User migration prediction in blockchain socioeconomic networks using graph neural networks

Cheick Tidiane Ba
cheick.ba@unimi.it
Computer Science Department,
University of Milan
Milan, IT

Alessia Galdeman
alessia.galdeman@unimi.it
Computer Science Department,
University of Milan
Milan, IT

Manuel Dileo
manuel.dileo@unimi.it
Computer Science Department,
University of Milan
Milan, IT

Matteo Zignani
matteo.zignani@unimi.it
Computer Science Department,
University of Milan
Milan, IT

Sabrina Gaito
sabrina.gaito@unimi.it
Computer Science Department,
University of Milan
Milan, IT

ABSTRACT

The growing popularity of online social media (OSM) has led to the creation of a wide amount of social media platforms. In this context, the increasing competition among platforms and the emergence of decentralized alternatives such as Blockchain Online Social Media (BOSM), have led to more frequent user migrations: individuals tend to switch platforms in search of improved features, content, or communities. Therefore there has been increasing interest in user migration studies modeling and predicting user migration. However, user migration, especially in blockchain-based platforms remains an understudied problem. Existing methods rely on user activity to derive interaction graphs and then address the user migration prediction problem as a node classification task, where user decisions are encoded as node labels. While the performance look promising, there are currently two important research gaps: *i)* there is no work using graph neural networks, the state-of-the-art in machine learning on graphs; and *ii)* there is a lack of methods designed to improve prediction performance in the case of class imbalance, i.e. the presence of dominant behavior among the ones to predict. In this paper, we propose a machine learning pipeline utilizing graph neural networks (GNNs) to predict user migration in BOSM. We model the data as a directed temporal multilayer graph, capturing social and monetary interactions among users. To address the problem of class imbalance in node classification, we introduce a data-level balancing technique following an undersampling approach. The evaluation, conducted on data describing user migration across blockchain online social media platforms, shows that graph neural networks are a suitable machine learning approach to perform user migration prediction. Furthermore, the proposed undersampling approach improves predictive power on severely imbalanced data. These results highlight how graph neural networks are effective in predicting user migration, without the need for manual feature engineering and in the absence of user information. Our methodology holds potential for applications beyond user migration, such as fraud detection and bot detection, and opens up venues for further research in other prediction tasks in online social networks and blockchain-based systems.

CCS CONCEPTS

• **Networks** → *Peer-to-peer networks*; **Network economics**; • **Applied computing** → **Digital cash**; **Electronic funds transfer**; **Economics**.

KEYWORDS

socio-economic network, blockchain, human behavior, transaction network, temporal network

ACM Reference Format:
Cheick Tidiane Ba, Alessia Galdeman, Manuel Dileo, Matteo Zignani, and Sabrina Gaito. 2023. User migration prediction in blockchain socioeconomic networks using graph neural networks. In *ACM International Conference on Information Technology for Social Good (GoodIT '23), September 06–08, 2023, Lisbon, Portugal.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3582515.3609552

1 INTRODUCTION

In recent years, online social media (OSM) has become an integral part of our lives, with almost the 60% of the world population being active social media users [30], connecting and sharing content with each other. As the popularity of these platforms grows, so does the competition among them to attract and retain users. In this highly competitive environment, user migration has emerged as a common phenomenon, with individuals switching from one platform to another in search of better features, content, or communities. In this context, the emergence of Web3 has given rise to more new online platforms that offer a decentralized alternative to traditional social networks. One such example is Blockchain Online Social Media (BOSM), which enables the redistribution of wealth generated by its users through rewards granted to those who contribute to the growth of the platform. User migration can also affect BOSM, with users switching between different blockchain-based social media platforms due to a variety of reasons such as ethical concerns, issues with the platform's infrastructure, or policy disagreements. Nonetheless, unlike traditional OSM, the user migration process in BOSM can be studied and measured thanks to the high temporal resolution data stored on the blockchains. These data are freely available and provide researchers with valuable insights into the dynamics of user behavior and migration in a decentralized social network.

Despite an increasing number of studies [1, 5, 21, 25, 27], user migration remains an understudied topic, particularly in BOSM platforms. One primary gap in the literature is the lack of methodologies for accurately predicting user migration, especially when there is a scarcity of user information or features. Existing methods that rely on interaction graphs built from user interactions show promise in addressing this challenge. Despite the graph-based representation of the task graph neural networks have not been yet applied to the user migration prediction, even though they have achieved state-of-the-art results in many machine learning tasks on graphs. This is an important research gap to be addressed in this context as GNNs do not require any feature engineering step on the interaction graph and they have shown their prediction power even without contextual information on users. Furthermore, user migration, as many other learning tasks on graphs, is often characterized by class imbalance, i.e. the target class sizes available in the dataset differ by a substantial margin [17], which can negatively impact the performance of machine learning models. The most used techniques primarily operate on the data level, aiming to modify the distribution of training data instead of altering the machine learning model itself. Typically, these methods utilize sampling-based approaches to tackle the issue of class imbalance. Existing methods often focus on oversampling-like strategies, differing in the methodologies for generating features, structures, or labels for the creation of artificial minority data instances [23]. However, in scenarios where there are sufficient data samples available for each class, the generation of new synthetic data is not preferable, as sampling may introduce bias. Interestingly, to the best of our knowledge, none of the current approaches address undersampling techniques. These aspects led us to the following research questions (RQs): *RQ1)* Are graph neural networks a suitable method for user migration prediction? *RQ2)* Can we improve performance in cases of severe class imbalance with a balancing method following an undersampling approach?

To fill these research gaps, we focused on predicting the phenomenon of user migration in the context of Blockchain Online Social Media (BOSM) platforms. Specifically, we design a machine learning pipeline to verify the effectiveness of graph neural networks for user migration prediction, where we model the data as a directed temporal multilayer graph describing social and monetary interactions among users to predict user behavior as a classification task. We also designed a data-level balancing technique following an undersampling approach, comparing the results within the same pipeline. To evaluate our methodology, we gathered data from the ecosystem of social platforms based on the Steem blockchain, whose main member is Steemit, and Hive, the blockchain originating from a hard fork of the Steem blockchain on March 20, 2020.

