij}	$=$ weight of the edge from node i to j.
w^*_{ij}	$=$ normalized weight of the edge from node i to j.

Acting as extensions of the Eigenvector centrality, **PageRank** [6] and **ArticleRank** [28] were developed to assess a node's relevance according to its relationships as well. PageRank, which was originally developed to rank web pages, assigns scores by considering a random-walk based approach and assigning higher scores to nodes that have a high probability of appearing in any random walk in the graph. Following this approach, higher PageRank values are given to nodes that receive links from nodes with a high ranking as well. ArticleRank follows a similar principle when it comes to academic citations, where each citation is treated as a vote of importance. Citations from influential articles are considered of greater importance. Eq.(7) and Eq.(8) present the formal equations for PageRank and ArticleRank respectively.

$$PR(u) = \frac{1-d}{|V|} + d \sum_{v \in N_{in}(u)} \frac{PR(v)}{out\_deg(v)} \tag{7}$$

$$\mathrm{AR}_i(u) = (1-d) + d \sum_{w \in N_{in}(v)} \frac{\mathrm{AR}_{i-1}(w)}{|N_{out}(w)| + \overline{out\_deg}} \tag{8}$$

where:

d	$=$ damping factor $\in [0, 1]$
V	$=$ set of nodes (pages) in the graph.
$N_{in}(u)$	$=$ set of nodes linking to u.
$N_{out}(u)$	$=$ set of nodes that node u links to.
$out\_deg(u)$	$=$ number of outgoing edges from node u.
$\overline{out\_deg}$	$=$ average out-degree.

Both can also be adapted to weighted versions similarly to Eigenvector centrality.

Another popular centrality metric in network analysis is the **Hyperlink-Induced Topic Search (HITS)** algorithm [25]. Designed also initially to support web search, it however has considerable differences with PageRank. As mentioned, PageRank assesses the overall significance of a node, while HITS differentiates between two specialized roles: hubs and authorities. Nodes with high hub scores are treated as efficient aggregators that link authoritative sources, while nodes with high authority ratings are considered reliable sources of information. By evaluating roles separately, HITS provides a more sophisticated understanding of structural functionality in the graph. Eq. (9) and (10) show the results of the hub and authority score computations for the weighted graph, respectively. They can easily be applied to the unweighted graph if we consider all edge weights equal to 1.

$$h_v = \sum_{u \in N_{out}(v)} a_u \cdot w(v,u) \tag{9}$$

$$a_v = \sum_{u \in N_{in}(v)} h_u \cdot w(u,v) \tag{10}$$

where:

$w(v,u)$ = weight of the link between node v and node u.

Betweenness [17,4,5] centrality measures a node's importance based on how frequently it appears on the shortest paths between any given pair of nodes. It showcases to what extent a node acts as a bridge in the graph, enabling communications between nodes. Nodes with high betweenness scores are thought to be essential to the flow of information. Eq.(11) presents the formula for calculating the betweenness centrality score.

$$C_B(v) = \sum_{s,t \in V \ s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}} \tag{11}$$

where:

$\sigma_{st}(v)$ = number of those paths that pass through node v.
σ_{st} = total number of shortest paths from node s to node t.

The measure can be adopted for weighted graphs if, instead of just counting the number of shortest paths a node belongs to, we add the total edge weight of each such path, so as to account for its importance.

Closeness [18] centrality evaluates how close a node is to all other nodes in the graph. In its simplest form, it is defined as the inverse of the sum of the shortest path length from the given node to all other reachable nodes. However, normalizing this score, as presented in Eq.(12), to reflect the average of length of the shortest paths rather than their sum is more common.

$$C_C(v) = \frac{n-1}{\sum_{u \in V \setminus v} d(v,u)} \tag{12}$$

where:

$d(v,u)$ = shortest path length between node v and node u.

Similarly to betweeness, we can also adjust closeness centrality for weighted graphs by considering the total path weight instead of its length in our calculations.

The **Cost-Effective Lazy Forward (CELF)** [27] algorithm is an optimization-based greedy approach often used in influence maximization problems. It ranks nodes according to their marginal gain in a specific influence function, such as spread in a social network. CELF score at each step, given a monotonic submodular function f, is determined by Eq.(13) for both unweighted and weighted graphs.

$$\Delta f(v \mid S) = f(S \cup v) - f(S) \tag{13}$$

where:

$\Delta f(v \mid S)$	= marginal gain from adding v to S.
f	= influence spread function or centrality objective.
S	= current set of selected nodes.
v	= candidate node.

4.3. Implementation

To support efficient KG construction and graph analysis, we implement both KG models in a graph database so as to exploit its querying and indexing capabilities that offer scalability and high performance. In particular, we use Neo4j[1], which is one of the most popular graph databases. Neo4j uses the property graph model that aligns with the proposed model for our KGs and furthermore, offers the Graph Data Science Library (GDSL) that is a library designed to support network analysis and provides methods for the efficient evaluation of several centrality measures including the ones we have described above that we use in our experimental evaluation.

5. Experimental Evaluation

We experimentally evaluate the proposed approach both with respect to its effectiveness, i.e., the detection accuracy, and its efficiency in terms of time efficiency. We first establish a baseline approach using only tabular transactional data as features and compare the performance of our approach against it.

5.1. Experiment Setup

For the evaluation of the effectiveness of our approach, three well-known metrics suited for our problem are employed: Precision, Recall and F1-Score, Eq.(14), Eq.(15) and Eq.(16), respectively. In addition, the Receiver Operating Characteristic (ROC) Curve is utilized to visually assess the trade-off between the True Positive Rate (TPR) and the

[1] Neo4j

False Positive Rate (FPR) across various classifiers.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{14}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{15}$$

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{16}$$

where:

TP = true positives.
FP = false positives.
FN = false negatives.

The primary objective of CCFD is to minimize the amount of fraudulent transactions that go undetected, with emphasis on the accurate determination of such cases. Among the different metrics, recall is the most important, since it quantifies the percentage of correctly detected fraudulent transactions. Due to the nature of our problem, reducing false negatives - cases in which fraud activity is overlooked - is of most importance. On the other hand, precision illustrates the percentage of illegitimate transactions correctly identified, among all of the observations flagged by our model. As a result, it focuses on the number of false positives, which are valid transactions wrongfully classified as fraud. Given the context, we prioritize algorithms that maximize recall, even at the slight cost of precision. It is crucial to note that, because of the dataset's imbalance we opted not to include Accuracy in our analysis.

We utilized 5-fold cross-validation, a common approach that divides the dataset into training and testing subsets. However, to reduce the effects of the aforementioned class imbalance, especially the under-representation of fraudulent cases, we used stratified sampling. This method ensures equal proportional representation of all classes in all different folds.

