Jurnal Kejuruteraan 38(5) 2026: 2625-2659
https://doi.org/10.17576/jkukm-2026-38(5)-40

Tackling Imbalanced Data for Improved Machine Learning-Based Routing in Delay-Tolerant Networks

Usama Shaker Hachim[a], Asma Abu-Samah[a*], El Arbi Abdellaoui Alaoui[b], Nor Fadzilah Abdullah[a]
Amine Salah[b], Anand Nayyar[c] & Ziad Qais AlAbbasi[d]

[a]*Department of Electrical, Electronic and Systems Engineering, Faculty of Engineering and Built Environment, Universiti Kebangsaan Malaysia, 43600 Bangi Selangor, Malaysia*
[b]*Moulay Ismail University of Meknes*
[c]*Duy Tân University*
[d]*Middle Technical University*

[*]*Corresponding author: asma@ukm.edu.my*

Received 12 March 2026, Received in revised form 21 May 2026
Accepted 21 June 2026, Available online 30 August 2026

ABSTRACT

A Delay Tolerant Network (DTN) is designed for intermittent connectivity and long delays environments, using a store-and-forward approach to transmit data. Machine Learning in Delay Tolerant Networks (DTNs): Prediction of link availability, routing optimization and adaptation to network dynamics. Machine learning models improve the efficiency of data transfer, thus reducing latency and enhancing decision making. This improves the robustness of the communication in the applications of remote sensing and disaster recovery. In addition, machine learning has shown promising application to design intelligent routing strategies for DTNs, which can improve communication efficiency in highly dynamic environments. The data unbalance due to different message generation rates and network connectivity has a negative impact on the performance of ML-based DTN routing algorithms. This results in biased predictions and sub-optimal routing decisions. The issue of imbalanced data in machine learning based DTN routing is examined in this study using four common protocols: Epidemic, Spray and Wait, Prophet, and MaxProp. We compare different oversampling strategies with six classifiers: Random Forest, Extra Trees, XGBoost, Bagging Classifier, Decision Tree and K-Nearest Neighbors. We discuss different data pre-processing techniques, algorithmic approaches and pertinent evaluation measures. Our analysis finds the best approaches for handling data imbalance problems. Among the methods considered, SMOTENN with Extra Trees classifier consistently obtained the highest accuracy in all four procedures. The application of these tactics resulted in an impressive improvement in model performance - accuracy increased by 13.58% (from 81% to 92%) and thus improving the robustness and fairness of ML-based DTN routing protocols.

Keywords: Delay-Tolerant Networking, Machine Learning, Imbalanced Data, Routing Protocols Performance Evaluation Metrics, Data Pre-processing Techniques, Robustness

INTRODUCTION

Delay Tolerant Networks (DTNs) operate under the challenging circumstances of intermittent or delayed communication, which often occurs in remote or rural areas, and disaster-stricken regions where traditional internet infrastructure is not available (Le 2021; Sun et al. 2013; Li et al. 2017; Pereira et al. 2012). In such environments, conventional routing protocols, which assume constant connectivity between nodes, are ineffective. Therefore, the development of robust and efficient routing protocols that can handle such conditions is crucial. The emergence of the Internet of Things (IoT) has brought about the interconnection of numerous physical devices and sensors (Galán-Jiménez et al. 2019), which collect and exchange data. However, these IoT devices are often deployed in harsh or remote environments, where traditional communication networks are unreliable or entirely absent. In these scenarios, DTNs serve as a viable communication model, facilitating device communication despite the lack of a continuous network infrastructure (Seyhan et al. 2021; Mao et al. 2019; Abdellaoui Alaoui, Zekkori & Agoujil 2019).

The main objective and contribution of this paper are focused on tackling the prevalent problem of imbalanced data within the realm of DTNs, thereby enhancing routing efficiency, particularly in the context of the IoT. Imbalanced data, which is frequently encountered in machine learning, can bias the learning process and result in suboptimal routing decisions. To address this issue, several oversampling strategies are proposed in conjunction with various classifiers. In this paper, balancing this data is shown to improve the performance of machine learning-based routing in DTNs.

This work builds upon earlier research in which machine learning techniques were deployed for intelligent routing in DTNs (Abdellaoui Alaoui et al. 2023). The particular novelty of the present effort lies in addressing the issue of data imbalance, a problem not tackled directly in preceding studies. By addressing this previously overlooked challenge, a significant improvement in the performance of the machine learning-based routing strategies developed in earlier work is achieved.

DTNs play a crucial role in establishing reliable communication under the most challenging conditions, including space missions, disaster-stricken areas, and remote locations. The routing strategies of DTNs need to be optimized. Recently, a few ML-based routing algorithms have emerged as a potent solution for making informed decisions in enhancing DTN performance. Notable examples include the works of (Bhavani, Ramana & Chakravarthy 2023; Harkavy & Net 2020).

However, due to the spread of imbalanced data through each sector, such as smart cities, network domains, and security, it finally affects the performance of these algorithms. When trained on such skewed data, the resulting model tends to favor the majority class, leading to poor bundle classification, as shown in FIGURE 1. This bias in classification can significantly degrade routing decisions and, consequently, overall DTN performance. For example, poor classification of bundles due to imbalance in a DTN environment could result in incorrect predictions of whether a bundle will arrive at its destination. This may further translate to wasted resources in the network, such as processing power and memory, increasing the time taken for successful transmission. The most important challenge in this regard is that the majority of prior research has completely disregarded the imbalanced nature of DTN traffic data; This study aims to find a solution to the challenge of imbalanced data by investigating several data balancing methods, examining their impact on bundle classification accuracy within DTNs, and assessing the effectiveness of different oversampling strategies and their combinations with various classifiers.

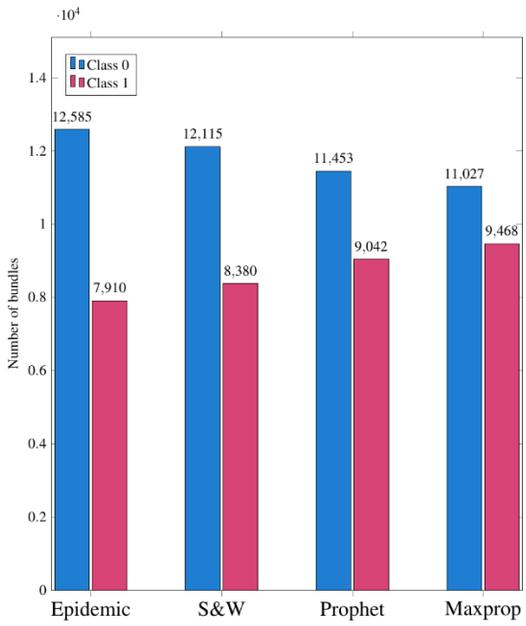


FIGURE 1. Distribution of bundles in each class for DTN protocols

This contributes to a critical knowledge gap in the field and has the potential to support the development of more robust, efficient, and effective routing strategies for DTNs. As such, the insights gained could play a vital role in today's increasingly interconnected world.

Delay-Tolerant Networks (DTNs) rely heavily on the efficient routing of bundles to provide communication reliability under demanding and intermittent connectivity conditions. In terms of boosting the performance of DTNs, machine learning-based routing algorithms have proven to be effective. The presence of unbalanced data, which happens when the distribution of classes within the training dataset is severely skewed (as illustrated in FIGURE 1), is a key obstacle that inhibits the performance of various algorithms. This imbalance can result in biased classifier models that favor the majority class and poor generalization performance on the minority class, which can have a substantial effect on routing decisions and network performance as a whole. Class imbalance is a common challenge in classification problems. We speak of an unbalanced dataset when the ratio of the observations of a class to the total number of observations is very low. This notion of class imbalance is relatively frequent in several sectors such as the smart city sector, the network sector or the security sector and it becomes problematic when left unaddressed. By adopting a naive approach to classification, i.e. one that does not take into account this class imbalance, there is a strong risk of biasing the model. Thus, in the case of bundle classification in the context of intelligent routing,

the risk of a naive classification approach is to predict potentially non-arriving bundles as arriving or the opposite. The goal of this work is to investigate how data balancing tools may increase the accuracy of bundle classification and DTN routing protocols utilizing machine learning approaches. In light of this, the following research questions (RQs) are proposed:

RQ1: Are data balancing methods suitable for bundles classification?

RQ2: Which data balancing methods is better suited for bundles classification?

RQ3: How effective are data balancing methods compared to without balancing data for bundles classification?

In this paper, therefore, an ML approach for routing in DTNs will be proposed to handle such imbalance. This paper reviews general challenges in routing in DTNs and discusses weaknesses of earlier approaches (Abdellaoui Alaoui et al. 2023). It then proposes a new approach that uses a combination of oversampling and under sampling techniques to balance the data and improve the accuracy of the ML model. The proposed approach has been evaluated on real-world datasets, and the results show that it outperforms existing approaches in terms of accuracy and efficiency. It is concluded that the proposed approach improves routing performance in DTNs to allow efficient data delivery against challenging environments.

The contribution of this article to the literature under study is essential, notable, and valuable in the following ways: it detects and investigates the imbalance problem in ML-based DTN routing, showing its impact on routing algorithm performance; it researches and assesses methods that address unbalanced data to improve bundle classification and routing performance; it provides a comprehensive comparison of data pre-processing methods, algorithmic approaches, and performance evaluation measures for ML-based DTN routing; and through rigorous simulations, the proposed approach has been empirically validated, showing how solving the imbalanced data problem significantly improves bundle classification and overall routing performance.

This article is organized into four main sections in addition to this introduction section. Section 2 presents a general overview of the state of the art in DTN routing protocols, along with a review of research related to intelligent routing. Section 3 describes the methodology adopted to address the problem. Section 4 presents the experiments conducted in this study, along with a discussion of the obtained results and a comparison with the state of the art. The paper concludes with future scope in Section 5.

RELATED WORK AND LIMITATIONS

This section reviews the relevant literature and identifies existing limitations that provide context for this work. The section is divided into four subsections.

Firstly, the application of Artificial Intelligence (AI) and Machine Learning (ML) techniques in the context of Delay-Tolerant Networks (DTNs) is explored. This includes examining the application of various learning algorithms, their effectiveness in predicting and improving routing, and the unique challenges posed by DTNs.

The second subsection delves into the domain of DTN routing, highlighting several key protocols such as Epidemic, Prophet, Spray and Wait, MaxProp, and others. We analyze how these protocols work, their benefits, and their limitations.

Thirdly, attention is turned to imbalanced data processing methods. Given that the class imbalance problem can severely impact the effectiveness of ML applications, the common techniques applied for dealing with this issue are examined, including under sampling, oversampling, and synthetic data generation.

Finally, in the fourth subsection, we critically assess the limitations of the discussed techniques in the context of DTNs. Gaps in the current state of research are identified, and areas requiring further exploration are highlighted. This analysis sets the scene for the novel contributions of this paper by proposing a method addressing the identified limitations.

AI & MACHINE LEARNING FOR DTN

The studies reviewed in this section approach DTN routing from different angles. Rather than listing them in chronological order, they are grouped here by what each set of works is trying to achieve, whether that is classifying bundles, learning routing policies on the fly, predicting node behavior, combining several models, or solving challenges specific to vehicular environments. Each group is followed by a short discussion of what these works collectively bring to the table and where they fall short in relation to the problem addressed in this paper.

Group 1: ML for Bundle and Node Classification
Abdellaoui Alaoui et al. (2022) gives new insights into designing intelligent routing using ML methodologies. Their approach encompasses a complex data collection process for bundle classification, integration with DTN protocols to form an intelligent model, and using ML

algorithms for predictive bundle classification. The research also goes further to investigate the interpretability aspect of the ML models and further proposes the use of game theory as a tool for explaining the decision-making process of each ML model.

Kumar et al. (2023) applied Artificial Neural Network (ANN) and Convolutional Neural Network (CNN) models to optimize the alpha, beta, and gamma parameters of the Probabilistic Routing Protocol for Intermittently Connected Networks (PROPHET). Both models were trained and tested using data from the Opportunistic Network Environment Simulator (ONE). The results showed that the ANN model outperformed CNN, achieving a classification accuracy of 0.99. In DTN-Internet routing, various ML algorithms have been studied to optimize network performance. One such study, by Bajpai & Chauhan (2022), explores different classifiers, namely Naïve Bayes, k-NN, XGBoost, AdaBoost, and Random Forest (Bagging), for their effectiveness in optimizing delay-tolerant routing. The authors also discussed a range of performance metrics such as the Jaccard similarity score, Zero-one loss, F-1 score, and Hamming loss to measure accuracy and losses associated with these ML techniques.

The Optimal Routing with Node Classification (ORNC) approach, proposed by Tripathi et al. (2021) corresponds to developing an ML algorithmic solution based on node characteristics to select the best routing of packets from source to destination node, where packet routing between two different nodes belonging to either to a MANET, DTN or Bluetooth network. As classification machine learning algorithms, K-Nearest Neighbor (K-NN), Support Vector Machine (SVM), Multi-Layer Feature (MLF) and NN were used. There is a classification for this which is useful for deciding whether or not use collocation and therefore which routing strategy is optimal. By applying the node class and trust factor, a fitness value is calculated which quantifies the node's suitability to perform the role of an intermediate. The neural network model achieves an accuracy of 95.21%, outperforming the other three machine learning-based approaches in classifying networks as MANETs, DTNs, and Bluetooth. This type of increased accuracy suggests that the most efficient choice of hops is selected.

A related study investigated the use of machine learning classifiers to improve routing efficiency and reliability in DTNs by predicting nodes with a higher likelihood of successful message delivery. It was built upon the history of delivery rates, which had then been tried with Institute of Operating Systems and Computer Networks Delay Tolerant Networking (IBR-DTN) framework. This was done using (Doering et al. 2008) and by location traces from Common Open Research Emulator (CORE) (Ahrenholz, Goff & Adamson 2011). Study results indicate that classification-based routing is superior to the traditional routing procedures, to enhance the message delivery and decrease latency in DTNs. The study also examines various components of the software architecture, including data pre-processing pipelines and prediction mechanisms, demonstrating that routing processes can be enhanced through machine learning algorithms. The authors subsequently evaluated the proposed classification-based routing method, and found it outperforms other methods (Dudukovich, Clark & Papachristou 2019) in DTNs.

These works share a common thread, they frame DTN routing as a classification problem, where the goal is to decide, based on available features, whether a bundle will reach its destination or which node is best suited to carry it forward (Abdellaoui Alaoui et al. 2022; Bajpai & Chauhan 2022; Kumar et al. 2023; Tripathi et al. 2021; Dudukovich, Clark & Papachristou 2019).

The classifiers used vary from decision trees and k-NN to neural networks and ensemble methods, and the results are generally promising in controlled simulation environments. What none of these studies account for, however, is the distribution of the training data itself. When one class say, bundles that failed to arrive significantly outnumbers the other, standard classifiers tend to learn the majority pattern and overlook the minority one. This is precisely the gap the present work addresses.

Group 2: RL for Routing Decisions Dalal & Khari (2022) proposed the Q-value Routing Protocol (QRP), which applies reinforcement-based deep Q-learning to determine whether a node should forward a packet or hold it, based on the estimated delivery value of that decision.