Our methodology for the selection of the best model and the proposed balancing approach have highlighted some interesting findings. Graph neural networks are an effective method to predict user migration in blockchain-based online social networks: the GNN model is able to leverage graph structure on the graph of monetary interactions, even with moderate data unbalance; however, the GNN model struggles on the graph of social interactions that is characterized by severe data imbalance (*RQ1*). However, after applying our proposed data-level balancing approach that produces a more balanced training set, graph neural networks show good predictive power even on severely imbalanced data (*RQ2*).

The paper is organized as follows. Section 2 provides a brief introduction to blockchain online social media and machine learning on graphs. In Section 3 we introduce the main research questions we focus on. In Section 4 we describe the dataset and its preprocessing. The methodology for modeling interaction data, performing user migration prediction, and the proposed balancing method is presented in Section 5. Section 6 reports the main findings on the effectiveness of graph neural networks and the impact of applying a balancing approach. Finally, Section 7 concludes the paper, pointing out possible future works.

2 BACKGROUND

Blockchain online social media and user migration. In the past decade, the world of Internet services has witnessed significant changes, where the focus has shifted from centralized services to decentralized and distributed approaches. One of these new paradigms, known as Web3 [29], strives to create a new type of web that leverages blockchain technology in various systems. Blockchain technology has led to Decentralized Finance (DeFi), Decentralized Autonomous Organisations (DAOs), and, in the social media context, Blockchain-based online Social Media (BOSM) [13]. One of the key innovations made possible by blockchain in social media is the concept of a cryptocurrency-based rewarding system, that can promote positive behaviors or high-quality content creation [8, 22]. While in traditional social media platforms, interactions among users are mainly social (such as sharing multimedia content, and interaction through comments or votes), in BOSM users have the ability to engage in "financial" or "monetary" interactions, i.e. the transfer of tokens from a source account to a destination account. Among the various proposals, **Steemit** [9, 13, 14], launched in 2016, has been one of the first and most successful platforms. Other platforms include **Hive Blog**, the primary web interface to Hive blockchain, developed by some users leaving Steemit, and other services built on Amazon Web Services that use the Ethereum blockchain to host their own ERC-20 tokens such as **Sapien** [26] and **Minds** [24].

User migration. User migration, i.e. the phenomenon of users moving from one online social platform to another, is a common occurrence in online social media. In the context of blockchain-based platforms, they are often associated with fork events. A fork event occurs when miners (validating nodes in the blockchain network) need to modify the consensus protocol, i.e. the set of rules for validating transactions and maintaining the network. A fork is a *soft fork* when miners make changes to the consensus protocol, while still ensuring compatibility with the previous protocol. In BOSM, these forks are typically used for making minor adjustments to the consensus protocol, freezing account funds, or reversing specific transactions. On the other hand, an *hard fork* introduces more significant changes in protocol, leading to a new distinct chain, that will reject blocks validated with a different protocol. This type of fork can support the creation of new platforms, such as in the case of Steemit, where some of its users created a new platform on the Hive blockchain with its own interface - Hive Blog - and cryptocurrency system.

User migration across online social media is a widely studied but not yet fully understood process, especially in blockchain-based systems since most studies are focused on traditional social media platforms. Kumar *et al.* [21], studied migration patterns across different platforms such as Twitter, Reddit, and Youtube after relying on external sources of information such as BlogCatalog to perform user account matching between platforms. Similarly, Newell *et al.* [25], focused on a survey-based approach to understand the reason behind migration events from Reddit to alternative websites. Also, Zia *et al.* [37] tracked and analyzed user migration from centralized Twitter to the decentralized microblogging platform Mastodon, highlighting the strong influence of the social network in platform migration. The same migration has been studied by La Cava *et al.* [4], showing the importance of the network structure of social interactions in migration. Senaweera *et al.* [27] studied migration patterns in Facebook groups using graph-based modeling. While Davies *et al.* [5] identified and quantified migration in Reddit, focusing on migration in COVID-19-related subreddits. Studies on user migration in blockchain-based platforms are still limited. Recent studies on user migration in Web3 include Galdeman *et. al.* [10], who studied the influence of hubs on the user migration decisions of their direct neighbors and found that users directly interacting with hubs tend to migrate, while Ba *et al.* [2], analyzed Steemit user migration from a mesoscopic perspective and observed different migration behaviors among communities. Finally, Ba *et al.* [1] have examined the effects of user migration on the graph structure of interactions in Web3 social platforms and evaluated the predictability of migrating using a graph structure. They treat the prediction of user migration as a binary node classification task, where two possible user future decisions (staying or leaving) are encoded as labels; the prediction is performed using user activity of social and monetary type modeled as interaction graphs, showing the predictive power of graph derived features such as PageRank in predicting user migration.

Machine learning on graphs. In the last decade, there has been a growing interest in developing machine learning techniques tailored for graphs to solve various tasks, such as node classification, link prediction, and graph generation. In this context, traditional approaches adopted a manual feature generation approach in order to get a vector of statistics for each node, that could later be fed into traditional learning models. However, these approaches are often time-consuming and inflexible as they cannot be adapted to the learning process. More recent approaches however rely on the concept of graph representation learning, i.e. encoding the structural information of nodes into a low-dimensional latent space. In the field of graph representation learning, graph neural networks (GNNs) have emerged as the state-of-the-art approach in many different tasks, such as node classification [15], link prediction [34], community detection [33] and graph classification [35]. GNNs were designed to perform predictions by exploiting both topology and graph attributes by redefining basic deep learning operations, such as convolution, for graph-structured data. The concept has been formalized as the *message passing framework* [11]: the convolution on graphs can be performed by aggregating the values of each node's features along with its neighboring nodes' features. One of the earliest examples is the Graph Convolutional Network (GCN) model proposed by [20]. Given a graph $G = (V, A, X)$ such that V is the set of vertexes, X is the node feature matrix, and A the adjacency matrix, at each layer k the embedding h of a node i is updated with the following computation:

$$h_i^{(k+1)} = \sigma\left(\sum_{j \in N(i)} \frac{1}{\sqrt{\widetilde{D}_{ii}\widetilde{D}_{jj}}} h_j^{(k)} W^{(k+1)}\right) \quad (1)$$

where $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$ corresponds to the degree of i, computed on A_{ij} the adjacency matrix with self-loops added. The aggregation is order-invariant, (examples of such functions are average or summation). The number of layers of a GNN defines the number of hops up to which a node will receive information. Starting from these, we have seen the proposal of many architectures such as GAT [28], graph autoencoders [19], GraphSAGE [15], and many more, to cover different tasks and types of graph data.

Class imbalanced learning on graphs. A classification problem is considered imbalanced when the target class sizes of a dataset differ relatively by a substantial margin [17]. There are several examples of real problems that are affected by this phenomenon, such as fraud detection, disease diagnosis, anomaly detection, and sentiment analysis. An imbalanced data sample can have a negative impact on the predictive performance of the model, especially for the minority classes. This is because the model has fewer opportunities to learn the characteristics of the samples within the minority classes, which can lead to poor generalization skills when applied to unseen testing data. Finally, a class imbalance can cause the model to be biased towards the majority classes, resulting in a tendency to predict the class with the larger number of instances. Class imbalance remains a challenging problem in machine learning, but there exist techniques and strategies that can be employed to mitigate its negative effects. Current approaches can be divided into two categories [23]: *i) data-level* methods, which modify the distribution of training data, and *ii) algorithm-level* methods, which modify learning algorithms. Acting at the data level is the most flexible approach as it allows the use of already available models. Data-level methods try to address the imbalance through sampling-based approaches [17]. Methods usually rely on under-sampling approaches to select a subset of instances from the majority classes or over-sampling approaches to create additional instances of the minority classes or even a mix of both (hybrid sampling). All those techniques have been designed on point-based data and have limitations when it comes to learning on graphs [36]: while in traditional cases it is just a matter of considering more or less independent data points, in graphs is more complicated, as removing nodes/edges will automatically modify the graph structure, and this can create issues during the model training, especially during the message-passing process in GNN models. On the other hand, adding a node requires managing both the node attributes and connectivity. As a result, some proposals have been made to address class-imbalanced learning on graphs, acting at both data-level and algorithmic-level. Current data-level methods are focused on oversampling-like approaches and they differ in their approach to generating features, structures, or labels for synthetically created minority data instances [23]. However, in cases where there are sufficient data samples for each class, generating new artificial data is not desirable, as it could introduce bias in the dataset. And yet, to the best of our knowledge, none of the current works addresses approaches following the undersampling approach.

3 RESEARCH QUESTIONS

The problem we address in this paper is user migration prediction, which has received limited attention in the context of Web3 platforms. While developing a machine learning pipeline to predict whether a user will migrate, stay on the original platform, on both, or become inactive, this work will answer two main research questions:
Research question 1 (RQ1): Are graph neural networks a suitable method for user migration prediction?
Research question 2 (RQ2): Can we improve performance in cases of severe class imbalance with a balancing method following an undersampling approach?
By answering these questions, we aim to contribute to the development of effective techniques for predicting user migration in Web3 platforms, which could have implications for improving user experience and enhancing platform design and management.

4 DATASET

Our work on user migration prediction using graph neural networks leverages as a case study the user migration following a hard fork event. As a case study, we consider data on interaction actions obtained from the blockchain of both *Steem* (the original blockchain) and *Hive* (the new descendant blockchain). Specifically, we focus on the hard fork that occurred on March 20th, 2020. This resulted in the creation of the Hive blockchain from Steem, which consequently enabled the phenomenon of user migration to take place. One advantage of studying user migration from Steem to Hive is the easy account matching phase. Due to the fact that users maintain the same usernames on both blockchain Hive and Steem, the issue of account matching that is typically encountered in user migration studies is non-existent, which allows for seamless tracking of user behavior before and after the fork event. The Steem and Hive blockchains support two social media platforms, *Steemit* and *Hive Blog*, respectively. Steemit was the original social media platform launched in 2016, where users can post and share multimedia content, and interact through comments and votes or by following other users. Thanks to a reward system, users can earn cryptocurrency tokens for high-quality and popular content. After the fork event, Hive Blog was born with similar characteristics. For both platforms, all interactions or operations that users perform are stored on the supporting blockchains and a complete list is available for both platforms [6, 7].

In this work, we focus on actions that represent an interaction between two users, either explicit or implicit. Specifically, we consider two main groups: *i) monetary* and *ii) social* operations. Monetary operations are those operations designated for the management of tokens, rewards, and asset transfer. In contrast, social operations are those that users are able to do on traditional social network platforms, such as posting, rating, voting, sharing, and following. All blocks and the corresponding operations can be gathered through official APIs for both platforms, whose structure and usage are similar. For the construction of the graph, we gathered operations from the very first block on the Steem blockchain, produced on 24th March 2016, up to the fork event, i.e. to block 41818752, with timestamp *2020-03-20T14:00:00*. While for migration status, we examine data after that timestamp, and up to January 2021. From there, Hive and Steem have different data, as they have become two different blockchains. Overall, from the Steem blockchain, we extract $993,641,075$ operations describing social interactions and $72,370,926$ operations describing economic interactions; from the Hive blockchain, we get a total of $206,224,132$ social operations and $4,041,060$ financial actions.

5 METHODOLOGY

Our objective is to leverage user interaction data to predict future user migration decisions. We utilize a similar setting to the one proposed in a previous work [1], where user migration is treated as a machine-learning task on graphs, using only the network structure of the graph to perform predictions, while user behavior is encoded in classes, allowing us to handle user migration as a multiclass node classification problem. In this section, we define the machine learning pipeline, that will be used to perform the user migration prediction task. Our proposed pipeline is presented in Figure 1.