Dataset One of the biggest challenges, in the domain of CCFD and FD in general, is the scarcity of publicly available real-world datasets, which is predominantly the result of growing privacy concerns. Due to that reason, research has shifted towards the usage of creative solutions, such as generating synthetic datasets for ML model training. Eliminating any possible identifiable personal information, these synthetic datasets adhere with major privacy protection frameworks, including the European Union's General Data Protection Regulation (GDPR) [32,11] and the California Consumer Privacy Act (CCPA). Since synthetic data is completely anonymized with no real-world identifiers, it can be used and distributed without breaking privacy laws.

For this study, we use the Synth Financial Dataset for Fraud Detection [29], generated using the PaySim mobile money simulator [30]. Paysim attempts to generate synthetic transaction data that realistically mirror the behavior of real-world mobile money systems. It replicates different transaction types and simulates user-agent interactions using probabilistic modeling based on real transaction records from a mobile service provider in an African country. Those agents perform one of five financial actions: *CASH-IN*, *CASH-OUT*, *DEBIT*, *PAYMENT*, or *TRANSFER*, depending on behavior rules and restrictions,

Table 1. Overview of the synth dataset attributes. In the table, we define abbreviations using (), which are to be used later when referring to those features

Attribute	Description	Possible Values	Example
Step (9)	Simulation time step (in hours)	1 – 744	0
Type (8)	Transaction type	`CASH-IN, PAYMENT, TRANSFER, etc.`	CASH-OUT
Amount (7)	Value of the transaction	Any positive float	4782.71
NameOrig (1)	ID of transaction initiator	Alphanumeric string	B90293651
OldbalanceOrg (2)	Sender's balance before transaction	Any float	75631.32
NewbalanceOrg (3)	Sender's balance after transaction	Any float	12398.65
NameDest (4)	ID of transaction receiver	Alphanumeric string	X239613845
OldbalanceDest (5)	Receiver's balance before transaction	Any float	31231.12
NewbalanceDest (6)	Receiver's balance after transaction	Any float	93841.43
isFraud	Fraud label (ground truth)	0 = No, 1 = Fraud	0
isFlaggedFraud	Flag if over threshold (e.g., 200K)	0 = No, 1 = Yes	0

such as daily withdrawal limits that an account may have. This framework enables PaySim to produce transactional data that is incredibly realistic and flexible, closely resembling genuine financial transactions.

The synthetic dataset is composed of 11 attributes, as described in [30]. The *Step* feature represents hour simulation over the course of a 30-day period. Transaction category is identified by its *Type*, with the 5 aforementioned actions. *Amount* records the monetary value of the transaction. The originator (sender) and recipient (destination) are denoted by the *NameOrig* and *NameDest* respectively. *OldbalanceOrg*, *NewbalanceOrg*, *OldbalanceDest*, and *NewbalanceDest* provide the account balances of the originator and destination prior and after each transaction occurs. Fraudulent transactions are identified with the binary *isFraud* flag, whereas *isFlaggedFraud* marks transactions tagged as suspicious based on specific heuristics. Table 1 provides a sample entry and synopsis of these fields.

To improve computational efficiency, the original dataset was reduced to 25% of its original size. Of these, only 1142 entries were identified as fraud, accounting for just 0.108%, which indicates a notable class imbalance. Thorough examination of the dataset was conducted in order to tackle this issue. It was observed that, out of all transactional activities only in two, namely *CASH-OUT* and *TRANSFER*, fraudulent transactions occurred. Therefore, the remaining transaction types which did not exhibit any fraudulent behavior, were excluded. This resulted in a reduction of the downscaled dataset to about 50% while maintaining all fraudulent cases. Finally, attribute *isFlaggedFraud* which had a value of only 0, providing no real value for classification, was removed.

The Knowledge Graph (KG) constructed from the preprocessed dataset consisted of 1,006,918 nodes and 1,381,182 relationships, with the resulting database (including the KG structure and metadata) occupying approximately 4 GB of storage.

Table 2. Hyperparameter configurations for the evaluated ML models

Model	**Hyperparameters**	**Values / Ranges**
K-Nearest Neighbors	Number of neighbors (n)	{4, 5, 6}
	Weights	{uniform, distance}
Logistic Regression	Regularization parameter (C)	{1e-6, 1e-5, ..., 1e-4}
	Maximum iterations	{500, 1000, 1500}
Support Vector Machine	Kernel type	{linear, polynomial}
	Regularization parameter (C)	{1, 2}
Naïve Bayes	Variance smoothing	{1e-9, 1e-8, 1e-7}
	Class priors	{[0.4, 0.6], [0.3, 0.7], None}
Decision Tree	Split criterion	{gini, entropy}
	Maximum leaf nodes	{4, 5, 6}
Random Forest	Number of estimators	{50, 100, 150}
	Split criterion	{gini, entropy}
XGBoost	Number of estimators	{50, 100, 150}

All experiments were carried out on a Windows 10 with 16GB of RAM and AMD Ryzen 5 processor. The implementation was developed in Python 3.10, utilizing essential libraries such as scikit-learn and Neo4j. The source code needed to replicate our approach is publicly available on GitHub[2].

Classification Algorithms To expand upon existing research, this study leverages seven well established ML algorithms, each known for its efficiency in previous FD attempts. These include *K-Nearest Neighbors (KNN)*, which groups data points based on the majority class among their nearest neighbors; *Logistic Regression (LR)*, a linear model that calculates the likelihood of a binary result; *Support Vector Machine (SVM)*, which finds the best hyperplane which separates classes as much as possible; *Naïve Bayes (NB)*, a probabilistic classifier which assumes independence based on the Bayes theorem; *Decision Tree (DT)*, a tree-like structure of decisions dividing data into branches based on a criterion and feature values; *Random Forest (RF)*, an ensemble of DTs utilizing a technique called Bagging, averaging the results of those DTs to enhance prediction performance; *XGBoost*, a highly effective gradient boosting algorithm that builds DTs sequentially to correct previous errors and *Bagging* with a custom architecture which combines XGBoost, KNN and LR algorithms. Table 2 highlights the specific hyperparameters used for each model.

5.2. Baseline Performance

The establishment of a baseline, which serves as a reference point upon which we will compare our results is imperative and essential for all ML problems. In Table 3, we depict the performance metrics of the assessed classification models using only the original preprocessed dataset excluding the *isFlaggedFraud* column.