Kandhoul & Dhurandher (2022) proposed a state-of-the-art proactive security approach for deep reinforcement learning in Opportunistic IoT networks. The authors indicated the inability of traditional security solutions to address unique challenges in these types of networks, characterized by intermittent connectivity, resource limitation, and dynamic topologies. To this end, they proposed a Deep Q-learning methodology whereby optimal routing policies are learned by the used neural network during real runtime. The authors demonstrate through simulations that are performed on a real-world OppIoT network how the considered approach is more effective against conventional routing protocols both from efficiency and security standpoints.

Sharma et al. (2020) proposed RL-Proph, a reinforcement learning-based routing protocol designed for opportunistic IoT networks. The authors optimize the routing mechanism based on dynamic programming in nature and reinforcement learning with these optimizations amplifying the probability of successful delivery over the network. As a fully automated solution, Reinforcement

Learning PRoPHET (RL-Proph) outperforms existing ML-driven and context aware routing protocols in many performance metrics. The work is comprehensive in the sense that it includes extensive simulations, comparisons with existing protocols, and thorough investigation of the structure and deployment of the proposed protocol.

Harkavy & Net (2020) proposed an autonomous Reinforcement Learning approach for buffer management in DTN nodes. According to them, this novel method enables DTN nodes to avoid memory overflow using the automated management tasks of the buffer. The paper details three possible decisions an autonomous agent can make in response to overflow risk: throttling the client, acquiring additional resources from the Deep Space Network, or intentionally dropping packets when the buffer approaches capacity. The paper also argues that Reinforcement Learning is actually an ideal choice to be used, in light of the huge amount of states and variables to be monitored, and the impracticable interference from human intervention over immense distances between deep space spacecraft and Earth.

Dudukovich, Hylton & Papachristou (2017) explored the conceptual design and architectural blueprint of a machine learning infused router developed solely for delay tolerant space networks is investigated. Using reinforcement learning and Bayesian learning methodologies they supplement the decisions of the widely recognized Contact Graph Routing algorithm. The paper introduces the principles of Contact Graph Routing, Qrouting and Naïve Bayes classification entirely. Additionally, the authors discuss the design of a cross layer feedback framework targeted toward DTN protocols.

Mochizuki et al. (2019) introduced a deep reinforcement learning-based delay tolerance mobile data offloading method. In particular, deep reinforcement learning is used to balance the load by injecting Mobile Data Offloading Protocol (MDOP), taking into account the content delay tolerance. The Reinforcement Learning (RL) server determines the bandwidth priority for each user equipment based on content delay tolerance, available resources, and current eNB load, while the MDOP server handles load balancing and bandwidth control. The authors validated their approach through an initial evaluation to confirm that the proposed method can assign appropriate QoS levels based on delay tolerance.

Sharma et al. (2018) proposed Poisson's Probability Based Predictive QRouting, a routing scheme that integrates QRouting's decision-making capability with the Poisson's distribution. The routing system presented is appropriate for dynamic and unpredictable network environments and node mobility, as well as adapting to the current environment.

Rather than learning from a labelled dataset, these works let the routing agent learn by interacting with the network environment, receiving rewards for good forwarding decisions and adjusting its policy accordingly (Dalal & Khari 2022; Kandhoul & Dhurandher 2022; Sharma et al. 2020; Harkavy & Net 2020; Dudukovich, Hylton & Papachristou 2017; Mochizuki et al. 2019; Sharma et al. 2018). Reinforcement learning fits DTN well in principle, since the network state changes constantly and a fixed rule set often falls short. In practice, though, the training process in these studies still relies on interaction data that reflects whatever imbalance exists in the network, episodes where a bundle arrives are far less frequent than those where it does not, and this skew is rarely discussed, let alone corrected.

Group 3: ANN for Prediction and Trust Motivated by the PROPHET routing algorithm, Sharma et al. (2016) designed Machine Learning Prophet (MLProph), which is routing protocol based on machine learning. Recently the MLProph protocol has been proposed to enhance DTN routing performance by taking into account a number of input parameters, including popularity parameter, buffer memory size, node energy, hop count and the packet delivery success rate. If these factors are added, the protocol's routing decision would adjust based on conditions at the time that the network is experiencing along with those at the nodes. For MLProph protocol neural networks are used as a fundamental machine learning model, the protocol may learn from input parameters and predict a successful or unsuccessful packet delivery. The neural network is trained to produce two output probabilities: the probability that the delivery will succeed and the probability that it will fail. The MLProph protocol then compares these probabilities and can more effectively make routing decisions and select the best next-hop nodes to which to transmit packets.

A multi-copy routing protocol for urban bus transit systems was proposed by Segundo, e Silva & Farines (2016). Each vertex in the multi-graph is identified using an artificial neural network (ANN) inspired following node (bus) contact predictor as part of the protocol. In this way, the routing protocol makes informed decisions about the contacts between nodes (buses) in the urban transit network, given their predicted contact states. In such environments where connectivity can be sporadic and encounters between nodes may be predictable based upon bus schedules, it is possible to improve the routing efficiency of the protocol by utilizing ANN predictions effectively. Built upon this, Singh, Juyal & Saggar (2017) developed the Trust Based Intelligent Routing (TBIR) protocol that employs artificial neural networks (ANNs) to enhance DTN routing performance by using trust values at the routing decisions. ANN-based trust values calculated

and learned can be used by the protocol at nodes of networks to calculate such values and can be helpful in smarter routing decisions. The function of trust within the protocol TBIR has its basis in three key metrics: i) The interval between the most recent connection and the last connect, ii) the frequency of network traffic and, iii) the total time of contact between nodes.

Karami & Derakhshanfard (2020) proposed Remaining Path Routing for Time to Destination (RPRTD), a routing protocol that addresses the problem of the remaining time to meet the destination node, RTD, in DTNs. Unlike traditional routing protocols, RPRTD aims to facilitate faster connections between nodes with smaller RTD values to the destination node. This approach can potentially improve the overall routing efficiency and reduce delays in packet deliveries. The RPRTD protocol employs artificial neural networks as the underlying foundation to calculate the RTD values for each node. By utilizing ANNs, the protocol can effectively learn from past encounters and predict the remaining time for a node to meet the destination node in the future.

These protocols use neural networks not to classify routing outcomes directly, but to estimate quantities that feed into routing decisions, how long before two nodes meet again, how trustworthy a relay is, or how likely a delivery is to succeed (Sharma et al. 2016; Segundo, e Silva & Farines 2016; Singh, Juyal & Saggar 2017; Karami & Derakhshanfard 2020). The idea is sound, and the reported results show meaningful improvements over traditional protocols. The limitation is that the networks are trained on historical encounter data, which in real DTN deployments tends to be heavily skewed successful contacts are rarer than missed ones. Training on this data without correction quietly biases the predictions, a problem none of these works acknowledges.

Group 4: Hybrid and Ensemble Approaches Vashishth, Chhabra & Sharma (2019) presented a machine learning (ML) based method called Cascade Machine Learning (CAML) for improving routing for Opportunistic Internet of Things (OppIoTs). The basis of the CAML is for the cascade learning, a technique which combines several learning models to get better results. The authors also extend the ideas from the preceding MLProph routing protocol whose basic idea of routing decisions also uses machine learning in OppIoTs. As in their approach, they demonstrated that the overall performance can be significantly increased through adding a Logistic Regression classifier in front of a Multi-Layer Perceptron (MLP) Neural Network classifier as an input cascade.

Garg, Dixit & Sethi (2021) introduced a new routing protocol called the Machine Learning-enabled Fresh (ML-Fresh) Routing Protocol, which is specially designed for Opportunistic Networks. The unsupervised learning benefits used in Cluster Opportunistic Networks are presented in detail with the analysis regarding its potential to enhance conventional schemes. Also, details are given on Fresh Protocol, which is argued as one of the excellent routing protocols among many alternatives available within Opportunistic Networks. There is also the proposal to integrate machine learning within Fresh Protocol, in order to improve performances of it in terms of efficiency and effectiveness. The authors explained the methodology and the strategy of operation of this proposal, the ML-Fresh Protocol, together with validation of efficacy in simulating and presenting outcomes.

Banyal et al. (2021) proposed a new scheme called Hierarchical Learning-based Sectionalized Routing (HiLSeR). The scheme organizes the network topology based on node characteristics by exploiting one principle of intelligent transmission grouping. In HiLSeR, the mechanism adopted is that of hierarchical learning where a soft clustering paradigm is applied on multi-dimensional data for both topology sectionalization and routing decision-making. Performance evaluation of HiLSeR is carried out against state-of-the-art protocols, namely, Reinforcement Learning PRoPHET (RL-Proph), Gaussian Mixture Model Routing (GMMR), K-Nearest Neighbor based Routing (KNNR), and Firefly PRoPHET, through ONE-based simulations. The paper stresses the need for intelligent routing in opportunistic scenarios that present intermittent connectivity and are very resource-constrained, showing a strong requirement to develop efficient routing strategies.

Kumar et al. (2022) proposed a new routing strategy for Opp-IoT networks, based on machine learning methods, in order to increase routing efficiency. Their scheme deploys a distributed probability density-centric approach for the optimization of an array of objectives that include energy consumption, buffer utilization, and the rate of message delivery. Also, they introduce the Fuzzy Cognitive Maps (FCM)-based clustering model to outline clearly the network topology for routing decisions optimization. This paper carries out not only extensive analysis of the proposed scheme against various performance metrics but also presents its contribution in the area of sustainable computing.

These works go a step further by combining multiple learning models cascading classifiers, layering clustering with routing decisions, or integrating unsupervised and supervised stages (Vashishth, Chhabra & Sharma 2019; Garg, Dixit & Sethi 2021; Banyal et al. 2021; Kumar et al. 2022). The added complexity does yield better results in most reported experiments. What is harder to overlook, however, is that stacking more models on top of imbalanced data does not fix the underlying problem it can even amplify it, since each stage inherits the bias from the one before it.

None of these studies take a step back to examine whether the data feeding into their pipelines is balanced to begin with.

Group 5: Vehicular and Application-specific DTN

Singh et al. (2021) propose an efficient routing scheme for Vehicular Delay-Tolerant Networks (VDTNs) based on different machine learning classifiers, where the classifiers are applied to filter effective vehicular nodes in order to guarantee packet delivery from source to destination. Considered are factors like the location of nodes, duration of contact within the interface range, data transfer rate, buffer capacity, and network topology in deciding the best path. The proposed strategy proves effective in scenarios where traditional networks fall short, such as emergency response, sensor-based applications, intelligent transportation systems, and weather forecasting services.

Liang, Shang & Wang (2021) explored the application of ML in developing a DTN routing protocol for vehicular ad hoc networks. Their work examined several ML-based approaches for network virtualization and resource management, while acknowledging that the full potential of ML in vehicular DTN routing remains underexplored. Simulation results demonstrated that ML techniques can contribute to developing more effective and robust routing protocols for vehicular ad hoc networks.

Chourasia, Pandey & Kumar (2021) developed an approach to improve VDTN efficiency in information broadcasting. The spread of messages has been quantified by three main metrics, delay, overhead ratio, and delivery ratio. In this regard, the probabilistic forwarding technique has been developed to consider the social features of a network, social strength, trust, friendship and selfishness. In addition, a multiobjective optimization method for balancing the network's parameters has been proposed. The Multi-Objective Particle Swarm Optimization (MOPSO) method proposed here maximizes the gains from balancing the sets of metrics and returns a set of Pareto optimal solutions. A comprehensive AI-based strategy is also included for selecting an optimum solution to satisfy the needs of a particular application situation.

Uchida et al. (2019) designed an Adaptive Array Antenna as a constituent part of a Delay Tolerant Network System for Vehicle-to-Vehicle (V2V) Networks. The system uses image recognition to take care of network communication management. Factors such as whether there are obstacles, and the vehicle speed, are used to adjust the orientation of the antenna using a Kalman Filter method. A machine learning image recognition system is used to identify target vehicles, as well. The study focuses on the image recognition capabilities and antenna orientation controls of the prototype system, analyzing their effectiveness through experimental data.

This group applies machine learning to DTN in more constrained and application-driven settings urban bus networks, road vehicles, social-aware forwarding, and even antenna orientation in V2V communication (Singh et al. 2021; Liang, Shang & Wang 2021; Chourasia, Pandey & Kumar 2021; Uchida et al. 2019). The diversity here is notable, and some of these works introduce creative solutions tailored to their specific scenarios. At the same time, the specialized focus means that fundamental data quality issues, including class imbalance are largely set aside in favour of domain-specific optimization. The challenge of imbalanced training data is present in all of these environments but goes unaddressed across the board.

DTN ROUTING

DTN routing strategies have evolved considerably over the years. The most popular routing algorithms for DTN are based on either flooding (i.e. flood-based strategies) or store-carry forward (i.e. store-carry-forward-based strategies) (Soares, Rodrigues & Farahmand 2014). Routing involves the decision of whether a node should forward a copy of a message to a newly encountered node, based on the estimated probability that this node will eventually reach the destination. Routing refers to the process of directing data bundles through multiple paths toward their final destination across a network capable of tolerating delays. Researchers have developed several routing protocols and methods, some of which are described below:

For Delay-Tolerant Networking (DTN) networks, different routing algorithms have been developed by researchers in the area because traditional routing algorithms created for the Internet (Transmission Control Protocol (TCP)/IP) may not be effective or appropriate in a DTN context. The Epidemic protocol is one of the first and maybe the most well-known (Vahdat, Becker et al. 2000). The replication idea serves as the foundation for this algorithm. It falls under the heading of a strategy based on flooding. Each node in this protocol broadcasts and distributes a copy of its message to newly found nodes. Once again, this recently identified node sends messages in the same manner.

Epidemic routing protocols come in a variety of forms, including P-Q epidemic, encounter epidemic, epidemic with immunity table, and epidemic with TTL. This approach requires a limitless buffer size and furthermore, unlimited energy to provide a high delivery rate. To all intents and purposes, these requirements are challenging to apply. In (Spyropoulos, Psounis & Raghavendra 2005), the proposed DTN protocol is called Spray and Wait. It is a simple but efficient routing protocol that uses a limited

number of replications in the network and is composed of the following phases:

1. The Spray phase: for each new message that has just been generated by a source application, this protocol ensures that L copies of it will be propagated Epidemically in the DTN network so that the message will be present in L different relay nodes.
2. The Wait phase: if a message m has not been able to reach its destination during the Spray phase, each of the relay nodes that have a copy of the message m are no longer allowed to pass the message to any other node in the network except its destination, so the nodes that have the message m must pass it directly to its destination.
3. Delivery predictability is described in the Prophet routing protocol (Lindgren, Doria & Schelen 2004) as an estimate probabilistic indicator, i.e.
4. When two nodes in a network scenario meet, P(node A, node B), the summary vector, which contains delivery predictability values, is swapped at each source node (node a) and each destination node (node b) at each node.
5. All nodes update their own delivery predictability in the summary vector after the swap procedure.
6. If connections between two nodes are very infrequent or do not occur, a low predictability rating is given to the node.
7. If nodes are met at regular intervals, delivery predictability is quite high.
8. According to the transitivity property of the Prophet routing protocol, if node A frequently encounters node B and node B frequently encounters node C, then C is the proper node for A, and as a result, A ranks Cs delivery predictability value highly in the summary vector.