In the following, we describe the methodology adopted in each step, which will allow us to leverage interaction data as input for machine learning models, to verify the effectiveness of graph neural networks in the setting of a user migration prediction task, as well as to address the class imbalance in datasets.

Modeling user interactions and user decisions: graphs and labels. User interactions can be modeled as a set of tuples $I = (u, v, t, r)$, where u and v are users, who explicitly or implicitly interact at time t through an action of type r. As we are interested in the graph structure before the fork, we can consider the interactions before t_{Fork} (March 20th, 2020, 2:00 PM), which we denote as $I_{t_{Fork}}$. From this subset of interactions, we are able to construct a temporal directed multilayer graph [16], that we denote as $\mathcal{G} = \{\mathcal{G}^r_{t_{Fork}} \forall r\}$, where each element $\mathcal{G}^r_{t_{Fork}}$ is a layer of the multilayer graph. More precisely for each interaction type r, a layer of the graph can be seen as a temporal weighted graph $\mathcal{G}^r_{t_{Fork}} = (V^r_{t_{Fork}}, E^r_{t_{Fork}})$ that stores the interactions of type r that happened up to t_{Fork}. Each edge $(u, v, t, c) \in E^r_{t_{Fork}}$ encodes the operations from node u to node v, described by the counter c and timestamp t. Specifically, the counter c keeps track of the number of operations within the directed pair of nodes, while the timestamp t corresponds to the time of the first operation from u to v. While the obtained graphs could be used to perform prediction on all users, they may have not been active before the fork, therefore it is important to filter users that stopped using the platform before the fork event. We define a set U of users of interest, in which we consider only users active before the fork while including new users that would appear in the following time period. Similarly to what has been done in [1], a user u belongs to the set U (therefore *active*) if it performed at least one operation in the 3 months before the fork event. In this way, we are able to

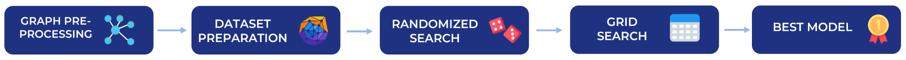


Figure 1: The proposed methodology to solve node classification tasks.

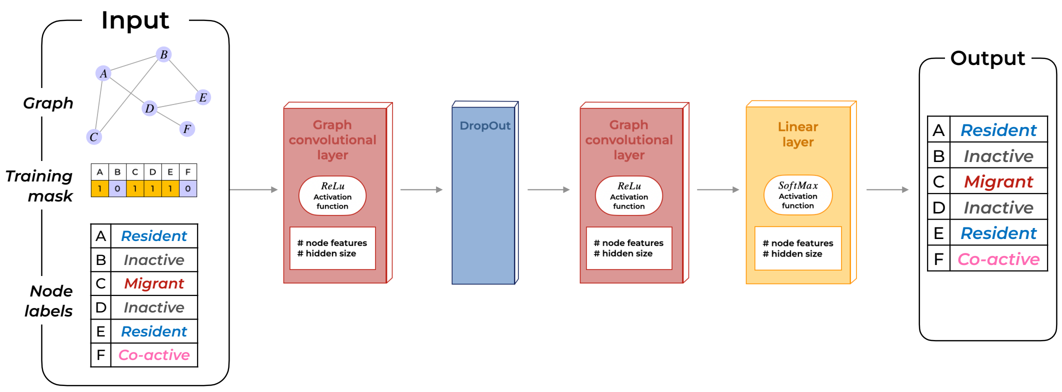


Figure 2: Representation of selected graph neural network architecture. The selected architecture is inspired by the classical GCN architecture by Kipf [20].

extract $G_{t_{fork}}$, i.e. the subgraph of $\mathcal{G}_{t_{fork}}$ induced by the set U of active nodes. If we consider the set of $r \in \{monetary(m), social(s)\}$, we can denote the layer graphs $G^m_{t_{Fork}}$ and $G^s_{t_{Fork}}$, representing monetary interactions and social interactions respectively, that will be leveraged to predict behavior after the fork. We then need to process interaction data to encode user behavior after the fork, in a way that can be learned by machine learning models. This means defining labels for each node based on the user activity after the fork. If we observe the interactions that happened after the fork event involving a user u, we can consider 4 possible cases:

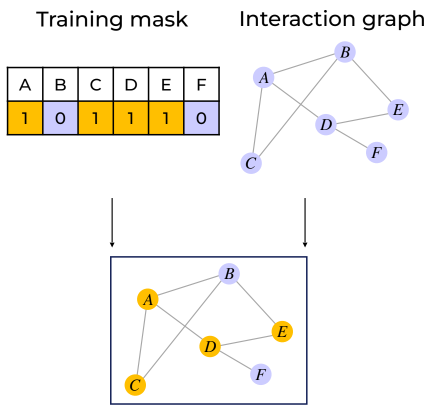
- *resident*: a user active only on the original platform (Steemit)
- *migrant*: a user active only on the new platform (Hive Blog)
- *co-active*: a user that performs actions on both platforms
- *inactive*: a user that stops using both platforms

These cases are defined at the end of an observation period, after the fork event, considering the activity up to the last interaction in the available data. So each user a is assigned to one of the four labels after observing the interactions $I = (u, v, t, r)$ where $u = a$ and with $t > t_{Fork}$. The assigned label (*resident*, *migration*, *co-active*, or inactive) is defined as the *migration decision* l of user u.