[2] KG-Fraud-Detection

Table 3. Baseline performance of ML models using the original features

Algorithm	Precision	Recall	F1-Score
K-Nearest Neighbors	0.830	0.436	0.572
Logistic Regression	0.798	0.382	0.516
Support Vector Machine	0.806	0.420	0.552
Naïve Bayes	0.044	0.322	0.074
Decision Tree	0.850	0.460	0.594
Random Forest	0.980	0.778	0.870
XGBoost	**0.984**	**0.826**	**0.898**
Bagging	0.982	0.812	0.889

Table 4. Confusion matrix for the XGBoost classifier

		Prediction Outcome	
		Legitimate	Fraudulent
Actual Value	Legitimate	91847	3
	Fraudulent	42	187

The classifiers are generally accurate on the positive predictions with few false positives, achieving much higher precision scores than recall, indicating that they miss a significant chunk of real fraud cases, which is far more important given the nature of our problem. This is particularly apparent in models such as KNN, SVM and LR, which maintain acceptable precision around 80% but exhibit relatively low recall from 0.38 to 0.43. NB underperforms significantly when compared to all other algorithms, especially in precision with a value as low as 0.044, indicating weak performance for the task.

Tree-based models on the other hand, show much more promising results with RF and XGBoost being the best among them. More specifically, XGBoost achieves the best scores across all metrics, with an impressive F1-Score of 0.898, correctly identifying the majority of fraudulent and legitimate transactions as indicated by the confusion matrix shown in Table 4, which also highlights the strong class imbalance between the two classes.

5.3. Evaluation of Detection Effectiveness

The impact of adding graph centrality measures into the FD pipeline is explored in this section. Our goal is to evaluate the influence these structural features have on model performance under various configurations. The analysis is divided into multiple subsections, each focusing on a different experimental dimension. We first start by assessing the different centrality metric combinations, followed by the effects of removing features initially in our dataset while keeping our centralities. Finally, we examine how resampling affects the outcomes and explore weighted centralities. In the following, centrality measures are abbreviated as: (D) Degree (unweighted), (E) Eigenvector (unweighted), (H)

Table 5. F1-Score performance using single centralities

Centrality	KNN	LR	SVM	NB	DT	RF	XGBoost	Bagging
(D)	0.669	0.678	0.515	0.210	0.626	0.873	**0.902**	0.898
(H)	0.576	0.528	0.459	0.076	0.659	0.876	**0.903**	0.9
(E)	0.577	0.516	0.461	0.075	0.659	0.876	**0.897**	**0.897**

Table 6. F1-Score performance using pairs of centralities

Combination	KNN	LR	SVM	NB	DT	RF	XGBoost
(D) + (E)	0.674	0.68	0.75	0.14	0.802	0.876	0.91
(D) + (H)	0.674	0.68	0.75	0.14	0.628	0.872	**0.92**

HITS, (P) PageRank, (B) Betweenness, (HC) Harmonic Closeness, (C) CELF, (WD) Degree (weighted), (WE) Eigenvector (weighted).

Combinations of Centralities In this set of experiments, we examine the performance of different unweighted centrality measure combinations when used as additional features in the ML pipeline. We assess the results for individual centralities and various combinations (pairs, triplets and quadruplets).

Initially, we examine performance for single centralities. Table 5 shows the results for Eigenvector, HITS and Degree centrality as these were the best performing centrality measures. First of all, we observe that even the addition of one centrality measure has a positive effect compared with the performance of the baseline approach, as both Degree and HITS reach an F1-Score higher than 90% for the best performing classifier XGBoost. We find that HITS and Degree alone yield good results, especially when paired with tree-based models, while Eigenvector individually lags behind the other two. Other metrics showed poor standalone performance, which did not justify any additional review and for that reason we omit their results.

Following the same idea, we also examined the performance when adding pairs of centrality measures into the classification features. We observed that most of the combinations did not result into a significant performance improvement. However, both combinations of Degree with Eigenvector or HITS returned promising results, as seen in Table 6. With relatively little variation in performance when it comes to simpler algorithms such as KNN, LR, SVM and NB, both combinations produce similar F1-Score. Interestingly, in DT, Degree with Eigenvector showcases a substantial difference of approximately 18%, suggesting that even though both centrality pairs are generally good predictors, model architecture may affect how effective they are. XGBoost with the pair of Degree and HITS slightly outperformed that of Degree and Eigenvector reaching the highest F1-Score of 0.92 compared to 0.91.

We then investigate several triplets of centralities. Table 7 provides a summary of the findings. Interestingly, combinations including HITS and Eigenvector showcase improved performance, particularly when combined with PageRank or CELF. XGBoost with HITS, Eigenvector and PageRank (or CELF) outperform all single centralities, achieving an F1-Score of 0.91.

Table 7. F1-Score performance using triplet combinations of centralities

Combination	KNN	LR	SVM	NB	DT	RF	XGBoost	Bagging
(D) + (H) + (P)	0.67	0.682	0.52	0.212	0.658	0.856	0.906	0.896
(D) + (H) + (B)	0.67	0.682	0.52	0.212	0.628	0.872	0.904	0.9
(D) + (H) + (HC)	0.67	0.684	0.522	0.212	0.658	0.856	0.906	0.9
(D) + (H) + (C)	0.67	0.682	0.52	0.212	0.628	0.868	0.904	0.896
(H) + (E) + (P)	0.578	0.662	0.518	0.076	0.658	0.854	**0.91**	0.9
(H) + (E) + (B)	0.578	0.642	0.514	0.076	0.658	0.864	0.902	0.9
(H) + (E) + (HC)	0.578	0.642	0.516	0.076	0.658	0.862	**0.91**	0.9
(H) + (E) + (C)	0.578	0.642	0.514	0.076	0.658	0.866	0.902	0.9
(E) + (B) + (C)	0.578	0.642	0.514	0.076	0.688	0.876	0.908	0.898

Table 8. F1-Score performance using quadruplet combinations of centralities

Combination	KNN	LR	SVM	NB	DT	RF	XGBoost	Bagging
(D) + (E) + (B) + (C)	0.67	0.682	0.518	0.212	0.688	0.864	**0.906**	0.902
(D) + (E) + (B) + (HC)	0.67	0.682	0.52	0.212	0.688	0.858	0.902	0.9
(D) + (E) + (B) + (P)	0.67	0.684	0.52	0.212	0.688	0.862	0.902	0.898
(H) + (E) + (B) + (C)	0.578	0.642	0.514	0.076	0.658	0.866	0.902	0.9
(D) + (H) + (E) + (B)	0.67	0.68	0.514	0.212	0.658	0.852	0.902	0.894

Furthermore, we also assessed four centrality combinations, which are outlined in Table 8. We note that the quadruplet of Degree, Eigenvector, Betweenness and CELF exhibits consistent performance across all classification models, and performs very well when used with XGBoost. The slight losses, of 1%, compared with the best triplet combinations, however, suggest a diminishing return with additional centralities, as seen when comparing the best performing algorithm XGBoost using triplets (HITS, Eigenvector and PageRank or Harmonic Closeness) and the aforementioned quadruplet.