MaxProp is another routing scheme for Delay-Tolerant Networks (DTN). It uses the same ideas as the Prophet protocol, which you can read about in (Burgess et al. 2006) and (Lindgren, Doria & Schelen 2004). MaxProp protocol is a routing protocol designed for DTNs and relies on several mechanisms to optimize the following two routing metrics: message delivery rate and delay in the presence of bandwidth and storage space constraints. A lack of bandwidth results in the establishment of a very short-lived contact, a contact that does not complete all the planned exchanges between two DTNs. In order to overcome this problem, MaxProP defines the order in which messages will be transmitted, referring to the priorities associated with the different messages, knowing that the priority of each message corresponds to the cost associated with its destination. The message with the highest priority will be the first to be transmitted and in the event of congestion in the storage unit of the DTN in question, the message with the lowest priority will be the first to be deleted (Burgess et al. 2006).

Resource Allocation Protocol for Intentional DTN routing (RAPID) (Balasubramanian, Levine & Venkataramani 2007) differs from the rest of the DTN routing protocols in the assumptions it makes about constraints on the network resources, the message scheduling policy and the mobility model. Most routing protocols that were proposed before RAPID are based on one of the following constraints:

1. Storage capacity of the DTN.
2. Bandwidth for the contacts which are established between DTN.

On the other hand, RAPID (Balasubramanian, Levine & Venkataramani 2007) suggests a routing strategy that simultaneously considers these constraints. With the Internet of Things (IoT) increasingly becoming more pervasive, it becomes more important than ever to have efficient and effective routing protocols that are specifically suited to the IoT environment. To address the unique challenges and requirements for IoT-based DTN networks, several routing protocols have been proposed.

The Internet of Bikes (IoB) protocol (Zguira, Rivano & Meddeb 2018), which is a routing protocol for DTNs, developed in the context of IoT. The IoB protocol is based upon the Spray and Wait protocol by using a constrained replication method in order to reduce resource consumption and improve routing efficiency. IoB adapts the Spray & Wait strategy to more suitable behavior of the IoT environments' specific characteristics and constraints. In literature, Borah et al. (2017) presented the Game Theoretic Approach for Context-based routing (GT-ACR) protocol for the DTN networks, specifically for IoT. The routing decisions in this protocol are made using game theory and adapt to the dynamic nature of IoT networks. Identification of a nodes key attributes (e.g. encounter value and distance to its destination) using the GT-ACR protocol enables more informed routing decisions that can increase overall network performance. Mao et al. (2019) designed a Scheduling-PROPHET protocol based on the Prophet protocol, targeting IoT-based DTN networks. This protocol incorporates two scheduling strategies: Message management and message delivery. These strategies collectively enhance the routing efficiency of the DTN in IoT applications by working intelligently to manage and prioritize message transmissions according to delivery probability as well as network conditions.

These protocols form the backbone of the experimental setup in this work. Each generates traffic data where failed deliveries consistently outnumber successful ones, a natural consequence of intermittent connectivity in DTN environments. This imbalance in the data they produce is what makes the problem addressed in this paper worth solving.

IMBALANCED DATA PROCESSING METHODS

Imbalanced data processing refers to a set of techniques designed to handle datasets in which the classes are not equally represented. This disparity often leads to classifiers that perform well on the majority class while producing poor results on the minority class. In literature, various techniques are developed to deal with this problem. From the type of problem, they are grouped into 3 main families.

1. Algorithm-level modification-based methods.
2. Data Weighting-based methods.
3. Cost-sensitive methods.

ALGORITHM-LEVEL MODIFICATION-BASED METHODS

It concerns adapting existing classification learning algorithms to direct learning toward minority data. These methods require knowledge about the classifier meant to be used as well as the domain of application. Algorithm-level modification-based methods emphasize the proposal of new versions of the existing classification algorithms to strengthen their learning capacities to learn the minority class. It is a way of making them less sensitive to the imbalance. Most of the algorithms of this family are based on the Support Vector Machine (SVM) (Tang et al. 2009; Ren et al. 2023; Zhao, Zhong & Zhao 2011; Yu et al. 2018), K-Nearest Neighbors (KNN) (Zeraatkar & Afsari 2021; Ding et al. 2022; Kim 2021; Zhang et al. 2022; Xu et al. 2019; Zang et al. 2017), Artificial Neural Networks (ANN), and Random Forest (RF) (Gu et al. 2022; More & Rana 2022; Xu et al. 2020).

Working at the classifier level rather than the data level is what sets these methods apart, the original training set remains unchanged (Branco, Torgo & Ribeiro 2016). This removes the need to introduce synthetic samples, which can sometimes distort the original data distribution (Ren et al. 2023). The downside is that each adaptation is built around a particular classifier. A modification designed for SVM, for instance, cannot simply be carried over to a tree-based model without rethinking the underlying mechanics (Tang et al. 2009; Zhao, Zhong & Zhao 2011). These methods are a reasonable choice when the classifier is fixed early in the design and the team has enough familiarity with its internals, such as in network intrusion detection or medical diagnosis systems where SVM or ANN architectures are prescribed by domain requirements.

DATA WEIGHTING-BASED METHODS

The Data Weighting-based techniques make it possible to rebalance the distribution of the classes by sampling the data sets. It is a pre-processing step that avoids modifying the learning algorithms. Imbalanced Data Processing refers to methods for handling datasets with unequal representation of the various classes. As a result, there may be problems with a classifier that does well with the majority class but poorly with the minority class. Oversampling the minority class, undersampling the majority class, and employing various metrics for assessment are the standard methods for handling unbalanced data. Sampling techniques are among the simplest and easiest to implement. They are used to deal with the dataset imbalance problem and are also known as dataset preprocessing methods. The primary sampling objective is to restore the balance between the numbers associated with each class. These techniques are independent of the underlying classifier and can be achieved in three ways: undersampling of the majority class.(Xie et al. 2021; Yen & Lee 2009; Dai, Liu & Liu 2022; Zheng et al. 2021), oversampling of the minority class(Xiaolong, Wen & Yanfei 2019; Czarnowski 2022; Sun et al. 2022; Wongvorachan, He & Bulut 2023; García et al. 2020), Or by the combination of the two sampling techniques (Hybrid Methods).

Unlike algorithm-level approaches, sampling methods impose no constraints on the choice of classifier, which makes them easy to slot into existing pipelines (Wongvorachan, He & Bulut 2023; García et al. 2020). Reducing the majority class is straightforward to implement, but it comes at the price of discarding samples that could have contributed to a more informed model (Xie et al. 2021; Yen & Lee 2009). Expanding the minority class through duplication or synthesis carries its own risk the model may end up fitting the repeated samples rather than learning the underlying pattern (Xiaolong, Wen & Yanfei 2019; Czarnowski 2022). Combining both directions softens these extremes but adds tuning overhead. They remain a practical starting point when the dataset is large enough to tolerate some redistribution without distorting the original structure, as seen in network traffic classification and fraud detection tasks where majority-class samples are abundant (García et al. 2020).

COST-SENSITIVE METHODS

These methods concern the simultaneous modification of data and learning algorithms (Lim & Mahinderjit Singh 2020). This approach integrates transformations at the data level by applying costs and changes at the algorithm level by modifying the learning process to accept added costs, steering the classifier toward the minority class. When the penalty for misclassifying a minority instance is known and quantifiable, incorporating it directly into the objective function tends to produce decisions that are more aligned with real-world consequences (Petrides & Verbeke 2022; Guo et al. 2021). In practice, however, those costs are rarely handed to the researcher they have to be estimated, and getting that estimate wrong can quietly push the model in the wrong direction (Lim & Mahinderjit Singh 2020; Ling et al. 2021). These methods fit best in problems where the cost of each type of error is well-understood beforehand, such as fraud detection or network intrusion classification.

LIMITATIONS OF RELATED WORK

Looking across the three areas covered in this section, a consistent gap emerges. The ML-based routing works whether they rely on classifiers, reinforcement learning, neural networks, or hybrid pipelines all assume that the training data is reasonably balanced. The DTN routing protocols reviewed here tell a different story, Epidemic, Spray and Wait, Prophet, and MaxProp each generate traffic where bundles that fail to arrive outnumber those that succeed. This imbalance is not a minor statistical quirk, it is built into how these networks operate, and it directly affects what any ML model trained on their data will learn. The imbalanced data processing techniques reviewed here offer well-established tools for handling this kind of skew, yet none of the DTN routing studies have drawn on them. That combination, ML-based DTN routing on one side, imbalanced data handling on the other has not been brought together in any prior work, and addressing it is the central contribution of this paper.

METHODOLOGY AND APPROACH

Machine learning is a field of artificial intelligence (AI) that gives computers the ability to learn without being explicitly programmed. Researchers in this field attempt to mimic as closely as possible the functioning of the human brain and the patterns of information processing and communication observed in the biological nervous system, to give machines the ability to learn from data, interpret it and make informed decisions. Machine learning methods/algorithms have been successfully applied in several applications (image recognition, machine translation, medical diagnosis, networks, etc.), as well as in other different technological sectors in recent years (autonomous cars, intelligent robots, smart cities, etc.). However, the performance of the latter implicitly relies on the quality of the training data and requires a critical step called data pre-processing. It is defined as a method of preparing the data before applying the machine learning algorithms. This phase is of great interest especially when the data is noisy, sparse or also unbalanced.

Data sampling is one of the most widely used pre-processing methods. It is applied to address the problem of imbalance in datasets and are also known as dataset preprocessing methods. The main objective of sampling is to restore the balance between the membership associated with each class. These techniques are independent of the underlying classifier and can be performed in three ways, undersampling of the majority class, oversampling of the minority class, or a combination of both sampling techniques (hybrid methods) (G.S. et al. 2022a; Saez et al. 2015; Douzas, Bacao & Last 2018; Branco, Torgo & Ribeiro 2016; Ijaz, Attique & Son 2020).

This section begins by presenting the flow chart, which serves as a visual representation of the main challenge addressed in this paper. The data sampling techniques employed in the study are then outlined, followed by a comprehensive explanation of the oversampling and machine learning methods utilized in this research. To offer a clear depiction of the prediction process within this context, present FIGURE 2, which illustrates the overall system flowchart for addressing the main challenge of this paper.

INTELLIGENT SYSTEM FOR DTN ROUTING

This paper proposes an intelligent system capable of addressing the challenges of routing in DTNs and providing an efficient bundle classification system for different routing protocols. FIGURE 2 provides the overall architecture of the system including several components performing distinct tasks. This system tackles one of the important hurdles in machine learning, called the unbalanced learning problem where we have a skewed distribution of samples that have arrived and those that have not. Using the ML approach and data sampling technique, this solution balances the sample and improves the performance.

The principal components for the intelligent system for DTN routing include:

1. Data Collection: In this module, raw data is collected from DTN using a Java and Python-based emulator for each DTN protocol. The emulator generates data about bundle transmissions, node movements and so forth, emulating network conditions.
2. Data Preprocessing: The block which cleans the raw data acquired from the DTN. These values will go through the process of eliminating any inconsistencies and filling in any missing data so that the data so as to be fit enough to go through normalization further.
3. Feature Extraction: In this, the preprocessed data is used to extract relevant features. In the reporting on features, many of the features have been based on network characteristics such as node density, connectivity, and mobility patterns which may be used in predicting bundle delivery and classifying routing protocols.
4. Data Sampling: A collected dataset may be imbalanced sometimes. This step attempts to balance a dataset before feeding it to the machine learning model. With the dataset thus balanced and the model generalized and resistant to inaccurate training, techniques in oversampling, undersampling or SMOTE are performed.
5. Machine Learning Model: The ML model is trained on the balanced dataset generated by the data sampling techniques. The ML model is the core of an intelligent system, the ML model itself. By classifying routing protocols according to their performances, the prediction of bundle delivery success in DTN is realized.
6. Model Performance Evaluation: The performance of the new ML model is discussed in this section in terms of comparison of the ML model performance with the empirical evidence of predictions. Later, the model is evaluated by different performance metrics: precision, recall, F1 score, and accuracy.
7. Adaptive Routing: The system enables the selection of the most appropriate routing protocol according to the predictions and performance of the machine learning model, in order to achieve the best overall DTN routing performance.

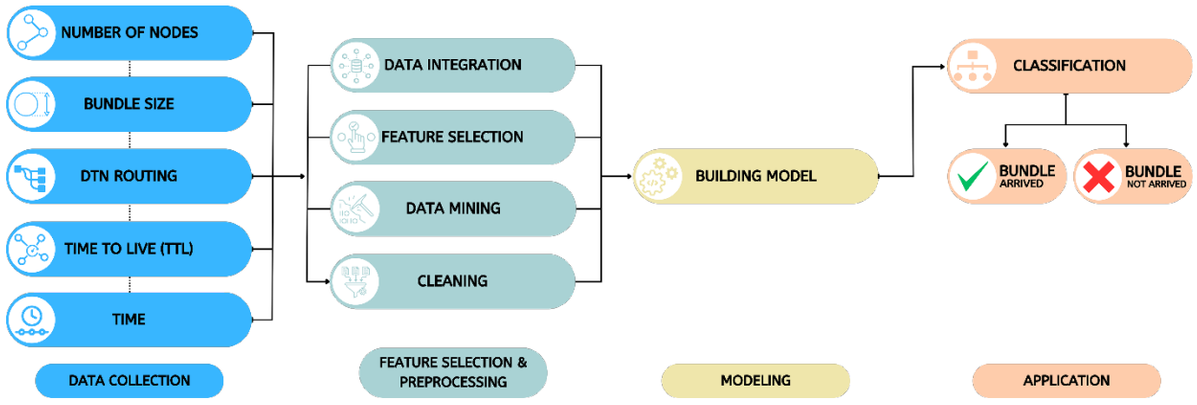


FIGURE 2. Flowchart for the DTN routing system based on intelligent techniques

Finally, the intelligent system discussed in this paper can provide a smart DTN routing in terms of unbalanced data and choose an adequate routing protocol. It adapts to a wide range of network conditions using machine learning and sampling and observes an overall DTN routing performance. FIGURE 3 further illustrates the conceptual model for bundle classification using machine learning techniques.

DESCRIPTION OF DATA

The dataset used in this study was collected through an emulator developed specifically for this purpose. It contains bundle information for the DTN protocols (see FIGURE 4), which is critical for studying the dynamics of DTN routing and the behavior of different protocols under varying network conditions.

This dataset was previously used, with its acquisition phase detailed in (Abdellaoui Alaoui et al. 2022). It comprises two classes of bundles, specifically those that arrived (class 1) and those that did not arrive (class 0). The dataset encapsulates a total of 20496 instances.

To provide further insight into the structure of our dataset, FIGURE 5 presents the class distribution of bundles for each of the considered DTN protocols: Epidemic, Spray and Wait, Prophet, and MaxProp. These graphs show the number of instances in each class (i.e., bundles that arrived and those that did not) for each

protocol. As illustrated in FIGURE 5, there are noticeable differences in the number of instances of Class 0 (bundles that did not arrive) and Class 1 (bundles that did arrive) across all considered DTN protocols: Epidemic, Spray and Wait, Prophet, and MaxProp.

Specifically, for the Epidemic protocol FIGURE 5(a), there are 12585 instances of Class 0 and 7910 instances of Class 1, indicating a significant imbalance. Similarly, the Spray and Wait protocol FIGURE 5(b) has 12115 instances of Class 0 and 8380 instances of Class 1. The imbalances continue with the Prophet protocol FIGURE 5(c) and the MaxProp protocol FIGURE 5(d), albeit with somewhat less discrepancy between classes.