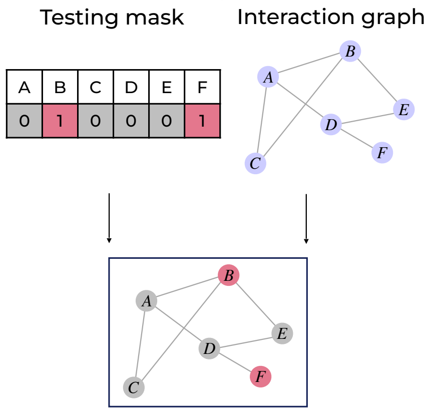
Leveraging graph neural networks: model training and best model selection. The first step is the selection of the architecture for our machine learning model: we selected the GCN architecture from [20]. We implemented a similar version, as represented in Figure 2. First, the input features and the adjacency matrix are then leveraged by two graph convolutional layers that create node embeddings. Finally, a linear transformation layer uses the embeddings generated by the GNN, to return a vector with a dimension equal to the number of target classes of the task. Then we obtain a vector representing a probability distribution on the target classes by applying to the output of the previous layer z a softmax function $\sigma(z)$.

The selected graph neural network model needs to be trained i.e. its weights need to be adjusted so that it can learn to predict the right classes. When the ground truth is available, GNNs can be trained in a supervised setting. For node classification tasks, supervised learning requires the so-called train-test split [12]. While in traditional machine learning tasks, the split requires the separation into two sets of training samples, when dealing with graphs, the split is not as straightforward: for graph neural networks, the training and test sets are defined as the creation of masks $M_1 \in \mathbb{R}^n$, like in Figure 3. The masks indicate which labels should be visible for the GNN model during training.

As in traditional supervised learning frameworks, the objective is to make the model output as close as possible to the ground truth values. This is done by adjusting model parameters through the data learning process to minimize a loss function. A common choice for classification problems is the cross-entropy loss function [20]. Alongside the model parameters, the selection of the best model configuration for the task requires the optimization or tuning of hyperparameters, i.e. parameters that can not be estimated from data learning and must be set before training an ML model because they define the model architecture [31]. Testing all the possible combinations of hyperparameters from the grid of possible parameters - *grid search* - can be a computationally demanding and time-consuming phase as there are many hyperparameters in GNN models, leading to a huge number of combinations to verify. For this reason, a popular strategy is to perform a *random search* [3]: only a subset, of possible hyper-parameter combinations, is chosen at random and tested. In this work, we combine the two approaches. After the first exploratory step is conducted with a random search, the best configurations are used to refine the candidate configurations, so that we can reduce the number of combinations before performing a full grid search.



(a)



(b)

Figure 3: Supervised training example. On the left side, the training mask is defined, while on the right side, an example of the corresponding test mask. Training and message passing is performed using the complete graph structure, but the loss function is computed only for training nodes. During testing, message passing is performed over the entire graph, but evaluation is conducted on test nodes.

Dealing with class imbalance: a new undersampling based approach. Formally, in a multiclass supervised learning task, there are m classes in total, $\{C_1, ...C_m\}$, and $|C_i|$ is the size of the i-th class, referring to the number of samples belonging to that class. Here, we introduce an under-sampling technique to balance the distribution of the target variable at the data level. Formally we balance the target variable as follows: we choose a percentage p, and compute the number of samples $n = min_i|C_i| * p$ to get the number of samples per class to include in the training set. To build a balanced training set, we perform under-sampling of each class C_i: we consider a random subset of cardinality n of samples, creating a uniform distribution. This leads to a reduced training set size, but each target class is equally represented. In Figure 4, we report a toy example with two classes. The selected method can be applied seamlessly in the pipeline we described previously in Figure 1.

Experimental setting. In this work, for both RQ1 and RQ2, we are interested in evaluating the performance of graph neural networks in the task of user migration. Performance can be evaluated with different *evaluation metrics.* We selected some of the most used metrics for multiclass classification problems, *accuracy* and *F1* [23]. Both metrics are computed from the evaluation of true positives (TP) and true negatives (TN) that represent the number of accurate classifications of positive and negative samples, while false positives (FP) and false negatives (FN) indicate the number of incorrect classifications of positive and negative samples. The $accuracy = \frac{TP+TN}{TP+TN+FP+FN}$ represents the relationship between observations correctly predicted and total observations. While the $F1 = \frac{2*TP}{2*TP+FP+FN}$, represents the average of $Precision = \frac{TP}{TP+FP}$ and $Recall = \frac{TP}{TP+FN}$. However, in multiclass classification, with C classes and N samples, F1 can be adjusted to account for each class size, leading to the *weighted F1* $= \sum_{i=1}^{C} w_i * F1(C_i)$, as the weighted average of class-wise F1 scores, where for each class C_i, we have a weight $w_i = \frac{|C_i|}{N}$. The metrics are evaluated both on training and test sets: to obtain more robust results. It is common in the literature to consider the average performance over multiple random seeds for each combination, therefore we report the average over 3 random seeds as done in [32]. Through the selected metrics, we compare the predictive performance of graph neural networks to two baseline classifiers: the *Uniform Baseline classifier* that generates predictions uniformly at random (hence it will make a correct prediction in around 1/4 of the cases) and the *Most Frequent Baseline classifier*, which predicts always the most frequent class observed in the training set.

For RQ1, data is separated into training and test set through a random train test split, with 70% of the nodes as a training set, and 30% of the nodes as a test set. Whereas we answer RQ2 by applying our under-sampling technique for balancing, generating various training and test sets, with different sizes. In this work, for both RQ1 and RQ2, we are interested only in the impact of network structure. Therefore, node attributes from the dataset are not considered in the prediction: a constant attribute (equal to 1) is associated with each node. The weight update over the training in this work is done by Adam optimizer [18].

6 RESULTS

In this section, we present the graph and labels obtained by applying the graph preprocessing methodology shown earlier. Then we show how we apply the proposed methodology to answer our research question.