To get a better understanding of the dynamics between precision and recall we present the scores for the top-performing single centrality (Degree), pair (Degree and HITS), triplet (HITS, Eigenvector and PageRank) and quadruplet (Degree, Eigenvector, Betweenness and Closeness) in Fig. 4. Across the top-performing models, precision remains consistently high (> 0.98%). Recall on the other hand, tends to be quite lower but improves when combinations of centralities are used instead of solo measures. The gap between metrics narrows slightly when more centralities are added, enhancing FD without sacrificing either metric. Figure 5 additionally depicts the corresponding ROC curve for each best centrality combination for all classifiers, showing that the models with multiple centralities in the feature set tend to have more steep rises in the True Positive Ratio (TPR) early on, suggesting faster identification of fraudulent transactions at lower False Positive Rates (FPR).

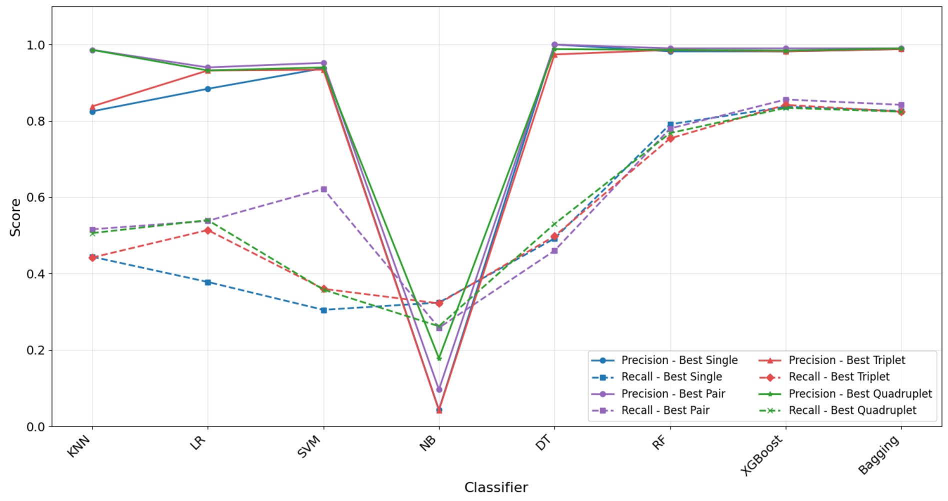

Fig. 4. Precision and recall for best single, pair, triplet, and quadruplet of centralities across all classifiers

Removal of Features In this experiment, we study feature selection with respect to the original transactional features of our dataset. We utilize the best pair of centralities as our network-based features, i.e., Degree with HITS, and evaluate the performance when successively removing attributes from the features vector. The impact of sequential initial feature removal on model performance, as measured by the F1-Score, is summarized in Table 9. Features are abbreviated using the numbers: NameOrig (1); OldbalanceOrg (2); NewbalanceOrg (3); NameDest (4); OldbalanceDest (5); NewbalanceDest (6); Amount (7); Type (8) and Step (9), defined previously in Table 1.

Initially, most models retain relatively good performance up until feature (4) is eliminated (around 0.1 decline). Almost all models show a progressive decline in performance as more features are removed on (5) through (7). Having said that, the ensemble techniques (RF, XGBoost, Bagging) demonstrate exceptional resilience, maintaining F1-Scores above 0.7 even after the removal of six features (7), which may be significant when considering that the cost of training the models with less features becomes more efficient.

Sampling To counter the effect of class imbalance, we evaluate the effect of deploying sampling techniques. As highlighted in Table 6, we include the results based on the best two centrality pairs: Degree with Eigenvector and Degree with HITS. As seen in Fig. 7, we use SMOTE [8] to evaluate the effects of data resampling and compare the outcomes before (Fig. 6) and after resampling. Even though comparable results can be seen between the two figures, we observe slightly lower performance, around 1%, in terms of F1-Scores after applying SMOTE on KNN, RF and XGBoost. On the other hand, a substantial difference can be observed in SVM with a 24% and 23% drop in performance on Degree with Eigenvector and Degree with HITS, respectively. Surprisingly, NB is the only algorithm which benefits from the usage of resampling with 7% increase on both combinations, but its overall low performance does not justify the use of sampling in our pipeline.

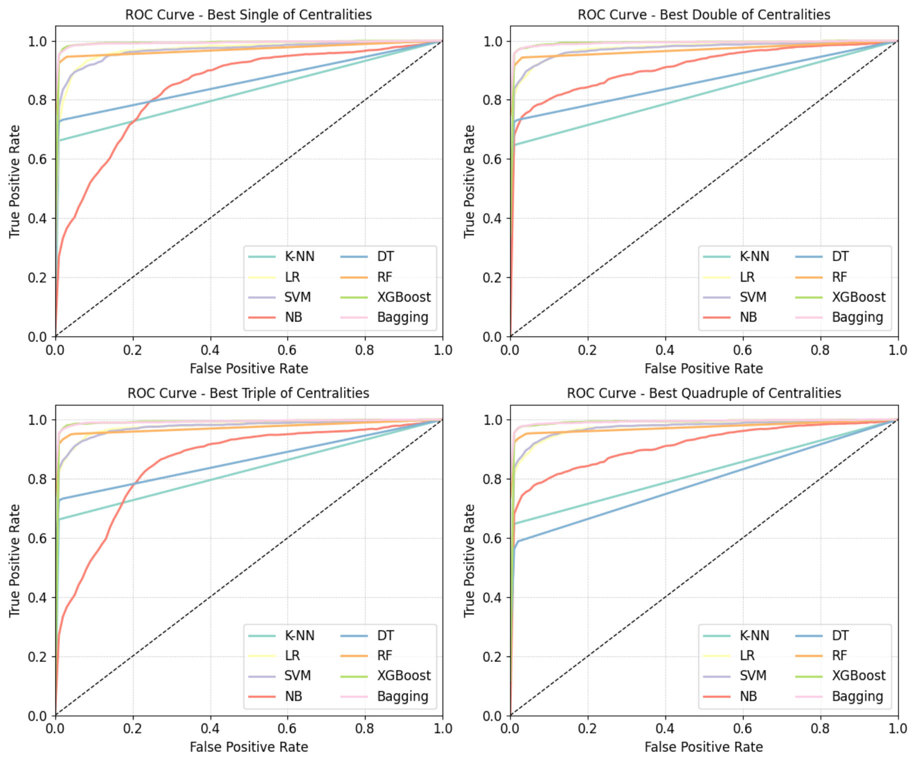

Fig. 5. ROC Curves for best centrality combinations across all classifiers. From left to right: best (1) single, (2) pair, (3) triple, and (4) quadruplet of centralities. The curves represent the mean true positive rate (TPR) across 5-fold cross-validation for the different classifiers, while the diagonal dashed line indicates the performance of a random classifier