These imbalances are not mere statistical aberrations; rather, they present a substantive challenge to machine learning-based DTN routing. Class imbalances can bias the training of machine learning models, leading them to over-predict the majority class and under-predict the minority class. Addressing these imbalances is essential for developing accurate and fair machine learning models for DTN routing. The subsequent sections detail the approach adopted for tackling this critical issue.

TABLE 1. Our datasets feature description. provides a comprehensive description of the dataset features, which include a total of nine attributes. These attributes represent various aspects of bundle information, such as the distance between nodes, estimated delivery time, bundle identification, departure time, bundle size, source and destination nodes, bundles time to live, and finally, the class label indicating whether a bundle arrived at its destination. The table provides a clear overview of the dataset features, which can be useful for understanding the relationships between the different attributes and the outcome of the bundle delivery.

METHODS OF RESAMPLING DATA

Class imbalance is a fairly common problem in the pre-processing phase of machine learning. It is related to variables with two classes and can be observed in various situations, notably in marketing for predicting customer churn, in banking for fraud detection, and in networks for packet or security classification. In these cases, the number of negatives is often very small compared to the number of positives (sometimes the opposite): this is called class imbalance. The problem arises when modeling a variable where one of the classes concerns a very small number of observations. Learning from imbalanced data requires adapted strategies to obtain a correct classification of the minority class. In the literature, various approaches have been proposed to achieve more efficient classification, among them the sampling methods adopted in this paper. Sampling techniques are very simple, and easy to program, their main objective is to re-establish the balance between the numbers associated with each class, either by eliminating the number of majority examples (undersampling), or by artificially increasing the number of minority examples (oversampling). In addition, there are hybrid methods combining undersampling and oversampling.

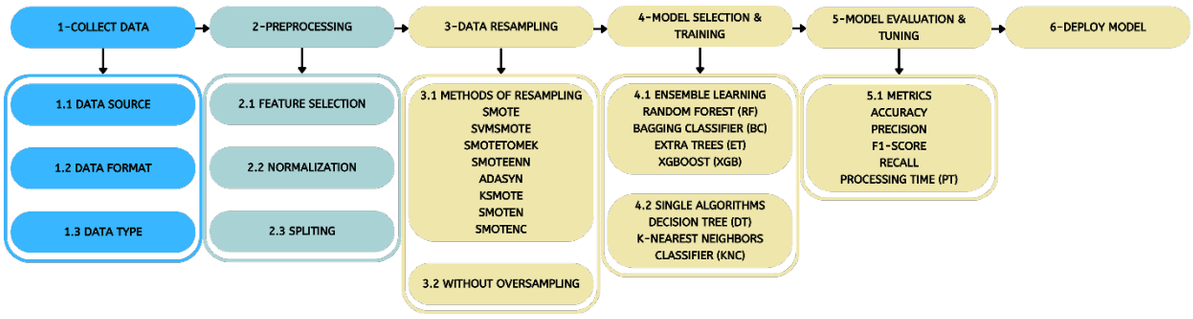


FIGURE 3. Conceptual model for bundle classification using machine learning techniques.

TABLE 1. Our datasets feature description.

Feature	Description
Distance at msg send	This numerical attribute represents the distance between the source and destination nodes when the message is sent.
Delivery time	An estimated delivery time of a bundle is the total transit time of the bundle and this numerical attribute.
ID	This categorical attribute is the identification of the bundle.
Time	This numerical attribute represents the departure time of a bundle from the source (fromHost) to the destination (toHost).

continue...

...cont.

Size	This numerical attribute is the size of a bundle.
FromHost	This categorical attribute represents the source node.
ToHost	This categorical attribute represents the destination node.
TTL	This numerical attribute is the time to live of a bundle, which indicates how long the bundle should be retained before being discarded or forwarded.
Class	This numerical attribute shows whether the bundle arrived at the destination or not (1 for arrived, 0 for not arrived).

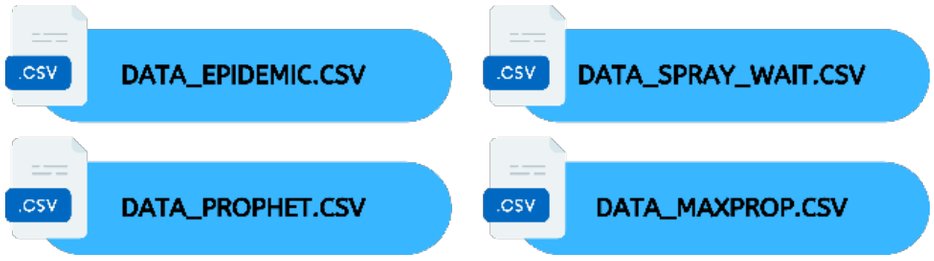


FIGURE 4. Data of DTN protocols.

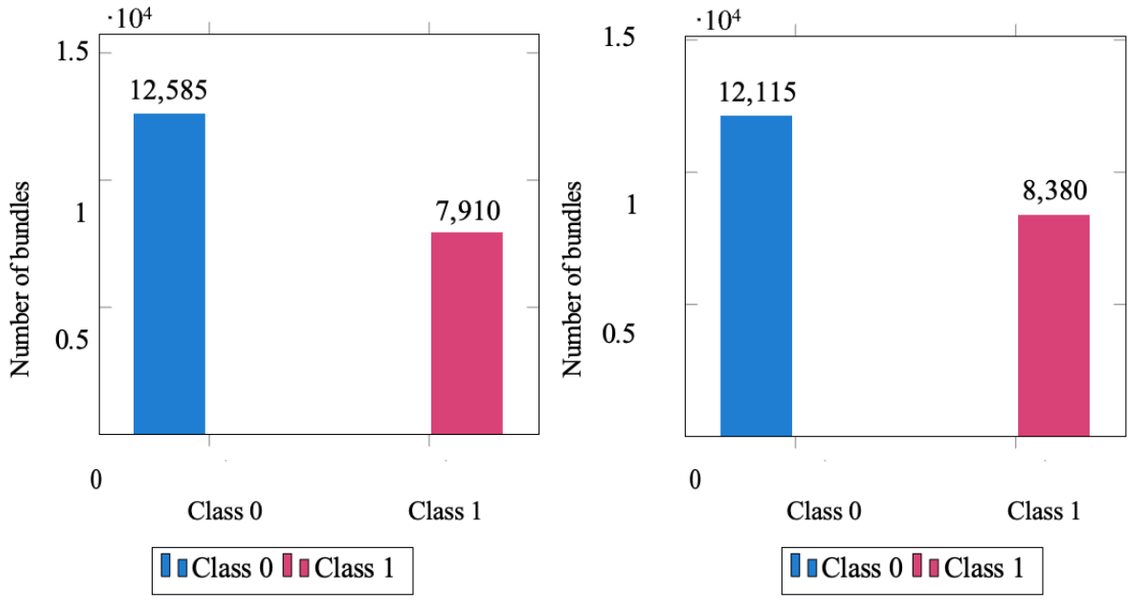
More formally, consider a dataset S with m observations (i.e., the cardinality of S: |S|=m).Let's define $S=\{(x_i,y_i),i=1,\ldots,m\}$,with $x_i \in X$, where X is the n- dimensional space of explanatory variables and $y_i \in \{0,1\}$ is the response variable. The new dataset obtained by resampling will be noted S'. Finally, the majority class will be noted S_{maj}, the minority class S_{min} such as $S_{maj} \subset S$, $S_{min} \subset S$, $S_{min} \cap S_{maj} = \varnothing$, $S_{min} \cup S_{maj} = S$. We talk about oversampling when adding data (i.e., $|S'_{min}|>|S_{min}|$) and about undersampling when the data size is reduced (i.e., $|S'_{maj}|<|S_{maj}|$).

In this paper, the Random Oversampling method is adopted to improve bundle classification performance. Random Oversampling consists in increasing the number of observations of the minority class by duplicating them by random drawing with replacement.

Consequently $|S'_{min}|>|S_{min}|$. The samples on the minority class are therefore increased without the loss of data on the majority class. The only parameter to take into account is the desired ratio β between the majority and minority classes, $\beta = \frac{|S'_{min}|}{|S_{maj}|}$:

$\beta = \frac{|S'_{min}|}{|S_{maj}|}$ No data is generated ($(S_{min} = |S_{min}|)$).

Data is generated to balance the minority and majority classes.

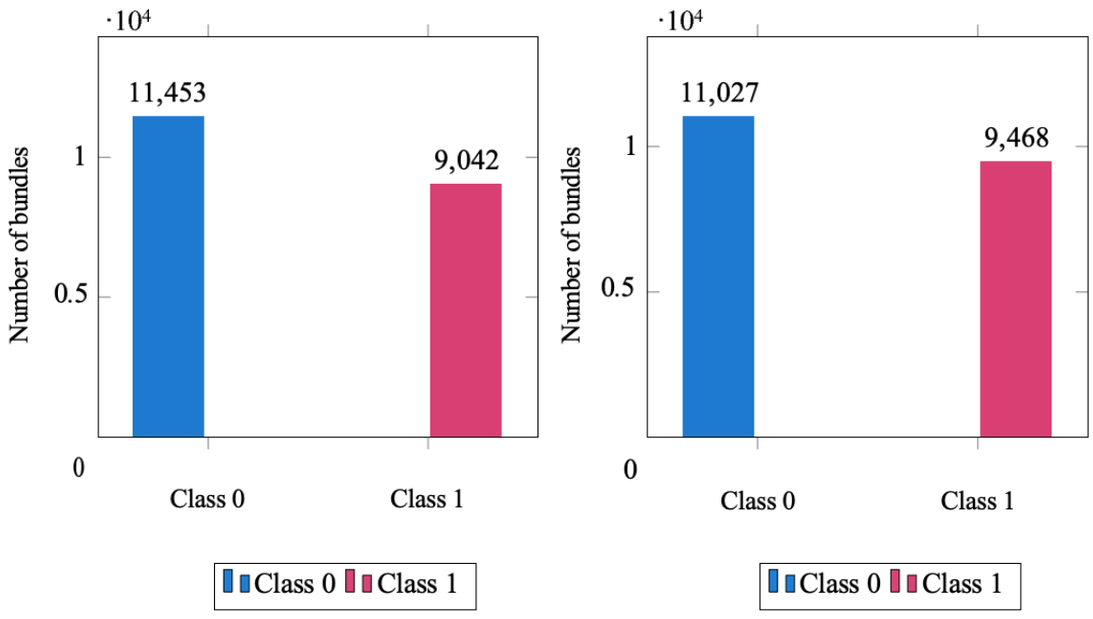


(a) Class Distribution for Epidemic

(b) Class Distribution for Spray and Wait

continue...

...cont.



(c) Class Distribution for Prophet

(d) Class Distribution for Maxprop

FIGURE 5. Class Distribution for each DTN protocol considered in this study

The number of observations of the minority class to be generated is therefore given by:

$$G = |S_{maj}| * \beta - |S_{min}| \quad (1)$$

Typically, the data is balanced such that $|S'_{\text{min}}| = |S_{\text{maj}}|$. When talking about oversampling, a commonly used notation is also '*a;b*', meaning *a* observations of the minority class for *b* observations of the majority class. So for a balanced dataset, the notation becomes 1:1. We will use this notation for all oversampling methods. With the random oversampling process, it is essential to note that:

- The number of duplications of each observation can be different. Observations may not be selected, leading to a loss of stability during the modeling phase.

Its biggest selling point is how little it asks, no synthetic data needs to be generated, and it works reasonably well when the class imbalance is moderate (Wongvorachan, He & Bulut 2023). Repeating the same minority samples over and over, though, tends to push the classifier toward fitting those exact points rather than learning the broader pattern (García et al. 2020).

The following section discusses aggregative strategies that can be used to address this issue.

SMOTE

SMOTE is an oversampling technique that generates new synthetic samples rather than simply duplicating existing ones. This technique creates new synthetic examples instead of oversampling by replacement. SMOTE introduces synthetic examples into the line segments to oversample the minority class samples. It connects each minority class sample to its k nearest neighbors. The choice of neighbors from the k nearest neighbors is random. The number depends on how much oversampling the model needs (Chawla et al. 2002; Roy et al. 2018; Faris et al. 2020).

The pseudo-algorithm of SMOTE is given below:

1. The oversampled data is selected (general data with minority class labels).
2. Select an instance of your data.
3. Randomly select a data point, which is within *k* nearest neighbors to the selected data point, and generate a synthetic data point anywhere on the line between these two data points.
4. Keep repeating until the data is balanced.

Unlike simply copying existing samples, generating synthetic points through interpolation gives the classifier a richer and more varied view of the minority class, which tends to improve its ability to generalize (Chawla et al. 2002). The risk, though, is that the interpolation is blind

to class boundaries, synthetic points can end up in regions that belong to the majority class, creating noise that confuses the model rather than helping it (Xiaolong, Wen & Yanfei 2019).

SMOTENN

SMOTENN is a hybrid technique that combines two widely used methods, SMOTE and ENN,, which was proposed by (Batista, Prati & Monard 2004; G.S. et al. 2022b), with the aim of removing noisy and redundant samples in the dataset by employing the rule that every misclassified sample by its nearest neighbors will be removed. It works so that for every sample in the dataset, its iteration computes the K-nearest neighbors and removes any sample misclassified by the majority of its neighbors. First, SMOTENN applies SMOTE to the minority class in the generation of synthetic examples as done in the original SMOTE technique. However, instead of using the whole dataset for training the model, SMOTENN applies ENN to clean any noisy or redundant samples generated by SMOTE, along with the majority of class samples that were wrongly classified. This results in a more balanced and cleaner dataset that one can use in training a classification model. Generally, SMOTENN is useful for handling imbalanced datasets comprising noisy samples and helps in enhancing the performance of a classification model by balancing the dataset and removing noisy samples. The steps of SMOTENN are as follows:

1. Apply SMOTE to the minority class to generate synthetic samples until the desired balance ratio is reached.
2. For every sample in the merged dataset, find its k nearest neighbors (default $k = 3$).
3. If the majority class among the k nearest neighbors differ from the sample's own class, remove that sample and its k nearest neighbors from the dataset.
4. Return the cleaned and balanced dataset.

Cleaning out noisy samples after oversampling, rather than just generating new ones, is what makes SMOTENN produce a noticeably tidier decision boundary than SMOTE alone, which tends to help the classifier distinguish between classes more confidently. The aggressive cleaning does come at a cost, though SMOTENN removes more samples than most other hybrid methods, and in some cases it discards examples that were actually informative (Batista, Prati & Monard 2004).

SMOTETOMEK

Synthetic Minority Oversampling Technique (SMOTE) and "Tomek link" are two different data sampling methods combined to create "SMOTEtomek" used in machine learning (Batista, Prati & Monard 2004; G.S. et al. 2022b), while "Link Tomek" is a method for finding and eliminating noisy edge samples from a dataset, the "SMOTETomek" method first uses "SMOTE" to sample collision classes. A small group was taken and later used the "Tomek link" to exclude any noisy regions or samples. This should create a balanced data set that is indicative of more fragmentation of hidden data and not be as prone to overexertion. It has been seen that the machine learning models perform very well when the approach used is "SMOTETomek", especially in the unbalanced datasets where minorities are highly underrepresented. However, it should be noted that "SMOTETomek" may not always be the best strategy for every dataset. The steps of SMOTETomek are as follows:

1. Apply SMOTE to the minority class to generate synthetic samples until the desired balance ratio is reached.
2. Identify all Tomek links in the resulting dataset, where a Tomek link is a pair of samples from different classes that are each other's nearest neighbor.
3. Remove the majority-class sample from each identified Tomek link.
4. Return the cleaned and balanced dataset.