Graph and labels. Applying the proposed methodology we obtain a multilayer graph $G_{t_{fork}}$. Note that $G_{t_{fork}}$ is the active users' subgraph, i.e. the subgraph induced by the set of active users on the two layers $r \in \{monetary(m), social(s)\}$. The monetary layer graphs $G^{m}_{t_{Fork}}$ contains 38, 566 nodes connected by 949, 046 edges. While the social layer graph $G^{s}_{t_{Fork}}$ has 90, 055 nodes and 42, 556, 877 edges, Overall, the social layer has more active users and links: this is consistent with the selected operations; in fact, social operations are far more common than monetary transactions. For these users, we encoded their behavior in the 4 possible classes whose frequencies are shown in Figure 5a for the monetary interactions and in

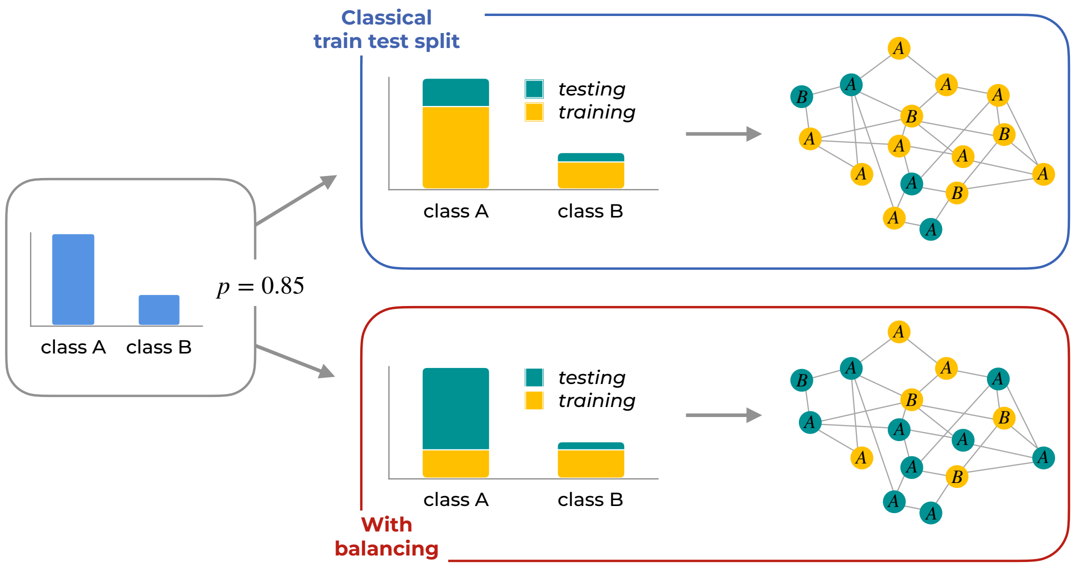
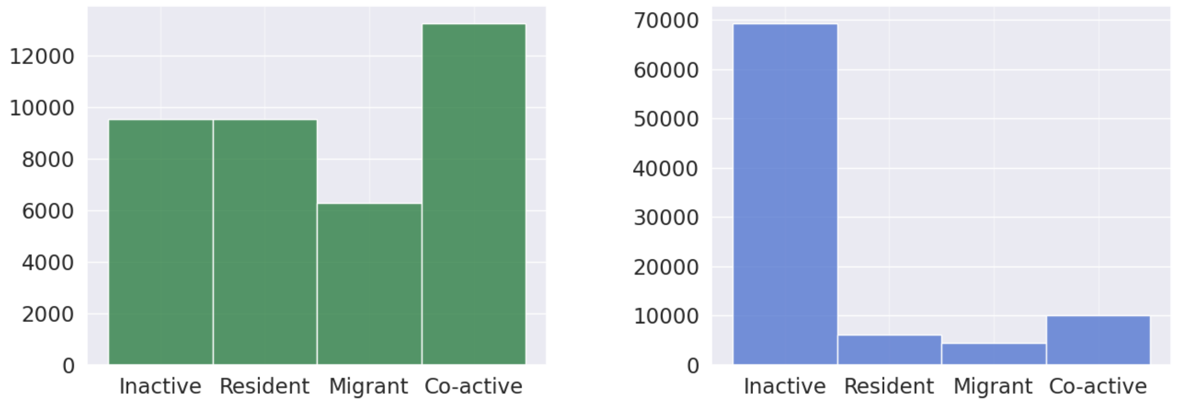


Figure 4: Train-test split with unbalanced classed. A visual example of an imbalanced dataset with 2 classes (A, B). On the top half, a representation of a classical 85/15 train-test split: in this case, the training set presents more examples of class A (9) than class B (3). On the lower half, we illustrate our proposed approach: we select 85% of the minority class B as training data, and the same number of examples is kept for the other classes. The obtained training set will present the same number of training nodes for each class (3).



(a) Decisions on monetary layer

(b) Decisions on social layer

Figure 5: The distribution of the generated labels encoding the user migration decision, in the two layers a) monetary and b) social respectively.

Figure 5b for social interactions. We can observe how the distribution of labels is not balanced: in the monetary layer, there is a slight skew in the number of *co-active* users, and the minority class is composed of *migrant* users. Whereas the social layer is severely imbalanced as the majority of users become *inactive* after the fork event.

Predicting user migration. We now investigate whether graph neural networks are a suitable method for user migration prediction (RQ1) by applying the methodology presented in Section 5 on our dataset. We first train our models for prediction on the graph $G^{m}_{t_{Fork}}$ representing monetary interactions before the fork. In Table 1 we show the obtained results on the monetary layer. The trained GNN model surpasses both Baseline classifiers by a significant margin, both in terms of accuracy and weighted F1. These results indicate that the model can learn by exploiting only the topology derived from monetary interactions. We then perform the prediction task

Model	Train accuracy	Test accuracy	Train weighted F1	Test weighted F1
Baseline most-frequent	0.346 ± 0.001	0.336 ± 0.003	0.178 ± 0.0011	0.169 ± 0.002
Baseline uniform	0.249 ± 0.001	0.249 ± 0.004	0.253 ± 0.001	0.2515 ± 0.001
Best model	**0.426 ± 0.003**	**0.424 ± 0.006**	**0.381 ± 0.002**	**0.379 ± 0.003**

Table 1: Accuracy and weighted F1 (mean and standard deviation over 3 random seeds [32]) obtained by the Baseline classifiers and the best GNN model on the monetary graph $G^m_{t_{Fork}}$.