Table 9. F1-Scores across models after sequential removal of features. Deleted feature sets are labeled (1) - (9). Empty cells indicate models not evaluated or not applicable

Deleted Feature(s)	KNN	LR	SVM	NB	DT	RF	XGBoost	Bagging
(1)	0.668	0.686	0.52	0.212	0.628	0.872	0.904	0.902
(1) - (2)	0.646	0.452	0.172	0.184	0.68	0.84	0.848	0.846
(1) - (3)	0.646	0.44	0.168	0.208	0.68	0.828	0.844	0.85
(1) - (4)	0.646	0.588	0.166	0.208	0.768	0.828	0.85	0.844
(1) - (5)	0.642	0.588	0.138	0.184	0.644	0.782	0.8	0.798
(1) - (6)	0.65	0.528	0.088	0.164	0.628	0.71	0.72	0.722
(1) - (7)	0.64				0.802	0.646	0.662	0.656
(1) - (8)	0.64				0.628	0.648	0.666	0.664
(1) - (9)	0.64				0.606	0.618	0.664	0.668

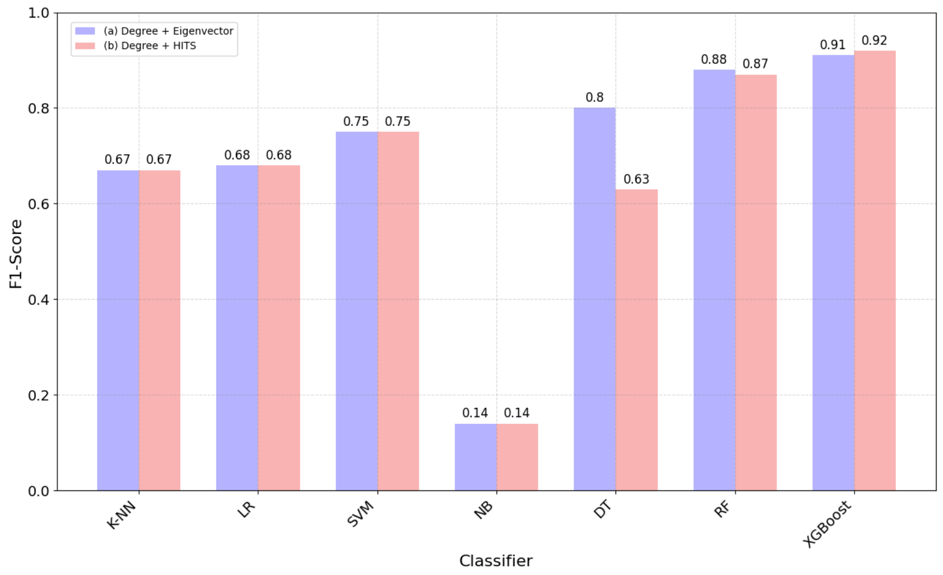

Fig. 6. F1-Scores without SMOTE for each centrality combination across classifiers

Table 10. F1-Scores across classifiers for different **weighted** centrality combinations

Method	**K-NN**	**LR**	**SVM**	**NB**	**DT**	**RF**	**XGBoost**	**Bagging**
(WD)	0.66	0.68	0.50	0.25	0.63	0.88	0.91	0.90
(WE)	0.58	0.53	0.51	0.08	0.71	0.88	0.90	0.91
(WD) + (WE)	0.66	0.66	0.51	0.24	0.71	0.89	0.90	0.91
(WD) + (H)	0.66	0.67	0.51	0.25	0.66	0.87	0.90	0.89
(WD) + (WE) + (H)	0.67	0.67	0.52	0.25	0.66	0.86	**0.92**	0.91

Weighted Model The introduction of weights into our graph shows promising results, as most classifiers benefit from them to a certain degree (Table 10). Among the different combinations of weighted centralities explored, only Weighted Degree (WD) and Weighted Eigenvector (WE) were retained for reporting, as they significantly outperformed the rest. WD already provides strong performance, notably for XGBoost with an F1-Score of 0.91. WE individually performs slightly worse than WD for simpler models such as KNN and LR, while SVM and DT show an increase of 1% and 8%, respectively. Both weighted metrics perform slightly better than their unweighted counterparts with around 1% increase, suggesting that the introduction of transaction-based weights assists in the identification of fraudulent from legitimate transactions. The addition of HITS (H) to the weighted centralities, enhances even further the performance achieving the best overall F1-Score of 0.92. Finally, it is worth noting that our custom Bagging architecture provides results comparable with XGBoost and in some cases exceeding them ((WD) + (WE)).