Targeting only the samples sitting right on the class boundary, rather than cleaning the whole dataset, keeps the data loss to a minimum while still sharpening the separation between classes. The flip side is that Tomek links are quite selective, so when the imbalance is severe, removing a handful of borderline points barely moves the needle (Batista, Prati & Monard 2004).

ADASYN

Adaptive Synthetic Sampling (ADASYN) was developed by He and colleagues (He et al. 2008; Aditsania, Adiwijaya & Saonard 2017), which aims to improve the performance of the SMOTE algorithm by considering the distribution of Minor class instances.

This method generates synthetic observations based on the k-nearest neighbors of each observation in the minority class, with the sole parameter being the desired balancing ratio β.

Equation 1 corresponds to the calculation of the number of minority class observations to be generated (G).

For each minority class observation x_i, r_i is defined as the ratio of k-nearest neighbors belonging to the minority class. r_i is calculated by dividing Δ_i (the number of k-nearest neighbors belonging to the majority class) by K:

$$G = |S_{maj}| * \beta - |S_{min}| \quad (2)$$

r_i takes values in the interval [0, 1]. To obtain a probability density, $\hat{r}_i$ is normalized such that the sum of all equals 1:

$$G = |S_{maj}| * \beta - |S_{min}| \quad (3)$$

Using this density, it is possible to calculate the number of synthetic observations g_i to be generated for each minority class observation:

$$G = |S_{maj}| * \beta - |S_{min}| \quad (4)$$

Thus, ADASYN generates synthetic samples by considering the distribution of the minority class instances and their k-nearest neighbors, allowing the sample generation to adapt to regions where the minority class is more challenging to learn.

The steps of ADASYN are as follows:

1. Calculate the total number of synthetic samples to generate G based on the desired balance level β.
2. For each minority class example x_i, find its K nearest neighbors and compute the ratio $r_i=\Delta_i/K$, where Δ_i is the number of majority class samples among the neighbors.
3. Normalize r_(i)to obtain the density distribution $\hat{r}_i$.
4. For each minority example x_i, generate $g_i=\hat{r}_i\times$G synthetic samples by interpolating between x_i and a randomly chosen neighbor.
5. Return the balanced dataset.

Putting more synthetic samples where the minority class is hardest to distinguish is what separates ADASYN from flat oversampling, it lets the classifier spend more effort on the tricky regions rather than treating all minority examples equally (He et al. 2008). Where it can go wrong is when a minority sample looks hard to learn simply because it sits near noisy data in that case, the extra generation just amplifies the noise (García et al. 2020).

SVM-SMOTE

Is an oversampling technique that creates new data points along the boundaries of the minority class (MiC) and majority class (MaC) in the original dataset (Nguyen, Cooper & Kamei 2011). This method expands the ethnicity classes. The minority class goes to the majority class while still maintaining the level. Low-density data points in MaC. This creates new data points with appropriate density and minimal overlap. To construct the hyperplane and determine the support vectors of MiC, the first approach uses the Support Vector Machine (SVM) algorithm, and then the best hyperplane for the dataset is obtained by determining the margins. The largest distance between two classes using a Radial Basis Function (RBF) kernel can be determined using Equation 5. This technique is useful for handling unbalanced datasets in heuristic-based routing protocols for ML-based DTN.

$$G = |S_{maj}| * \beta - |S_{min}| \quad (5)$$

where the bias is denoted by 0 and the weight vector by. It is possible to utilize Equation 5 to optimize the hyperplane.

$$G = |S_{maj}| * \beta - |S_{min}| \quad (6)$$

Especially for the adopted hyperplane the numerator is equal to 1 and the distance from the support vector is given by Equation 7:

$$G = |S_{maj}| * \beta - |S_{min}| \quad (7)$$

Equation 8 may be used to define the Radial Basis Kernel (RBK).

$$K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right); \quad \gamma > 0 \quad (8)$$

For creating new instances by SVM-SMOTE, it identifies the support vectors of the minority class and then generates new instances by interpolating or extrapolating every minor data point with all the points within its kNN group. If the density of data points in the majority class around the minor data point is less than half of its nearest neighbors, new instances will be created outside of the kNN group to expand the minority class area toward the majority class. Conversely, if the density of the minority class around the data point is less, a new instance will be created inside the kNN group. The process aims to remove

less dense instances in the majority class and consolidate the minority class. However, instead of randomizing the selection of the nearest neighbor, the point is chosen based on the order of the kth nearest neighbor. This method helps to address the issue of unbalanced data in machine learning-based routing protocols for DTNs.

Equation 9 and Equation 10 represent the new data point for the extrapolate and interpolate, respectively.

$$X_{\text{new}}^{+} = SV_i^{+} + (SV_i^{+} - A_k) * \lambda \tag{9}$$

$$X_{\text{new}}^{+} = SV_i^{+} + (A_k - SV_i^{+}) * \lambda \tag{10}$$

Where, SV_i^{+} is positive support vectors, A_k is an array containing k positive nearest neighbors of each positive and $\lambda \in [0,1]$.

All equations used in this part form the theoretical foundation for our analysis. These equations were originally proposed by the reference (Islam & Mustafa 2023). For more detailed information, please refer to this source. The steps of SVM-SMOTE are as follows:

1. Train an SVM classifier on the original dataset to identify the support vectors of the minority class along the decision boundary.
2. For each minority class support vector, find its k nearest neighbors.
3. If more than half of the k nearest neighbors belong to the majority class, generate new synthetic samples by interpolation between the support vector and its neighbors.
4. If less than half of the k nearest neighbors belong to the majority class, generate new synthetic samples by extrapolation to expand the minority class region toward the majority class.
5. Return the oversampled training dataset.

Training an SVM first to locate the actual decision boundary means the new samples end up where they matter most right at the edge between classes, rather than scattered across the minority region. That said, fitting an SVM before any oversampling happens adds a noticeable computational overhead, and the quality of the result depends quite a bit on how well the SVM itself is tuned (Nguyen, Cooper & Kamei 2011).

KSMOTE

Kalman-SMOTE (KSMOTE) is an oversampling technique for unbalanced datasets which combines the SMOTE with Kalman filtering was proposed in the article (G.S. et al. 2022a).

KSMOTE uses the SMOTE algorithm to generate synthetic minority class samples, while Kalman filtering is applied to reduce noise in the generated samples.

A Kalman filter is a mathematical algorithm that uses a set of measurements to estimate the state of a system over time. Even with noisy measurements, a Kalman filter is then used to adjust the generated samples to make them more realistic and closer to the actual minority class distribution.

This filtering process reduces noise and variance in the generated samples, resulting in more accurate and efficient oversampling. In terms of classification accuracy and F1 score, KSMOTE has been shown to outperform other oversampling techniques such as SMOTE and ADASYN. However, the additional Kalman filtering step makes KSMOTE computationally more expensive than other techniques.

The steps of KSMOTE are as follows:

1. Apply SMOTE to the minority class to generate synthetic samples until the desired balance ratio is reached.
2. Combine the synthetic samples with the original dataset to form a single merged dataset.
3. Apply the Kalman filter to the merged dataset to identify and remove noisy samples from both the original and synthetic instances.
4. Return the filtered and balanced dataset.

Running the Kalman filter over the merged dataset after SMOTE catches the noisy points that SMOTE alone would leave behind, which tends to give the classifier a cleaner signal to learn from. The added filtering step, though, makes the method noticeably heavier to run, particularly when the dataset is large (G.S. et al. 2022b).

SMOTENC

SMOTENC (Synthetic Minority Oversampling Technique for Nominal and Continuous features) is an extension of SMOTE proposed by Chawla et al. (2002) to address a key limitation of the original algorithm: standard SMOTE assumes all features are continuous, which leads to meaningless synthetic values when applied to datasets containing categorical attributes. SMOTENC handles this by applying a modified distance metric that accounts for both feature types, and by treating continuous and categorical features differently during sample generation. It requires that the dataset contains at least one continuous feature to function.

The steps of SMOTENC are as follows:

1. For each minority class sample, compute distances to all other minority samples using a modified metric that handles both continuous and categorical features.
2. Identify the *k* nearest neighbors of the sample.
3. For each continuous feature, generate a synthetic value by interpolating between the sample and a randomly selected neighbor, as in standard SMOTE.
4. For each categorical feature, assign the most frequent value observed among the *k* nearest neighbors.
5. Repeat until the desired class balance is reached.
6. Return the balanced dataset.

When the dataset mixes continuous and categorical features which is common in real-world routing data, standard SMOTE breaks down because it treats all attributes as numbers. SMOTENC handles this directly by processing each feature type on its own terms. The catch is that it still needs at least one continuous feature in the data to work, a fully categorical dataset falls outside what it can handle (Chawla et al. 2002).

MODEL DEVELOPMENT

Data sampling techniques are employed to balance the class distribution in imbalanced datasets. Following this balancing step, the process of training, validating, and testing a machine learning algorithm to classify bundles begins. In this section, we define the machine learning methods that can be used to accomplish this task.

K-NEAREST NEIGHBOR (K-NN)

It aims to search the class of new data points to find the nearest neighbors around the new data points (Shaikhina et al. 2019; Mahbooba et al. 2021; Sun et al. 2018; Jindal et al. 2016). It calculates distances to all nearby points and filters out points with short distances from unknown data. Therefore, it is often called a distance-based algorithm (Shaikhina et al. 2019; Mahbooba et al. 2021; Sun et al. 2018; Jindal et al. 2016). It develops a majority voting system, where Equation 11 computes the distance between data points using the Euclidean distance metric.

$$d(s,t) = \sqrt{\sum_{i=1}^{n}(s_i - t_i)^2} \tag{11}$$

s and *t*: two points in the Euclidean *n*-space.

Being a lazy learner, K-NN requires no training phase, the model simply stores the training data and defers all computation to prediction time, which makes it easy to update as new samples arrive (Cunningham & Delany 2021). The downside of this design is that every new prediction involves scanning the entire training set to find the nearest neighbors, so the computational cost grows directly with the size of the data (Cunningham & Delany 2021).

DECISION TREE (DT)

A common ML method used for both classification and regression applications is decision trees. It works by recursively partitioning the input data into smaller subsets based on feature values, with each split determined by the majority class for classification tasks or the average value for regression tasks (Shaikhina et al. 2019; Mahbooba et al. 2021; Sun et al. 2018; Jindal et al. 2016). What makes decision trees stand out is how easy it is to follow the logic behind a prediction, the tree structure maps directly to a sequence of if-else conditions that anyone familiar with the problem can read and verify (Kotsiantis 2007). That said, a tree that is left to grow without any restriction will tend to memorize the training data rather than learn from it, and even a minor change in the input can produce a structurally different tree (Kotsiantis 2007).

RANDOM FOREST CLASSIFIER (RF)

It builds a large number of decision trees on different subsets of the data. Rather than relying on a single tree, a random forest aggregates the predictions of all trees and determines the final outcome through majority voting (Shaikhina et al. 2019; Mahbooba et al. 2021; Sun et al. 2018; Jindal et al. 2016). Aggregating predictions from many trees, each built on a different data subset, is what gives random forests their edge over individual trees, Breiman showed that the error rate keeps improving and does not overfit as more trees are added (Breiman 2001). The cost of this, though, is that the model becomes a black box, unlike a single tree whose decisions can be read step by step, there is no practical way to look inside hundreds of trees and understand what drove a particular outcome (Schonlau & Zou 2020).

BAGGING CLASSIFIER (BC)

Bagging or bootstrapping is a method of combining statistical models used for classification or regression. Developed by Breiman (1996), it involves building multiple models using random drawings. These models are then combined through voting.

Retaining the previous notations, let S be a sample data of size m, $S=\{(x_i,y_i),i=1,\ldots,m\}$ with $x_i \in X$ being the dimension of explanatory variables and $y_i \in \{0,1\}$ being the response variable. By considering the random function $U:1,\ldots,m \rightarrow 1,\ldots,m$, the bagging method generates K new samples $SU^k=(X^k_{U(1)},\ldots,X^k_{U(m')})$ of size m' by random drawing with replacement in S (bootstrap (Tibshirani & Efron 1993)). Thus, it is possible to have repetitions of observations in each S_k or that some are never selected. The random variables U are defined over $U=(u_k)^k=1$, the set of drawings of size K with replacement, whose cardinality is m^K. $B=\{U_1,\ldots,U_{BK}\}$ is defined as the set of generated random functions. On these m samples, m models are constructed and combined by voting for classification or by averaging for regression. Assuming that $\varphi(x,S)$ is the predictor, by using the data from S and x_i, the explanatory variables of the individual i, then $y_i=\varphi(x_i,S)$.

Bagging tends to pay off most when the base learner is unstable, that is, when small changes in the training data lead to noticeably different models. In those cases, averaging over many bootstrap samples brings meaningful accuracy gains (Breiman 1996). When the base learner is already stable, however, the same averaging process adds computational overhead without much return (Breiman 1996).

EXTRATREES CLASSIFIER (ET)

The Extra Trees (ET) classifier is an ensemble learning method that belongs to the family of decision tree-based methods, similar to the Random Forest classifier. The main difference between Extra Trees and Random Forest is that Extra Trees uses a random subset of features for each split in the tree, whereas Random Forest uses the best split based on a random subset of features. This difference makes Extra Trees more computationally efficient and less prone to overfitting than Random Forest. Randomizing both the feature selection and the split threshold at each node rather than searching for the best split is what gives Extra Trees its speed advantage over Random Forest (Geurts, Ernst & Wehenkel 2006). The same randomization that cuts computation time, however, tends to introduce more bias into individual trees, which can affect accuracy when the data has clear and informative split points that the algorithm may miss (Geurts, Ernst & Wehenkel 2006).

XGBOOST CLASSIFIER (XGB)

XGboost, proposed by Chen & Guestrin (2016), is a very efficient variant of the so-called Friedman gradient boosting methods (Friedman 2001). It is currently one of the most competitive methods in machine learning. The process is based on the boosting method combined with the prediction of K weak learners to build a strong predictor using a particular learning strategy. The weak learners used are decision trees (CART). The main goal is to prevent overfitting and optimize hardware resources by proposing an inexpensive method in terms of computation time. This can be done by simplifying the cost functions and using regularization terms.

XGboost works as follows. Let X_i be the explanatory variable vector of the observation i ($X_i=\{x_1,x_2,\ldots,x_n\}$, i=1,2,…,$N$) and y_i the target variable to predict ($y_i \in \{-1,+1\}$, $i=1,2,\ldots,N$). The learning phase is done using an additive strategy (identical to boosting), i.e., each classifier is built one after the other. The first classifier is built on the original learning base. The weights are redefined so that the second will give more importance to the observations misclassified by the first. The final prediction function is given by:

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), \quad f_k \in \mathcal{F}, \\ \text{with } \mathcal{F} = \{f(x) = w_{q(x)}\}(q: \mathbb{R}^m \rightarrow T, w \in \mathbb{R}^T) \tag{12}$$

the space of the regression trees, q the structure of each tree and T the number of leaves. Each f_k represents a structure tree with leaf weights w. Consequently, learning is possible by minimizing the following regularized function:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k), \\ \text{where } \Omega(f) = \gamma T + \frac{1}{2}\lambda \parallel w \parallel^2, \tag{13}$$

where l is a convex differentiable loss function that measures the difference between the prediction $\hat{y}_i$ and the actual value y_i. The second term Ω penalizes thecomplexity of the model to avoid over-fitting problems.