Model	Train accuracy	Test accuracy	Train weighted F1	Test weighted F1
Baseline most-frequent	**0.770 ± 0.001**	**0.770 ± 0.002**	**0.671 ± 0.001**	**0.670 ± 0.004**
Baseline uniform	0.250 ± 0.001	0.250 ± 0.001	0.319 ± 0.001	0.318 ± 0.001
Best model	**0.770 ± 0.001**	**0.770 ± 0.002**	**0.671 ± 0.001**	**0.670 ± 0.004**

Table 2: Accuracy and weighted F1 (mean and standard deviation over 3 random seeds [32]) obtained by the Baseline classifiers and the best GNN model on the social graph $G^s_{t_{Fork}}$.

Model	Train accuracy	Test accuracy	Train weighted F1	Test weighted F1
Best model imbalanced	0.426 ± 0.003	0.424 ± 0.006	0.381 ± 0.002	0.379 ± 0.003
Best model balanced	**0.427 ± 0.001**	**0.424 ± 0.001**	**0.386 ± 0.007**	**0.382 ± 0.007**

Table 3: Accuracy and weighted F1 (mean and standard deviation over 3 random seeds [32]) obtained by the best GNN model trained on the imbalanced training set and the best model trained on the balanced training set, on the monetary graph $G^m_{t_{Fork}}$.

Model	Train accuracy	Test accuracy	Train weighted F1	Test weighted F1
Best model imbalanced	**0.770 ± 0.001**	**0.770 ± 0.002**	**0.671 ± 0.001**	0.670 ± 0.004
Best model balanced	0.403 ± 0.002	0.725 ± 0.006	0.359 ± 0.003	**0.788 ± 0.004**

Table 4: Accuracy and weighted F1 (mean and standard deviation over 3 random seeds [32]) obtained by the best GNN model trained on the imbalanced training set and the best model trained on the balanced training set, on the social graph $G^s_{t_{Fork}}$.

on the *social* layer $G^s_{t_{Fork}}$ that represents social interactions before the fork. In Table 2, we show the evaluation results. The gap between the trained model and the baseline classifiers is not as large. The most frequent baseline classifier (the one that predicts the most frequent class observed in training) obtains an accuracy score similar to the best GNN model, while the Uniform Baseline lags severely behind. When we consider the weighted F1 scores we observe a similar trend: the Baseline performs similarly to the GNN model. As the accuracy scores coincide with the percentage of the most frequent class for both the baseline and the GNN model, we investigated the predictive behavior of the best model; we discovered that after a few epochs, begins to predict always the same class, the most frequent class in the training set. In the case of a severely imbalanced dataset, the graph neural network model struggles in the prediction of less frequent classes, it acts similarly to the baseline classifier. In general, we can say that the GNN has learned from the input data, making it a suitable model for solving the problem on the monetary layer. While prediction in more imbalanced settings, like in the social graph requires addressing the class imbalance problem.

Dealing with class imbalance. In the following, we now analyze how we can deal with class imbalance (RQ2) by applying the methodology presented in Section 5. We compare the best GNN model obtained on the two layers and compare the best model using the balancing approach.

We first make the comparison on the monetary layer: in Table 3 we report the evaluation metrics for both the approaches. The models that learned from the balanced graph and those that learned from the original graph have roughly the same performance. This is expected as the target variable is not very imbalanced in the monetary layer. Moreover, the fact that the performances are similar is an additional positive factor: the model trained on the balanced train set, actually learns from fewer examples, and yet does not lose in performance. In fact, we actually observe the opposite effect, with slight improvements overall. We then present the evaluation results obtained on the social layer: where target labels are more imbalanced, In Table 4. We can see the impact of the balancing technique we proposed. First, we verified that the model that learns from the balanced set actually returns as a prediction not just the most frequent class, but other classes as well. The model that learns on the balanced dataset exhibits a drop in both accuracy and weighted F1 over the training sets, however, the performance over the test sets is high, especially in terms of weighted F1.

Overall the balancing technique constitutes an improvement for the GNN model. In more balanced datasets, we are able to obtain good performance but training on fewer data, while on the more imbalanced datasets, it improves the learning phase for the model, which learns to better predict minority classes.

7 CONCLUSION

In this work, we addressed the problem of user migration prediction, focusing on some understudied aspects like the effectiveness of graph neural networks as a prediction method, as well as addressing the class imbalanced learning problem typically observed in classification tasks, and also in blockchain-based systems. Our findings show that graph neural networks are an effective method to predict user migration in blockchain-based online social networks as our methodology, modeling user interaction data into multilayer temporal graphs suitable for graph neural network modeling, leads to a model able to leverage the graph of monetary interactions but struggling on the severely imbalanced social layer. However, after applying our proposed data-level balancing approach that produces a more balanced training set, graph neural networks show increased predictive power even on severely imbalanced data. The obtained performances are an important result since they highlight the predictive power of graph structure, without the need for manual feature engineering. Moreover, the trained models perform well even with a lack of node features, something that is typical of blockchain-based systems. Future research will look into the applications, as user migration is not limited to online social networks: leaving for another social, leaving for another crypto or other Dapp. The proposed methodology could lead to significant improvement in other prediction tasks typical of online social networks or blockchain-based systems such as fraud detection, and bot detection. Future additional works could focus on developing other balancing strategies.

ACKNOWLEDGMENTS

This work was partially supported by project SERICS (PE00000014) under the MUR NRRP funded by the EU-NextGenerationEU.