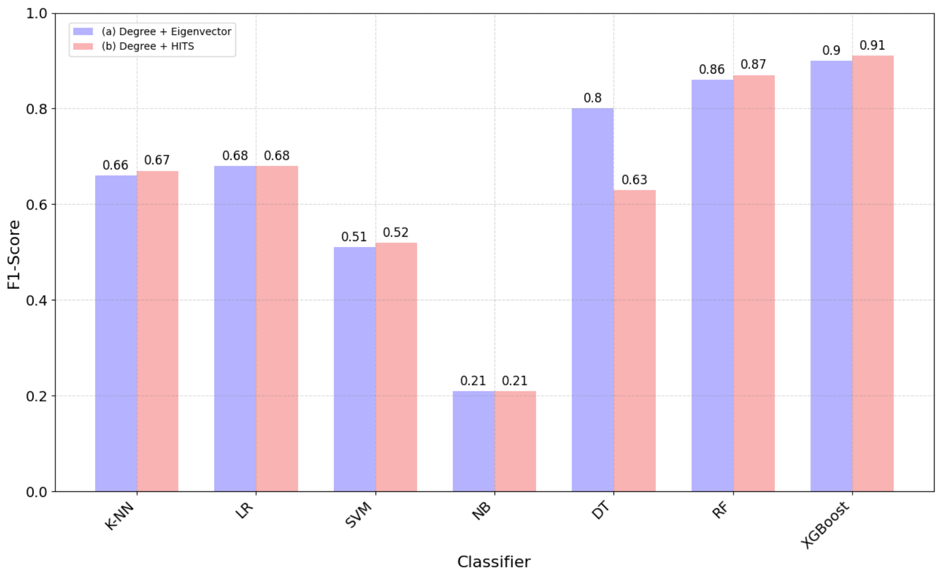

Fig. 7. F1-Scores after SMOTE for each centrality combination across classifiers

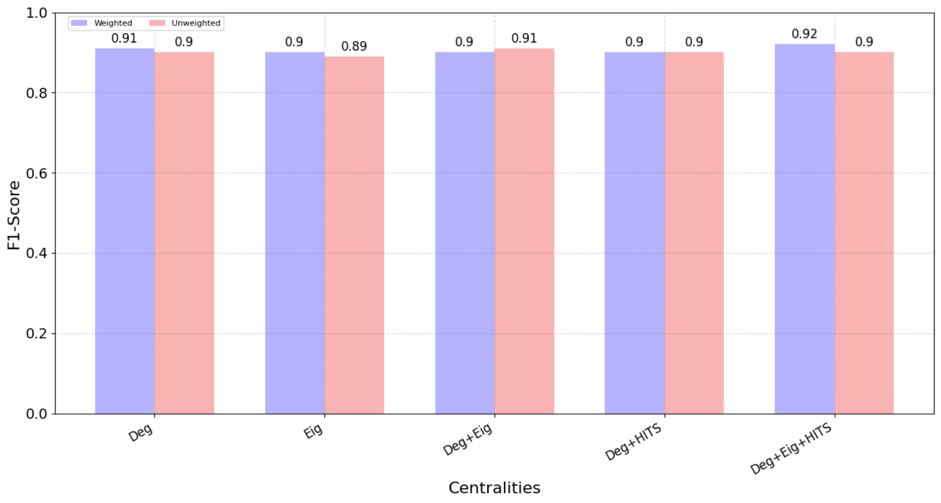

Fig. 8. Comparison of F1-Scores for XGBoost: weighted vs unweighted centralities

In Fig. 8, we provide a side-by-side comparison of the F1-Scores for the highest performing algorithm XGBoost, for both the weighted centralities and their unweighted counterparts, showing the slight superiority of the weighted model that leads us to conclude that the exploration of alternative weighting schemes could also be another research direction worth investigating in the use of KGs for fraud detection.

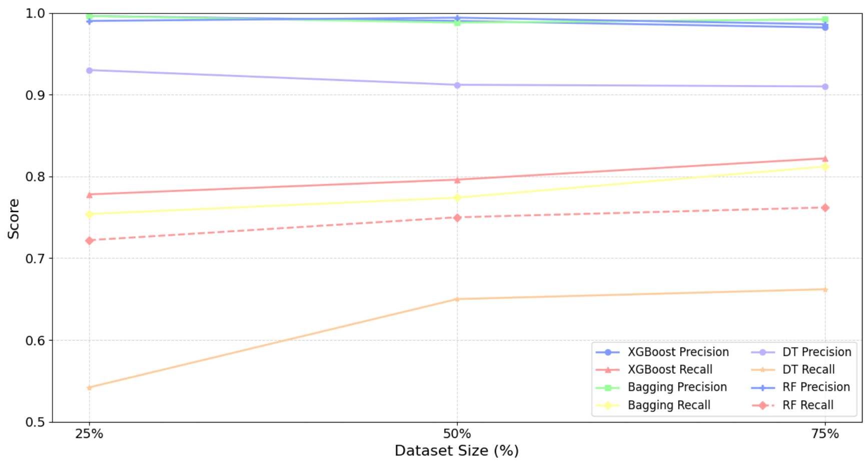

Fig. 9. Precision and Recall of various classifiers across different dataset sizes

5.4. Evaluation of Efficiency and Scaling

Finally, in this section, we evaluate the scaling abilities and time efficiency of the proposed approach. In particular, we examine the behavior of the suggested methods with a constantly increasing dataset size, with emphasis on both predictive performance and scalability. Figure 9 illustrates how various algorithms exhibit excellent resilience to data volume increase while maintaining high precision and recall, for the pair of Degree and HITS, our best pair combination as exhibited in the previous experiments. Recall somewhat improves with increasing data availability (around 0.05), indicating improved generalization for larger dataset sizes on ensemble algorithms such as RF, XGBoost and Bagging. Interestingly, DT benefits even further with an increase of approximately 0.12.

Analysis of execution time is presented in Fig. 10, which reveals significant variations among classifiers. Bagging and RF exhibit notable increases of computational cost, around 55% and 46% from 25% to 75% of dataset size increase, respectively. On the other hand, all other algorithms remain relatively close in terms of additional overhead w.r.t dataset size increase.

Figure 11 illustrates the change in execution time as individual features are removed based on the corresponding experiment we have described. Notably ensemble methods such as RF and Bagging display noticeably decreases in execution times, with Bagging's runtime decreasing from more than 200 to less than 40 minutes in the last stages of feature removal. In contrast, lightweight classifiers such as LR, NB and DT show minimal runtime variations. Thus, while feature removal does not seem to be worth the decrease in results quality for simpler classifiers, for more complicated ones, the small deterioration in F1-Score may be worth it considering the significant gains in efficiency when considering less features for training.

Finally, Table 11 shows that among the different computations in Neo4j, Betweenness centrality is by far the most expensive with 49 minutes, while Degree, Eigenvector and HITS are computed almost instantly, showing that the cost of exploring these additional features does not significantly impact the total approach performance as using a graph

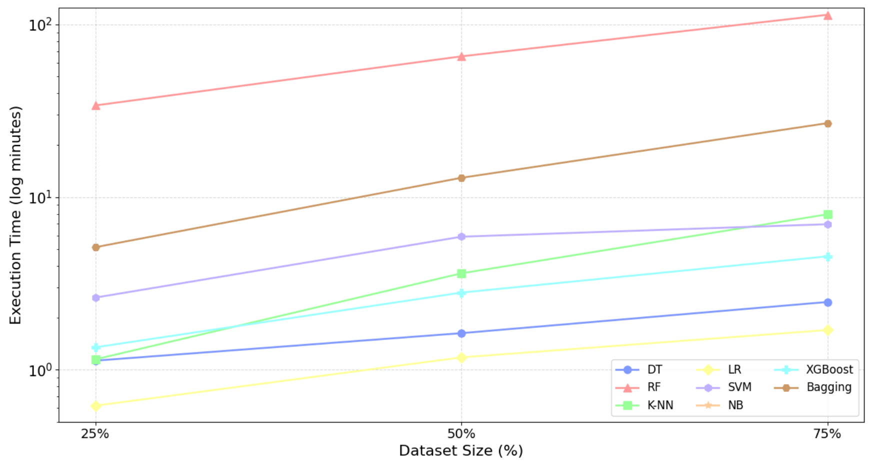

Fig. 10. Execution time (log-scaled minutes) of all classifiers across increasing dataset sizes