Equation 12 and Equation 13 represent the theoretical foundation for our analysis. They were originally proposed by the reference (Chen & Guestrin 2016). For more detail, please refer to this source.

What sets XGBoost apart in practice is how it handles scale built-in regularization keeps the model from memorizing the training data, and the system was designed from the ground up to run efficiently on large datasets (Chen & Guestrin 2016). That said, getting the best out of it is not straightforward: the number of hyperparameters to tune is considerable, and a poorly configured model can perform worse than simpler alternatives (Chen & Guestrin 2016).

EMPIRICAL ANALYSIS AND RESULTS

This section presents the results of performance experiments evaluating the proposed approach for four DTN routing protocols and comparing them to state-of-the-art techniques. The quality and quantity of the collected data are essential components for constructing a reliable classification system that classifies bundles as successfully delivered or unsuccessfully delivered for each DTN protocol. Given the modified technique, the characteristics of DTN, including bundle size, source, destination, distance between them, bundle time-to-live (TTL), and others, may influence the classification results. The precision of these statistics is crucial to achieving the objectives of this study, and is therefore closely monitored. The results are analyzed using the performance metrics described in the *Model Evaluation Indicators* subsection.

MODEL EVALUATION INDICATORS

Several types of performance indicators can be used to estimate the quality of binary classifiers with different objectives. The confusion matrix presented in TABLE 2. Confusion matrix for binary classification is used to distinguish the best (optimal) solution. The column of the table represents the predicted class, while the row represents the actual class. In this confusion matrix, *tp* and *tn* denote the number of positive and negative instances that are correctly classified. Meanwhile, *fp* and *fn* denote the number of negative and positive instances that are misclassified, respectively.

TABLE 2. Confusion matrix for binary classification

	Predicted positive class	Predicted negative class
Actual positive class	tp	fn
Actual negative class	fp	tn

Several commonly used metrics are presented in TABLE 3. Evaluation Metrics and Objectives to measure the predictive ability of classifiers. Accuracy is the most commonly used metric in practice for classification problems. With accuracy, the quality of the produced solution is judged by the percentage of correct predictions over the total number of instances.

TABLE 3. Evaluation Metrics and Objectives

Metric	Formula	Objective of the Evaluation
Accuracy	$\frac{tp+tn}{tp+fp+tn+fn}$	Measuring the ratio of correct predictions to the total number of instances evaluated.
Precision	$\frac{tp}{tp+fp}$	Calculate the ratio between the number of correctly predicted positive instances and the total number of predicted instances in a positive class.
Recall	$\frac{tp}{tp+fn}$	Represents the fraction of positive instances that are correctly classified.
f1-Score	$\frac{2\times P\times R}{P+R}$	Indicate the harmonic average between the recall and precision values.
PT	$end_{time}^{testing} - start_{time}^{testing}$	Measures how long it takes a learning algorithm to classify the entire testing data as a bundle arrived or not.

In evaluating our models performance, we employed only these five metrics: Accuracy, Precision, F1-score, Recall, and PT. We have deliberately chosen these metrics in line with our previous work (Abdellaoui Alaoui et al. 2022, 2023), thus facilitating a direct comparison. While other metrics such as F1 Macro and G-mean are beneficial when dealing with imbalanced data, in this study we have opted to use only these metrics in order to maintain consistency with our previous works (Abdellaoui Alaoui et al. 2022, 2023).

EXPERIMENTAL PROTOCOL

The studies described in this article were conducted on a Google Colaboratory tool (Carneiro et al. 2018), which has 12 GB of RAM, 110 GB of disk space, runs Ubuntu 17.10 64 bits, and has an Intel Xeon CPU (not specified). Google Colaboratory is a cloud-based service built on Jupyter Notebooks that allows users to develop machine learning and deep learning applications in Python, with access to a free GPU processor. To access this service, all we need is a Google account. TABLE 4. Tools utilized in this study presents the various tools used in this work.

TABLE 4. Tools utilized in this study

Tools	Google Colab	Sklearn	Jupyter lab	Python
Version	4.11.0	0.24.2	3.2.1	3.9.7

RESULTS ANALYSIS

This section analyzes the results of a comparison study of the proposed approach with respect to various Delay-Tolerant Networking (DTN) protocols. The performance of each protocol was evaluated using the metrics described in the *Model Evaluation Indicators* subsection.

CLASSIFICATION PERFORMANCE OF EPIDEMIC PROTOCOL

TABLE 5 examines the effectiveness of various classification algorithms using different oversampling methods to handle imbalaced data. The studied classifiers include Random Forest (RF), Bagging Classifier (BC), Extra Trees (ET), XGBoost (XGB), Decision Tree (DT), and K-Nearest Neighbors Classifier (KNC). The compared oversampling techniques include SMOTE, SVMSMOTE, SMOTETomek, SMOTENN, ADASYN, KSMOTE, SMOTEN, and SMOTENC, in addition to reference performances without oversampling. The performance indicators are accuracy, precision, F1 score, recall, and processing time (PT). The Epidemic protocol bundle classification results under different data balancing methods are presented in the same table. The findings show that the ET algorithm, combined with the SMOTENN data balancing method, is the most effective according to the performance metrics. Similarly, for other algorithms, the SMOTENN method has led to a considerable improvement compared to other preprocessing techniques, mainly introduced to solve the problem of imbalanced data. The ET algorithm, due to its nature (which tends to minimize prediction variance), requires balanced classes to be effective, which explains this result.

Looking at the individual classifiers more closely, KNC stands out as the most affected by the imbalance, its recall without oversampling reaches only (0.3470), meaning the model was essentially ignoring the minority class almost entirely. Even with SMOTE, KNC only reaches 0.6338, which is well below every other classifier at the same balancing level. The jump to (0.8957) with SMOTENN tells a clear story: what KNC needed was not more synthetic samples, but cleaner ones.XGB shows a different pattern, it starts with the highest accuracy among all classifiers without oversampling (0.5919), yet drops noticeably with SMOTE (0.7604), falling behind RF, BC, and ET. This suggests that XGB's boosting mechanism, which amplifies misclassified samples, is particularly sensitive to the noise that SMOTE can introduce at the class boundaries of the Epidemic dataset.

Furthermore, the ET classification performance is quite similar to that of ensemble methods algorithms (XGB, RF, BC), while individual techniques such as DT show the largest gap between preprocessing methods. Consequently, a performance of 81% to 92% in accuracy for the ET algorithm is a significant performance gain, highlighting the importance of the preprocessing phase, mainly when dealing with a situation where the data is imbalanced. In addition, the classifier achieved good recognition of the minority class and the majority classs observations. Moreover, the experiments demonstrate that the minority and majority class samples are correctly classified (i.e. TP and TN have increased, FP and FN have decreased). Finally, we observe that the performance results of each ML algorithm, in combination with the same data balancing methods, are consistent across all performance indicators (Accuracy, Precision, F1-Score, and Recall), confirming the improvement brought about by these preprocessing methods in terms of the ML models classification performance.

Taken together, the Epidemic results give a clear answer to the first two research questions: balancing methods do work for bundle classification, and the choice of method matters SMOTENN consistently outperforms SMOTE across all six classifiers in this protocol, with the margin widest for KNC and DT, the two classifiers most sensitive to noisy training data.

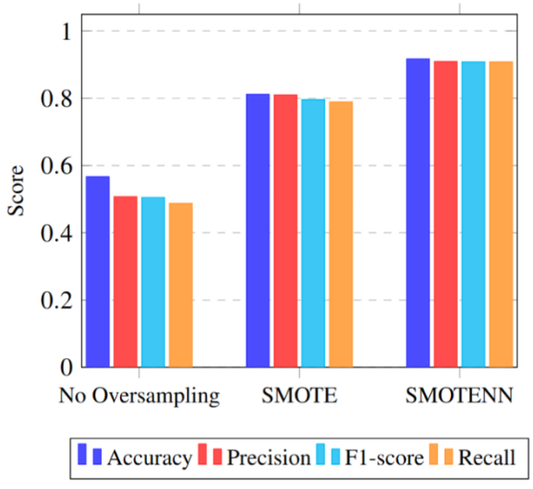


FIGURE 6. ET classifier performance on the Epidemic protocol under three data balancing conditions

FIGURE 6. ET classifier performance on the Epidemic protocol under three data balancing conditionsillustrates the classification performance of the Extra Trees classifier on the Epidemic protocol across three representative conditions: without oversampling, with SMOTE, and with SMOTENN. The Epidemic protocol is selected here as a representative example because it is the most widely studied DTN protocol in the literature and exhibits the

clearest class imbalance effect among the four protocols considered in this study. As the figure shows, all four metrics Accuracy, Precision, F1-score, and Recall follow a consistent upward trend from left to right, with the most substantial gain occurring between SMOTE and SMOTENN. The same pattern is observed across the remaining three protocols, as summarized in FIGURE 7.

CLASSIFICATION PERFORMANCE OF SPRAY AND WAIT PROTOCOL

This section presents the results of bundle classification for the Spray and Wait protocol, as illustrated in TABLE 6 The results show that the ET algorithm, in combination with the SMOTENN data balancing technique, is the most effective according to performance metrics. Once again, introducing pre-processing techniques has improved the classification capability of various algorithms, particularly for the ET algorithm, where this improvement exceeded 67%. The introduction of pre-processing techniques designed primarily to address the problem of data imbalance has improved the performance of different algorithms, highlighting the importance of this critical phase in automatic bundle classification in the context of DTN networks.

Looking at the numbers directly, ET without oversampling sits at (0.5423) and reaches (0.9067) with SMOTENN, which is where the 67% figure comes from. One observation worth noting is that BC comes very close to ET in this protocol (0.9049) versus (0.9067) a tighter gap than what was seen with the Epidemic dataset. This makes sense given that Spray and Wait's more moderate imbalance ratio (12,115 vs 8,380) leaves less room for ET's variance-reduction advantage to pull ahead. A separate observation concerns RF with SMOTETomek, which drops to (0.7784) actually below its SMOTE result of (0.8031). In this protocol, Tomek link removal appears to discard samples that were genuinely informative for RF rather than just borderline noise, a reminder that hybrid methods are not universally beneficial. KNC again shows the same pattern as in the Epidemic protocol: (0.6372) with SMOTE, then a sharp jump to (0.8934) with SMOTENN, confirming that cleaning matters more than generating for this classifier.

CLASSIFICATION PERFORMANCE OF PROPHET PROTOCOL

TABLE 7 presents a thorough analysis of the classification performances of the various algorithms when tested on the Prophet Protocol Dataset. In addition to the baseline performance, which was computed where no data balancing preprocessing technique was applied, different data balancing methods were utilized, and a comparative study was conducted. Preliminary results demonstrate that the ET algorithm outperforms other models when applied to the SMOTENN-based balanced dataset. An overall accuracy of 89.71% is observed, which represents about 68.7% improvement when compared to baseline performance. A remarkable improvement in the different metrics is noticed when using other data-balancing techniques regarding the performance of the different ML-based models. However, when comparing the performance of the various ML models on previous protocol datasets (i.e., Epidemic and Spray and Wait Protocols), we noticed that the performance of various models decreased slightly, which confirms the complexity of this protocol when compared with previous protocols.

What makes Prophet stand out from the previous two protocols is that its imbalance ratio is actually the most moderate so far (11,453 vs 9,042), yet the overall accuracy with SMOTENN drops to (0.8971) below both Epidemic (0.9168) and Spray and Wait (0.9067). The likely reason is that Prophet's probabilistic routing mechanism generates traffic patterns where the boundary between arriving and non-arriving bundles is inherently less clear, regardless of how well the classes are balanced. Two observations from the table support this reading. First, ET with SMOTE (0.7985) falls slightly below RF (0.8078) the only protocol where this reversal occurs which suggests that ET's low-variance splitting strategy is less helpful when the class boundary itself is noisy. Second, ADASYN performs noticeably worse here than in previous protocols: XGB drops to (0.7456) and KNC to (0.6037) with ADASYN, both well below their SMOTE results. ADASYN focuses generation on the hardest minority samples, and in a protocol where hard samples are hard because of genuine overlap rather than imbalance, this strategy amplifies the problem rather than addressing it.

TABLE 5. Performance of classification using Epidemic data

		Epidemic Protocol				
Classifier	Method	Accuracy	Precision	f1-score	Recall	PT
RF	Without oversampling	0.5752	0.5073	0.5051	0.4855	0.123
	SMOTE	0.8087	0.8045	0.7940	0.7879	0.066
	SVMSMOTE	0.8238	0.8266	0.8242	0.8245	0.044
	SMOTETomek	0.8346	0.8381	0.8360	0.8362	0.041
	SMOTENN	0.8990	0.9003	0.8992	0.8983	0.055
	ADASYN	0.8171	0.8189	0.8157	0.8157	0.041
	KSMOTE	0.8289	0.8277	0.8259	0.8262	0.064
	SMOTEN	0.8199	0.8207	0.8184	0.8188	0.042
	SMOTENC	0.8024	0.8042	0.8017	0.8022	0.119
BC	Without oversampling	0.5696	0.5051	0.5034	0.4859	0.032
	SMOTE	0.8021	0.7970	0.7870	0.7813	0.035
	SVMSMOTE	0.8200	0.8223	0.8196	0.8201	0.033
	SMOTETomek	0.8361	0.8379	0.8359	0.8361	0.029
	SMOTENN	0.8997	0.8982	0.8974	0.8969	0.029
	ADASYN	0.8157	0.8178	0.8150	0.8149	0.040
	KSMOTE	0.8262	0.8273	0.8260	0.8262	0.041
	SMOTEN	0.8196	0.8212	0.8193	0.8197	0.040
	SMOTENC	0.7978	0.8000	0.7973	0.7977	0.030
ET	Without oversampling	0.5668	0.5072	0.5051	0.488	0.047
	SMOTE	0.8119	0.8097	0.7964	0.7893	0.096
	SVMSMOTE	0.8286	0.8302	0.8286	0.8289	0.050
	SMOTETomek	0.8427	0.8472	0.8459	0.8460	0.037
	SMOTENN	0.9168	0.9093	0.9087	0.9084	0.044
	ADASYN	0.8201	0.8225	0.8198	0.8196	0.065
	KSMOTE	0.8340	0.8353	0.8331	0.8334	0.039
	SMOTEN	0.8191	0.8197	0.8166	0.8171	0.048
	SMOTENC	0.8090	0.8074	0.8063	0.8067	0.050
XGB	Without oversampling	0.5919	0.5109	0.5048	0.4601	0.405
	SMOTE	0.7604	0.7531	0.7392	0.7331	0.700
	SVMSMOTE	0.7870	0.7974	0.7849	0.7869	0.044
	SMOTETomek	0.8107	0.8172	0.8097	0.8107	1.130
	SMOTENN	0.9023	0.9010	0.9001	0.8997	0.349
	ADASYN	0.7864	0.7970	0.7836	0.7847	1.140
	KSMOTE	0.7986	0.8026	0.7977	0.7985	0.605
	SMOTEN	0.7910	0.7918	0.7908	0.7910	0.624
	SMOTENC	0.7854	0.7914	0.7842	0.7855	0.458

continue...