REFERENCES

[1] Cheick Tidiane Ba, Andrea Michienzi, Barbara Guidi, Matteo Zignani, Laura Ricci, and Sabrina Gaito. 2022. Fork-based user migration in Blockchain Online Social Media. In *Proceedings of the 14th ACM conference on web science.*

[2] Cheick Tidiane Ba, Matteo Zignani, and Sabrina Gaito. 2022. The role of groups in a user migration across blockchain-based online social media. In *2022 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. IEEE, 291–296.

[3] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of machine learning research* 13, 2 (2012).

[4] Lucio La Cava, Luca Maria Aiello, and Andrea Tagarelli. 2023. Get Out of the Nest! Drivers of Social Influence in the #TwitterMigration to Mastodon. arXiv:2305.19056 [cs.CY]

[5] Cai Davies, James Ashford, Luis Espinosa-Anke, Alun Preece, Liam Turner, Roger Whitaker, Mudhakar Srivatsa, and Diane Felmlee. 2021. Multi-scale user migration on Reddit. AAAI.

[6] Steemit developer documentation. 2021. Broadcast Ops. https://developers.steem.io/apidefinitions/broadcast-ops

[7] Hive Developer Documentation. 2021. API Docs - API Definitions. https://developers.hive.io/apidefinitions/

[8] Pierluigi Freni, Enrico Ferro, and G Ceci. 2020. Fixing social media with the blockchain. In *Proceedings of the 6th EAI international conference on smart objects and technologies for social good.* 175–180.

[9] Alessia Galdeman, Matteo Zignani, and Sabrina Gaito. 2022. Disentangling the Growth of Blockchain-based Networks by Graph Evolution Rule Mining. In *2022 IEEE 9th International Conference on Data Science and Advanced Analytics (DSAA)*. IEEE, 1–10.

[10] Alessia Galdeman, Matteo Zignani, and Sabrina Gaito. 2023 (Accepted). User migration across web3 online social networks: behaviors and influence of hubs. In *In Proceedings of IEEE International Conference on Communications*.

[11] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *International conference on machine learning*. PMLR, 1263–1272.

[12] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[13] Barbara Guidi. 2020. When Blockchain meets Online Social Networks. *Pervasive and Mobile Computing* 62 (2020), 101131.

[14] Barbara Guidi, Andrea Michienzi, and Laura Ricci. 2020. A Graph-Based Socioeconomic Analysis of Steemit. *IEEE Transactions on Computational Social Systems* PP (12 2020), 1–12. https://doi.org/10.1109/TCSS.2020.3042745

[15] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[16] Petter Holme and Jari Saramäki. 2012. Temporal networks. *Physics Reports* 519, 3 (2012), 97–125. Temporal Networks.

[17] Harsurinder Kaur, Husanbir Singh Pannu, and Avleen Kaur Malhi. 2019. A systematic review on imbalanced data challenges in machine learning: Applications and solutions. *ACM Computing Surveys (CSUR)* 52, 4 (2019), 1–36.

[18] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).

[19] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[20] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *International Conference on Learning Representations*. https://openreview.net/forum?id=SJU4ayYgl

[21] Shamanth Kumar, Reza Zafarani, and Huan Liu. 2011. Understanding user migration patterns in social media. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 25. 1204–1209.

[22] Liqun Liu, Weihan Zhang, and Cunqi Han. 2021. A survey for the application of blockchain technology in the media. *Peer-to-Peer Networking and Applications* 14, 5 (2021), 3143–3165.

[23] Yihong Ma, Yijun Tian, Nuno Moniz, and Nitesh V Chawla. 2023. Class-Imbalanced Learning on Graphs: A Survey. *arXiv preprint arXiv:2304.04300* (2023).

[24] Minds. [n. d.]. Minds. https://www.minds.com/

[25] Edward Newell, David Jurgens, Haji Saleem, Hardik Vala, Jad Sassine, Caitrin Armstrong, and Derek Ruths. 2016. User migration in online social networks: A case study on reddit during a period of community unrest. In *Proceedings of the International AAAI Conference on Web and Social Media*, Vol. 10. 279–288.

[26] Sapien. [n. d.]. Sapien. https://www.sapien.network/

[27] Malith Senaweera, Ruwanmalee Dissanayake, Nuwini Chamindi, Anupa Shyamalal, Charith Elvitigala, Sameera Horawalavithana, Primal Wijesekara, Kasun Gunawardana, Manjusri Wickramasinghe, and Chamath Keppitiyagama. 2018. A weighted network analysis of user migrations in a social network. In *2018 18th International Conference on Advances in ICT for Emerging Regions (ICTer)*. IEEE, 357–362.

[28] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, Yoshua Bengio, et al. 2017. Graph attention networks. *stat* 1050, 20 (2017), 10–48550.

[29] Shermin Voshmgir. 2020. *Token economy: How the Web3 reinvents the internet.* Vol. 2. Token Kitchen.

[30] We Are Social UK. 2023. Digital 2023. https://wearesocial.com/uk/blog/2023/01/digital-2023/

[31] Li Yang and Abdallah Shami. 2020. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* 415 (2020), 295–316.

[32] Jiaxuan You, Tianyu Du, and Jure Leskovec. 2022. ROLAND: graph learning framework for dynamic graphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2358–2366.

[33] Jiaxuan You, Rex Ying, and Jure Leskovec. 2019. Position-aware graph neural networks. In *International conference on machine learning*. PMLR, 7134–7143.

[34] Muhan Zhang and Yixin Chen. 2018. Link prediction based on graph neural networks. *Advances in neural information processing systems* 31 (2018).

[35] Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhi Yu, and Can Wang. 2019. Hierarchical Graph Pooling with Structure Learning. *ArXiv* abs/1911.05954 (2019).

[36] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. 2021. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *Proceedings of the 14th ACM international conference on web search and data mining*. 833–841.

[37] Haris Bin Zia, Jiahui He, Aravindh Raman, Ignacio Castro, Nishanth Sastry, and Gareth Tyson. 2023. Flocking to mastodon: Tracking the great twitter migration. *arXiv preprint arXiv:2302.14294* (2023).