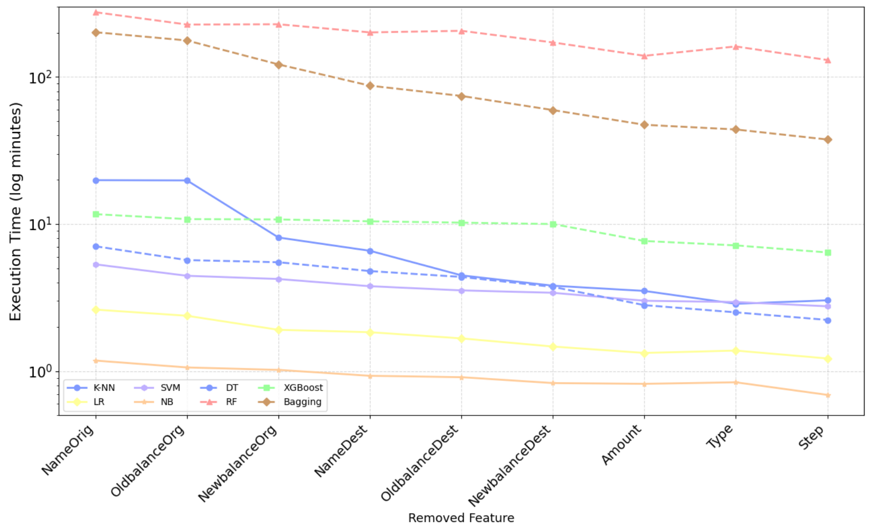

Fig. 11. Execution time (log-scaled minutes) variation as each feature is removed from the dataset

database for these computations practically alleviates the incurred overhead. Therefore, the additional costs are mainly due to the classifiers training as we explored in our previous experiments based on the dataset size, and the dimension of the features vector.

Table 11. Neo4j centrality computation times

Centrality Metric	**Degree**	**Eigenvector**	**HITS**	**Betweenness**	**Closeness**
Execution Time (min)	0.16	0.31	0.36	49.02	2.00

6. Conclusions and Future Work

In this study, we proposed a novel approach to address Credit Card Fraud Detection (CCFD) by utilizing a graph-based representation of the transaction network. Our methodology is focused on building a Knowledge Graph in which structural information obtained from centrality measures is added to the transaction data. Utilizing both unweighted and weighted centralities, specifically Degree, Eigenvector and HITS, we experimentally showed how the original feature vector is enhanced and demonstrates improved predictive performance across various classifiers. Our experimental results showed that, incorporating graph features, boost Fraud Detection effectiveness, with ensemble models such as XGBoost and our custom Bagging architecture consistently outperforming other models. The highest F1-Score was obtained from the combination of Degree and HITS and the use of transaction-based weights further improved the classifiers ability to distinguish between fraudulent and legitimate transactions. We also tested the scalability of our approach, demonstrating consistent performance as data volume increases, while extensive computation analysis provided valuable insights into trade-offs between feature richness and execution time.

In our future plans, we intend, in order to obtain richer end-to-end representations directly form the graph structure, to investigate the usage of graph neural networks (GNNs) in our ML pipeline. Furthermore, the exploration of temporal graph modeling, to dynamically update centrality measures for evolving fraud behavior and support real-time detection, could hold promising results, while a deeper exploration of different weighted centrality measures for specific classifiers could also be of interest.

References

1. Akoglu, L., McGlohon, M., Faloutsos, C.: oddball: Spotting anomalies in weighted graphs. In: Zaki, M.J., Yu, J.X., Ravindran, B., Pudi, V. (eds.) Advances in Knowledge Discovery and Data Mining. pp. 410–421. Springer Berlin Heidelberg (2010)
2. Alarfaj, F.K., Malik, I., Khan, H.U., Almusallam, N., Ramzan, M., Ahmed, M.: Credit card fraud detection using state-of-the-art machine learning and deep learning algorithms. IEEE Access 10, 39700–39715 (2022)
3. Bonacich, P.: Technique for analyzing overlapping memberships. Sociological Methodology 4, 176–185 (1972), http://www.jstor.org/stable/270732
4. Brandes, U.: A faster algorithm for betweenness centrality*. Journal of Mathematical Sociology 25(2), 163–177 (6 2001), https://doi.org/10.1080/0022250x.2001.9990249
5. Brandes, U., Pich, C.: Centrality Estimation in Large Networks. International Journal of Bifurcation and Chaos 17(07), 2303–2318 (7 2007), https://doi.org/10.1142/s0218127407018403
6. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. Computer Networks and ISDN Systems 30(1), 107–117 (1998)

7. Charizanos, G., Demirhan, H., İçen, D.: An online fuzzy fraud detection framework for credit card transactions. Expert Systems with Applications 252, 124127 (2024), https://www.sciencedirect.com/science/article/pii/S095741742400993X
8. Chawla, N.V., Bowyer, K.W., Hall, L.O., Kegelmeyer, W.P.: Smote: synthetic minority oversampling technique. J. Artif. Int. Res. 16(1), 321–357 (Jun 2002)
9. Cherif, A., Ammar, H., Kalkatawi, M., Alshehri, S., Imine, A.: Encoder–decoder graph neural network for credit card fraud detection. Journal of King Saud University - Computer and Information Sciences 36(3), 102003 (2024), https://www.sciencedirect.com/science/article/pii/S1319157824000922
10. Cherif, A., Badhib, A., Ammar, H., Alshehri, S., Kalkatawi, M., Imine, A.: Credit card fraud detection in the era of disruptive technologies: A systematic review. Journal of King Saud University - Computer and Information Sciences 35(1), 145–174 (2023), https://www.sciencedirect.com/science/article/pii/S1319157822004062
11. Commission, E., Centre, J.R., Hradec, J., Craglia, M., Di Leo, M., De Nigris, S., Ostlaender, N., Nicholson, N.: Multipurpose synthetic population for policy applications. Publications Office of the European Union (2022)
12. Dhankhad, S., Mohammed, E., Far, B.: Supervised Machine Learning Algorithms for Credit Card Fraudulent Transaction Detection: A Comparative Study. In: 2018 IEEE International Conference on Information Reuse and Integration (IRI). pp. 122–125 (2018)
13. Dimou, G.K., Koloniari, G.: Enhancing credit card fraud detection using knowledge graphs and centralities. In: Bădică, C., Gušev, M., Iftene, A., Ivanović, M., Manolopoulos, Y., Xinogalos, S. (eds.) Advances in ICT Research in the Balkans. pp. 14–29. Springer Nature Switzerland, Cham (2025)
14. Ding, Q., Katenka, N., Barford, P., Kolaczyk, E., Crovella, M.: Intrusion as (anti)social communication: characterization and detection. In: Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 886–894 (2012)
15. Dornadula, V.N., Geetha, S.: Credit Card Fraud Detection using Machine Learning Algorithms. Procedia Computer Science 165, 631–641 (2019)
16. Ehrlinger, L., Wöß, W.: Towards a definition of knowledge graphs. In: International Conference on Semantic Systems (2016), https://api.semanticscholar.org/CorpusID:8536105
17. Freeman, L.C.: A set of measures of centrality based on betweenness. Sociometry 40(1), 35–41 (1977), http://www.jstor.org/stable/3033543
18. Freeman, L.C.: Centrality in social networks conceptual clarification. Social Networks 1(3), 215–239 (1978), https://www.sciencedirect.com/science/article/pii/0378873378900217
19. Gupta, P.: Leveraging machine learning and artificial intelligence for fraud prevention. International Journal of Computer Science and Engineering 10(5), 47–52 (5 2023)
20. Hooi, B., Song, H.A., Beutel, A., Shah, N., Shin, K., Faloutsos, C.: Fraudar: Bounding graph fraud in the face of camouflage. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 895–904 (2016)
21. Ikhsan, W., Ednoer, E., Kridantika, W., Firmansyah, A.: Fraud detection automation through data analytics and artificial intelligence. Riset 4, 103–119 (Sep 2022)
22. Kamuangu, P.: A Review on Financial Fraud Detection using AI and Machine Learning. Journal of Economics, Finance and Accounting Studies 6, 67–77 (Feb 2024)
23. Kendall, M.G.: Further contributions to the theory of paired comparisons. Biometrics 11(1), 43 (3 1955), https://doi.org/10.2307/3001479
24. Khanum, A., K S, C., Singh, B., Gomathi, C.: Fraud detection in financial transactions: A machine learning approach vs. rule-based systems. In: 2024 International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE). pp. 1–5 (2024)