...cont.

		Epidemic Protocol				
Classifier	Method	Accuracy	Precision	f1-score	Recall	PT
DT	Without oversampling	0.5243	0.3827	0.3945	0.3885	0.002
	SMOTE	0.8012	0.7910	0.7914	0.7917	0.002
	SVMSMOTE	0.8035	0.8024	0.8007	0.7992	0.002
	SMOTETomek	0.8152	0.8136	0.8161	0.8188	0.002
	SMOTENN	0.8782	0.8902	0.8933	0.8966	0.001
	ADASYN	0.7906	0.7860	0.7880	0.7900	0.010
	KSMOTE	0.8068	0.8070	0.8098	0.8129	0.003
	SMOTEN	0.8061	0.7998	0.8062	0.8129	0.002
	SMOTENC	0.7883	0.7918	0.7868	0.7821	0.003
KNC	Without oversampling	0.5561	0.3990	0.3070	0.3470	0.107
	SMOTE	0.6338	0.6248	0.6247	0.6300	0.123
	SVMSMOTE	0.6913	0.6776	0.7028	0.7301	0.145
	SMOTETomek	0.7038	0.6844	0.7186	0.7566	0.144
	SMOTENN	0.8957	0.8868	0.9110	0.9370	0.078
	ADASYN	0.6751	0.6457	0.6941	0.7504	0.244
	KSMOTE	0.6992	0.6843	0.7109	0.8334	0.147
	SMOTEN	0.7013	0.6913	0.7088	0.7275	0.133
	SMOTENC	0.6416	0.6395	0.6439	0.6487	0.135

TABLE 6. Performance of classification using Spray & Wait data

		Spray & Wait Protocol				
Classifier	Method	Accuracy	Precision	f1-score	Recall	PT
RF	Without oversampling	0.5447	0.5048	0.50375	0.4886	0.117
	SMOTE	0.8031	0.7965	0.7925	0.7896	0.069
	SVMSMOTE	0.8110	0.8189	0.8162	0.8167	0.070
	SMOTETomek	0.7784	0.7794	0.7624	0.7563	0.144
	SMOTENN	0.8962	0.8952	0.8948	0.8945	0.046
	ADASYN	0.8142	0.8155	0.8135	0.8143	0.056
	KSMOTE	0.8138	0.8191	0.8154	0.8159	0.055
	SMOTEN	0.8044	0.8093	0.8071	0.8075	0.056
	SMOTENC	0.7951	0.7975	0.7950	0.7956	0.059
BC	Without oversampling	0.54110	0.5085	0.5065	0.4935	0.046
	SMOTE	0.7970	0.7907	0.7854	0.7819	0.700
	SVMSMOTE	0.8122	0.8144	0.8118	0.8123	0.044
	SMOTETomek	0.8307	0.8325	0.8304	0.8307	0.040
	SMOTENN	0.9049	0.9032	0.9027	0.9023	0.034
	ADASYN	0.8106	0.8119	0.8104	0.8110	0.055
	KSMOTE	0.8109	0.8137	0.8104	0.8109	0.094
	SMOTEN	0.8037	0.8055	0.8034	0.8038	0.040
	SMOTENC	0.7933	0.7952	0.7928	0.7933	0.040

continue...

...cont.

		Spray & Wait Protocol				
Classifier	Method	Accuracy	Precision	f1-score	Recall	PT
ET	Without oversampling	0.5423	0.5126	0.5102	0.4990	0.083
	SMOTE	0.8012	0.7957	0.7893	0.7853	0.106
	SVMSMOTE	0.8186	0.8208	0.8190	0.8193	0.082
	SMOTETomek	0.8352	0.8365	0.8354	0.8356	0.079
	SMOTENN	0.9067	0.9031	0.9032	0.9035	0.044
	ADASYN	0.8185	0.8207	0.8196	0.8201	0.073
	KSMOTE	0.8183	0.8199	0.8167	0.8171	0.195
	SMOTEN	0.8048	0.8073	0.8043	0.8048	0.076
	SMOTENC	0.7985	0.8037	0.8020	0.8023	0.051
XGB	Without oversampling	0.5645	0.5160	0.5097	0.4775	0.507
	SMOTE	0.7636	0.7605	0.7439	0.7377	0.700
	SVMSMOTE	0.7705	0.7819	0.7681	0.7705	0.469
	SMOTETomek	0.7995	0.8060	0.7983	0.7995	0.963
	SMOTENN	0.9020	0.9012	0.8997	0.8986	0.829
	ADASYN	0.7854	0.7930	0.7844	0.7867	0.703
	KSMOTE	0.7799	0.7875	0.7784	0.7798	0.578
	SMOTEN	0.7816	0.7834	0.7811	0.7817	0.549
	SMOTENC	0.7732	0.7789	0.7720	0.7733	0.558
DT	Without oversampling	0.5228	0.4123	0.4208	0.4162	0.008
	SMOTE	0.7943	0.7855	0.7873	0.7897	0.004
	SVMSMOTE	0.7916	0.7946	0.7912	0.7884	0.002
	SMOTETomek	0.8109	0.8122	0.8121	0.8123	0.002
	SMOTENN	0.8806	0.8872	0.8947	0.9025	0.008
	ADASYN	0.7871	0.7887	0.7906	0.7928	0.002
	KSMOTE	0.7998	0.8010	0.8007	0.8004	0.003
	SMOTEN	0.7915	0.7884	0.7929	0.7977	0.002
	SMOTENC	0.7833	0.7834	0.7803	0.7774	0.002
KNC	Without oversampling	0.5308	0.4100	0.338	0.3705	0.227
	SMOTE	0.6372	0.6294	0.6295	0.633	0.364
	SVMSMOTE	0.6781	0.6677	0.6878	0.7096	0.112
	SMOTETomek	0.6912	0.6734	0.7062	0.7426	0.113
	SMOTENN	0.8934	0.8786	0.9096	0.9430	0.106
	ADASYN	0.6698	0.6501	0.7003	0.7590	0.114
	KSMOTE	0.6793	0.6786	0.6800	0.6817	0.180
	SMOTEN	0.68197	0.6750	0.6881	0.7018	0.128
	SMOTENC	0.6358	0.6339	0.6382	0.6429	0.114

CLASSIFICATION PERFORMANCE OF MAXPROP PROTOCOL

TABLE 8 provides a comprehensive analysis of the classification performance of the various algorithms employing various data balancing techniques on the Maxprop Protocol dataset. Results indicate that, when applied to the SMOTENN-based balanced dataset, the ET algorithm outperforms other models. There is an aggregate accuracy of 89.52%, which represents a 70.97%

improvement over the performance at the outset. Regarding the performance of the various ML-based models, the use of other data-balancing techniques results in a significant improvement in terms of the various metrics. However, when comparing the performance of the various ML models on previous protocol datasets (i.e., Epidemic, Spray and Wait, and Prophet Protocols), we observed that the performance of the various models decreased a little, confirming the high complexity of this protocol in comparison to previous protocols.

MaxProp carries the lightest class imbalance of all four protocols (11,027 vs 9,468), yet the results follow the same downward trend seen in Prophet. ET with SMOTENN reaches (0.8952), slightly below Prophet's (0.8971) and well below Epidemic's (0.9168). This points to routing complexity, not class imbalance, as the main driver of difficulty MaxProp's priority-based message ordering creates classification patterns that are harder to separate regardless of how well the classes are balanced. One result that stands out is DT with SMOTE reaching (0.7995), nearly identical to RF (0.7996) at the same balancing level. This is the only protocol where DT performs this close to an ensemble method with SMOTE, which may reflect MaxProp's more structured traffic patterns giving a single tree enough signal to work with. ADASYN continues its weak showing from Prophet: XGB drops to (0.7310) and KNC to 0.6151, both the lowest ADASYN results across all four protocols. KNC with SMOTENN also records its lowest result here (0.8708 vs 0.8957 in Epidemic), suggesting that even after cleaning, the local neighborhood structure in MaxProp's data remains difficult for a distance-based classifier to navigate. Across all four protocols, the combination of ET and SMOTENN consistently delivers the highest accuracy, answering RQ3 directly, addressing the imbalance is not just beneficial but necessary without it, no classifier in this study rises above (0.60).

DISCUSSION

In this section, we discuss the results obtained for the different approaches to processing unbalanced data, focusing on three main methods: unbalanced, SMOTE and SMOTENN. This is illustrated in Figure 6, FIGURE 7, and FIGURE 8.

When examining the performance of classification algorithms without oversampling, the results show that the results are generally less satisfactory than those obtained with oversampling methods for the four protocols used in this study (see FIGURE 7). This can be attributed to the classifiers tendency to be biased toward the majority class, leading to inaccurate predictions and poor performance for the minority class. However, certain algorithms, such as Extra Trees (ET) and Random Forest (RF), exhibit relatively better performance even without oversampling, due to their robust nature and ability to minimize prediction variance.

The SMOTE oversampling method has shown a significant improvement in the performance of classification algorithms compared to the without oversampling approach for the four studied routing protocols. This improvement is estimated to be 43.24% compared to the results obtained without oversampling (Improvement rate = (New value - Old value) / Old value * 100). SMOTE creates synthetic samples of the minority class by interpolating between existing samples, which allows for a more balanced dataset and improved classifier performance. However, this method can sometimes lead to the creation of artificial samples that do not accurately represent the true characteristics of the minority class, which can result in overfitting. The SMOTE oversampling method has been used successfully in our previous work (Abdellaoui Alaoui et al. 2022, 2023). Building on this, the SMOTENN method further improves classification performance by combining the benefits of SMOTE oversampling with the removal of noisy samples using the KNN algorithm. This approach not only increases the number of samples in the minority class but also removes samples from the majority class that are likely to create classification errors. In our experiments, SMOTENN proved to be the most effective method for improving the performance of classification algorithms, particularly in combination with the Extra Trees (ET) algorithm. We have seen a considerable increase in accuracy after the use of our suggested strategies, going from 81% (SMOTE) to 92% (SMOTENN), representing a 13.58% gain in model performance.

TABLE 7. Performance of classification using Prophet data

		Prophet Protocol				
Classifier	Method	Accuracy	Precision	f1-score	Recall	PT
RF	Without oversampling	0.5292	0.5105	0.5095	0.5033	0.123
	SMOTE	0.8078	0.8054	0.8031	0.8015	0.085
	SVMSMOTE	0.8001	0.8042	0.8030	0.8034	0.055
	SMOTETomek	0.8143	0.8190	0.8178	0.8180	0.238
	SMOTENN	0.8914	0.8852	0.8848	0.8847	0.041
	ADASYN	0.7810	0.7827	0.7766	0.7747	0.056
	KSMOTE	0.7973	0.8005	0.8015	0.8001	0.055
	SMOTEN	0.7978	0.7982	0.7971	0.7974	0.173
	SMOTENC	0.7849	0.7872	0.7861	0.7864	0.071
BC	Without oversampling	0.5300	0.51161	0.51036	0.5040	0.042
	SMOTE	0.8026	0.8000	0.7979	0.7965	0.042
	SVMSMOTE	0.7995	0.8003	0.7994	0.7996	0.043
	SMOTETomek	0.8155	0.8164	0.8152	0.8154	0.041
	SMOTENN	0.8856	0.8847	0.8846	0.8847	0.032
	ADASYN	0.7843	0.7801	0.7851	0.7784	0.055
	KSMOTE	0.7993	0.8001	0.7989	0.7992	0.037
	SMOTEN	0.7967	0.7977	0.7965	0.7968	0.054
	SMOTENC	0.7853	0.7865	0.7850	0.7583	0.040
ET	Without oversampling	0.5318	0.5108	0.5097	0.5036	0.087
	SMOTE	0.7985	0.7968	0.7928	0.7905	0.075
	SVMSMOTE	0.8026	0.8064	0.8051	0.8054	0.059
	SMOTETomek	0.8283	0.8226	0.8212	0.8214	0.062
	SMOTENN	0.8971	0.8956	0.8957	0.8960	0.049
	ADASYN	0.7816	0.7849	0.7796	0.7777	0.110
	KSMOTE	0.8022	0.8042	0.8026	0.8030	0.126
	SMOTEN	0.7928	0.7959	0.7944	0.7948	0.157
	SMOTENC	0.7919	0.7857	0.7847	0.7850	0.071
XGB	Without oversampling	0.53628	0.51290	0.51060	0.4963	0.592
	SMOTE	0.7658	0.7639	0.7581	0.7555	1.050
	SVMSMOTE	0.7596	0.7629	0.7587	0.7597	0.055
	SMOTETomek	0.7892	0.7887	0.7917	0.7892	0.516
	SMOTENN	0.8949	0.8948	0.8939	0.8934	0.476
	ADASYN	0.7456	0.7519	0.7364	0.7347	0.555
	KSMOTE	0.7683	0.7702	0.7678	0.7686	0.592
	SMOTEN	0.7680	0.7682	0.7679	0.7681	0.574
	SMOTENC	0.7582	0.7600	0.7577	0.7583	0.581

continue...

...cont.

		Prophet Protocol				
Classifier	Method	Accuracy	Precision	f1-score	Recall	PT
DT	Without oversampling	0.5174	0.45157	0.45086	0.4510	0.003
	SMOTE	0.7841	0.7802	0.7810	0.7822	0.003
	SVMSMOTE	0.7874	0.7887	0.7878	0.7871	0.002
	SMOTETomek	0.8057	0.8040	0.8028	0.8017	0.002
	SMOTENN	0.8641	0.8752	0.8765	0.8780	0.002
	ADASYN	0.7683	0.7400	0.7434	0.7473	0.003
	KSMOTE	0.7857	0.7828	0.7874	0.7921	0.005
	SMOTEN	0.7830	0.7795	0.7836	0.7880	0.002
	SMOTENC	0.7741	0.7708	0.7729	0.7753	0.002
KNC	Without oversampling	0.5125	0.4413	0.3914	0.4147	0.130
	SMOTE	0.6265	0.6257	0.6245	0.6280	0.123
	SVMSMOTE	0.6502	0.6449	0.6563	0.6683	0.117
	SMOTETomek	0.6702	0.6607	0.6799	0.7004	0.124
	SMOTENN	0.8883	0.8838	0.8989	0.9150	0.12
	ADASYN	0.6037	0.5654	0.5492	0.5342	0.12
	KSMOTE	0.6612	0.6569	0.6658	0.6752	0.148
	SMOTEN	0.6571	0.6532	0.6613	0.6698	0.113
	SMOTENC	0.6292	0.6285	0.6303	0.6324	0.207

TABLE 8. Performance of classification using Maxprop data

		Maxprop Protocol				
Classifier	Method	Accuracy	Precision	f1-score	Recall	PT
RF	Without oversampling	0.5233	0.5172	0.5163	0.5127	0.076
	SMOTE	0.7996	0.7984	0.7966	0.7956	0.055
	SVMSMOTE	0.7852	0.784	0.7837	0.7839	0.075
	SMOTETomek	0.8083	0.8071	0.8064	0.8065	0.049
	SMOTENN	0.8799	0.8825	0.8822	0.8823	0.030
	ADASYN	0.7630	0.7628	0.7587	0.7575	0.047
	KSMOTE	0.7870	0.7890	0.7879	0.7882	0.050
	SMOTEN	0.7854	0.7815	0.7805	0.7807	0.153
	SMOTENC	0.7803	0.7839	0.7833	0.7834	0.044
BC	Without oversampling	0.52445	0.5164	0.5154	0.5116	0.035
	SMOTE	0.7897	0.7886	0.7875	0.7867	0.457
	SVMSMOTE	0.7835	0.7840	0.7833	0.7834	0.034
	SMOTETomek	0.8085	0.8088	0.8084	0.8085	0.029
	SMOTENN	0.8833	0.8826	0.8825	0.8828	0.022
	ADASYN	0.7615	0.7620	0.7575	0.7562	0.041
	KSMOTE	0.7879	0.7887	0.7876	0.7878	0.032
	SMOTEN	0.7825	0.7821	0.7821	0.7823	0.031
	SMOTENC	0.7852	0.7858	0.7850	0.7851	0.033

continue...