25. Kleinberg, J.M.: Hubs, authorities, and communities. ACM Comput. Surv. 31(4es), 5–es (1999)
26. Koronaios, A., Koloniari, G.: Graph-based bitcoin fraud detection using variational graph autoencoders and supervised learning. Procedia Computer Science 257, 817–825 (2025), https://www.sciencedirect.com/science/article/pii/S1877050925008427, the 16th International Conference on Ambient Systems, Networks and Technologies Networks (ANT)/ the 8th International Conference on Emerging Data and Industry 4.0 (EDI40)
27. Leskovec, J., Krause, A., Guestrin, C., Faloutsos, C., VanBriesen, J., Glance, N.: Cost-effective outbreak detection in networks. In: Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 420–429. KDD '07, Association for Computing Machinery, New York, NY, USA (2007), https://doi.org/10.1145/1281192.1281239
28. Li, J., Willett, P.: Articlerank: A pagerank-based alternative to numbers of citations for analysing citation networks. Aslib Proceedings 61, 605–618 (11 2009)
29. Lopez-Rojas, E.: Synthetic financial datasets for fraud detection (4 2017), https://www.kaggle.com/datasets/ealaxi/paysim1
30. Lopez-Rojas, E.A., Elmir, A., Axelsson, S.: Paysim: A financial mobile money simulator for fraud detection (2016)
31. Prasad, P.Y., Chowdary, A.S., Bavitha, C., Mounisha, E., Reethika, C.: A comparison study of fraud detection in usage of credit cards using machine learning. In: 2023 7th International Conference on Trends in Electronics and Informatics (ICOEI). pp. 1204–1209 (2023)
32. Raghunathan, T.E.: Synthetic data. Annual Review of Statistics and Its Application 8(Volume 8, 2021), 129–140 (2021), https://www.annualreviews.org/content/journals/10.1146/annurev-statistics-040720-031848
33. Rahman, M.J., Zhu, H.: Predicting accounting fraud using imbalanced ensemble learning classifiers – evidence from China. Accounting & Finance 63(3), 3455–3486 (2023)
34. Rej, M.: Credit card fraud Statistics (2 2025), https://merchantcostconsulting.com/lower-credit-card-processing-fees/credit-card-fraud-statistics/
35. S, M.R., V, N., N, M., B, P., M, R.A.: The impact of machine learning algorithms on credit card fraud detection: A comparative study. In: 2025 International Conference on Visual Analytics and Data Visualization (ICVADV). pp. 1576–1580 (2025)
36. Saleh Hussein, A., Salah Khairy, R., Mohamed Najeeb, S.M., Alrikabi, H.T.: Credit card fraud detection using fuzzy rough nearest neighbor and sequential minimal optimization with logistic regression. International Journal of Interactive Mobile Technologies (iJIM) 15(05), pp. 24–42 (3 2021)
37. Sanchez-Lengeling, B., Reif, E., Pearce, A., Wiltschko, A.B.: A gentle introduction to graph neural networks. Distill 6(9), e33 (2021)
38. Schneider, E.W.: Course modularization applied: The interface system and its implications for sequence control and data analysis. ERIC (10 1973), https://eric.ed.gov/?id=ED088424
39. Singh, A., Singh, A., Aggarwal, A., Chauhan, A.: Design and implementation of different machine learning algorithms for credit card fraud detection. In: 2022 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME). pp. 1–6 (2022)
40. Varmedja, D., Karanovic, M., Sladojevic, S., Arsenovic, M., Anderla, A.: Credit card fraud detection - machine learning methods. In: 2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH). pp. 1–5 (2019)
41. Vejalla, I., Battula, S.P., Kalluri, K., Kalluri, H.K.: Credit card fraud detection using machine learning techniques. In: 2023 2nd International Conference on Paradigm Shifts in Communications Embedded Systems, Machine Learning and Signal Processing (PCEMS). pp. 1–4 (2023)

42. Wen, S., Li, J., Zhu, X., Liu, M.: Analysis of financial fraud based on manager knowledge graph. Procedia Computer Science 199, 773–779 (2022)
43. Zaveri, A., Kontokostas, D., Hellmann, S., Umbrich, J., Färber, M., Bartscherer, F., Menne, C., Rettinger, A., Zaveri, A., Kontokostas, D., Hellmann, S., Umbrich, J.: Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago. Semant. Web 9(1), 77–129 (Jan 2018), `https://doi.org/10.3233/SW-170275`
44. Zhang, L., Wu, T., Chen, X., Lu, B., Na, C., Qi, G.: Auto Insurance Knowledge Graph Construction and Its Application to Fraud Detection. In: Proceedings of the 10th International Joint Conference on Knowledge Graphs. pp. 64–70. IJCKG '21, Association for Computing Machinery, New York, NY, USA (2022)
45. Zhao, T., Zhang, X., Wang, S.: Graphsmote: Imbalanced node classification on graphs with graph neural networks. In: Proceedings of the 14th ACM International Conference on Web Search and Data Mining. p. 833–841. WSDM '21 (2021)

George Konstantinos Dimou is an MSc student at Aristotle University of Thessaloniki and currently works as a Data and AI Consultant at Ernst & Young. He plans to continue his academic journey by pursuing a PhD in Artificial Intelligence.

Georgia Koloniari is an Associate Professor at the University of Macedonia at Thessaloniki, Greece and an ACM Member. She received her PhD in Computer Science from the University of Ioannina, Greece. Her current research interests include graph data management, entity resolution and social network analysis.

Received: April 05, 2025; Accepted: November 20, 2025.