...cont.

		Maxprop Protocol				
Classifier	Method	Accuracy	Precision	f1-score	Recall	PT
ET	Without oversampling	0.5236	0.5153	0.5144	0.5104	0.062
	SMOTE	0.7914	0.7910	0.7887	0.7875	0.049
	SVMSMOTE	0.7905	0.7888	0.7881	0.7883	0.042
	SMOTETomek	0.8090	0.8107	0.8100	0.8101	0.043
	SMOTENN	0.8952	0.8877	0.8876	0.8878	0.029
	ADASYN	0.7604	0.7648	0.7608	0.7595	0.076
	KSMOTE	0.7875	0.7879	0.7866	0.7868	0.063
	SMOTEN	0.7823	0.7857	0.7843	0.7846	0.058
	SMOTENC	0.7814	0.7836	0.7827	0.7829	0.045
XGB	Without oversampling	0.5295	0.5212	0.5197	0.5147	0.548
	SMOTE	0.7577	0.7562	0.7552	0.7545	0.457
	SVMSMOTE	0.7451	0.7471	0.7445	0.7452	0.433
	SMOTETomek	0.7773	0.7784	0.7770	0.7773	0.401
	SMOTENN	0.8889	0.8893	0.8876	0.8868	0.358
	ADASYN	0.7310	0.7341	0.7238	0.7227	0.372
	KSMOTE	0.7553	0.7562	0.7549	0.7553	0.372
	SMOTEN	0.7531	0.7533	0.7529	0.7531	0.435
	SMOTENC	0.7504	0.7513	0.7501	0.7504	0.411
DT	Without oversampling	0.5010	0.4620	0.4665	0.4641	0.002
	SMOTE	0.7995	0.7979	0.7982	0.7986	0.002
	SVMSMOTE	0.7787	0.7790	0.7749	0.7710	0.013
	SMOTETomek	0.7931	0.7931	0.7897	0.7865	0.002
	SMOTENN	0.8602	0.8673	0.8713	0.8760	0.002
	ADASYN	0.7570	0.7338	0.7373	0.7410	0.002
	KSMOTE	0.7801	0.7803	0.7785	0.7770	0.002
	SMOTEN	0.7733	0.7739	0.7735	0.7733	0.001
	SMOTENC	0.7708	0.7748	0.7678	0.7613	0.002
KNC	Without oversampling	0.5170	0.4770	0.4476	0.4616	0.085
	SMOTE	0.6253	0.6242	0.6241	0.6251	0.076
	SVMSMOTE	0.6468	0.6420	0.6524	0.6633	0.122
	SMOTETomek	0.6667	0.6614	0.6718	0.6828	0.131
	SMOTENN	0.8708	0.8690	0.8821	0.8958	0.123
	ADASYN	0.6151	0.5835	0.5692	0.5562	0.136
	KSMOTE	0.6435	0.6404	0.6470	0.6539	0.081
	SMOTEN	0.6472	0.6435	0.6512	0.6592	0.129
	SMOTENC	0.6285	0.6280	0.6292	0.6307	0.0868

To assess how consistently each balancing method performs across all six classifiers and all four protocols, FIGURE 7 plots the mean accuracy together with the min–max range observed across the six classifiers for each method. Two observations stand out immediately. First, SMOTE produces a wide spread at every protocol (half-range of ±0.083 to ±0.091), reflecting its sensitivity to classifier choice. Second, SMOTENN narrows this spread dramatically (±0.013 to ±0.019) while simultaneously raising the mean, confirming that its cleaning step removes

the noisy boundary samples that cause classifiers to disagree.

The balancing methods tested here do not all perform equally, and the pattern across the four protocols is consistent enough to draw some clear distinctions. SMOTENN stands apart from the rest, it is the only method that reliably pushes accuracy above 0.86 regardless of which classifier is used or which protocol the data comes from. SMOTE, SMOTETomek, SVMSMOTE, and KSMOTE generally land between 0.78 and 0.84 for RF, BC, ET, and DT, though KNC consistently falls below this range with the same methods, rarely exceeding 0.70 with SMOTE alone, and XGB also trends lower than the other ensemble methods, particularly under SMOTE. ADASYN is a different case, it holds its own on Epidemic and Spray and Wait, but drops off considerably on Prophet and MaxProp, where the class boundary is less clean and its tendency to pile synthetic samples onto the hardest minority instances ends up making things worse rather than better.

On the classifier side, ET and XGB are the steadiest options when paired with SMOTENN, with ET recording the highest minimum accuracy across all protocols (0.8952) and XGB showing the lowest variation between protocols. KNC tells the most dramatic story in the data, it goes from below 0.56 without any balancing to above 0.87 with SMOTENN, a range wider than anything seen with the other classifiers. DT never reaches the same peaks as the ensemble methods, but it gains consistently from balancing and stays a reasonable choice in situations where understanding the model's decisions matters as much as its accuracy.

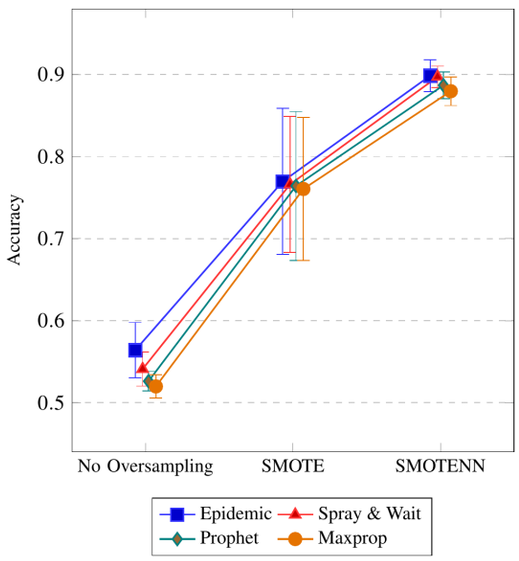


FIGURE 7. Mean accuracy with min–max range across all classifiers for three balancing methods over the four DTN protocols.

FIGURE 8 complements this picture by showing the F1-score achieved by each classifier under SMOTENN across all four protocols. F1-score is reported here rather than accuracy because it accounts for the residual class imbalance that remains even after balancing, making it a more informative measure of model quality in this setting.

The heatmap confirms the trends described above: ET and KNC achieve the highest F1-scores consistently, while DT records the lowest absolute values and KNC shows the greatest variation across protocols. The color gradient also reveals a systematic downward shift from Epidemic to Maxprop, reflecting the increasing routing complexity of the protocols rather than any failure of the balancing method.

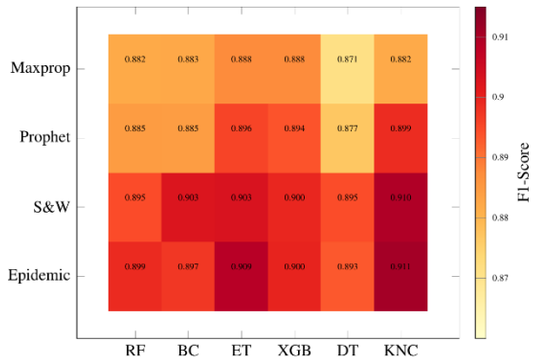


FIGURE 8. F1-scores of all classifiers under SMOTENN across the four DTN protocols.

For anyone looking to apply these findings in datasets. practice, ET with SMOTENN is the combination to start with when working on ML-based DTN routing with imbalanced data. When speed or simplicity is a priority, SMOTE with RF or BC gives solid results with less overhead. ADASYN is worth avoiding in protocols like Prophet and MaxProp where the data is already noisy near the class boundary.

Going back to the three questions this study set out to answer: yes, balancing methods work for bundle classification in DTN routing (RQ1). Among those tested, SMOTENN is consistently the strongest across all protocols and classifiers (RQ2). Finally, The impact of balancing is substantial: without it, no classifier in this study crosses 0.60, while with SMOTENN the same classifiers reach above 0.86 (RQ3).

In summary, the findings of this study demonstrate that data balancing plays a crucial role in improving the performance of classification algorithms for delay-tolerant networks (DTN). Oversampling methods, particularly SMOTE and SMOTENN, significantly improved classification performance in terms of accuracy, precision, F1-score, and recall. The SMOTENN approach combined

with the Extra Trees (ET) algorithm proved to be the most effective in handling imbalanced data in DTN routing scenarios.

CONCLUSION AND FUTURE SCOPE

Class imbalance represents one of the persistent challenges in data mining and machine learning. Addressing this problem requires careful data analysis and the application of appropriate balancing strategies. Several approaches exist for addressing this problem at the data level, including undersampling, oversampling, and feature selection. This paper addressed a major challenge encountered by supervised learning methods in many real-world applications, namely class imbalance, where at least one class is under-represented relative to the others. This is particularly evident in network domains, where models are needed to classify arriving and non-arriving bundles. Class imbalance poses a significant constraint on standard learning algorithms, which tend to perform well on majority classes while underperforming on minority classes, the latter often being of greater practical importance. A method based on the combination of machine learning algorithms and data balancing techniques was proposed, demonstrating improved classification performance on imbalanced DTN protocol data.As a future direction, more advanced data balancing techniques will be investigated to further improve bundle classification performance on imbalanced datasets.

ACKNOWLEDGEMENT

This research was supported by Universiti Kebangsaan Malaysia.

DECLARATION OF COMPETING INTEREST

None.

REFERENCES

Abdellaoui Alaoui, E., Zekkori, H. & Agoujil, S. 2019. Hybrid delay tolerant network routing protocol for heterogeneous networks. *Journal of Network and Computer Applications* 148.

Abdellaoui Alaoui, E.A., Nassiri, K., Koumétio Tékouabou, S.C. & Agoujil, S. 2023. Explainable prediction ofa˘intelligent dtn routing.Y. Farhaoui, A. Rocha, Z. Brahmia & B. Bhushab (eds.) *Artificial Intelligence and Smart Environment*, pp. 415–420. Cham: Springer International Publishing.

Abdellaoui Alaoui, E.A., Tekouabou, S.C.K., Maleh, Y. & Nayyar, A. 2022. Towards to intelligent routing for dtn protocols using machine learning techniques. *Simulation Modelling Practice and Theory* 117: 102475.

Aditsania, A., Adiwijaya & Saonard, A.L. 2017. Handling imbalanced data in churn prediction using adasyn and backpropagation algorithm. L. Riza, A. Pranolo, A. Wibawa, E. Junaeti, Y. Wihardi, U. Hashim, S. Horng, R. Drezewski, H. Lim, G . Chakraborty, L. Hernandez & S. Nazir (eds.) *2017 3RD INTERNATIONAL CONFERENCE ON SCIENCE IN INFORMATION TECHNOLOGY (ICSITECH)*, pp. 533–536.

Ahrenholz, J., Goff, T. & Adamson, B. 2011. Integration of the CORE and EMANE network emulators. *Proceedings - IEEE Military Communications Conference MILCOM*, pp. 1870–1875.

Bajpai, S. & Chauhan, A. 2022. Evolution of machine learning techniques for optimizing delay tolerant routing. *Proceedings - 2022 4th International Conference on Advances in Computing, Communication Control and Networking, ICAC3N 2022* pp. 294–299.

Balasubramanian, A., Levine, B. & Venkataramani, A. 2007. Dtn routing as a resource allocation problem. *ACM SIGCOMM Computer Communication Review*, vol. 37, pp. 373–384. ACM.

Banyal, S., Bharadwaj, K.K., Sharma, D.K., Khanna, A. & Rodrigues, J.J. 2021. Hilser: Hierarchical learning-based sectionalised routing paradigm for pervasive communication and resource efficiency in opportunistic iot network. *Sustainable Computing: Informatics and Systems* 30.

Batista, G.E.A.P.A., Prati, R.C. & Monard, M.C. 2004. A study of the behavior of several methods for balancing machine learning training data. *SIGKDD Explor. Newsl.* 6(1): 2029.

Bhavani, A., Ramana, A.V. & Chakravarthy, A. 2023. A review on machine learning based routing protocols for delay tolerant networks. *Intelligent Decision Technologies* (Preprint): 1–11.

Borah, S.J., Dhurandher, S.K., Woungang, I. & Kumar,V. 2017. A game theoretic context-based routing protocol for opportunistic networks in an iot scenario. *Computer Networks* 129: 572–584. Special Issue on 5G Wireless Networks for IoT and Body Sensors.

Branco, P., Torgo, L. & Ribeiro, R.P. 2016. A survey of predictive modeling on im balanced domains. *ACM COMPUTING SURVEYS* 49(2).

Breiman, L. 1996. Bagging predictors. *Machine learning* 24(2): 123–140.

Breiman, L. 2001. Random forests. *Machine Learning* doi:10.1023/A:1010933404324.

Burgess, J., Gallagher, B., Jensen, D. & Levine, B.N. 2006. Maxprop: Routing for vehicle-based disruption-tolerant networks. *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications,* pp. 1–11.

Carneiro, T., Medeiros Da NóBrega, R.V., Nepomuceno, T., Bian, G.B., De Albuquerque,V.H.C. & Filho, P.P.R. 2018. Performance analysis of google colaboratory as a tool for accelerating deep learning applications. *IEEE Access* 6: 61677–61685.

Chawla, N.V., Bowyer, K.W., Hall, L.O. & Kegelmeyer,W.P. 2002. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research* 16: 321–357.

Chen, T. & Guestrin, C. 2016. Xgboost: A scalable tree boosting system. Proceedings of the 22nd *acm sigkdd international conference on knowledge discovery and data mining*, pp. 785–794.

Chourasia, V., Pandey, S. & Kumar, S. 2021. Optimizing the performance of vehicular delay tolerant networks using multi-objective PSO and artificial intelligence. *Computer Communications* 177: 10–23.

Cunningham, P. & Delany, S.J. 2021. k-nearest neighbour classifiers: 2nd edition. *ACM Computing Surveys* doi:10.1145/3459665.

Czarnowski, I. 2022. Weighted ensemble with one-class classification and over-sampling and instance selection (wecoi): An approach for learning from imbalanced data streams. *Journal of Computational Science* 61: 1877–7503.

Dai, Q., Liu, J.W. & Liu, Y. 2022. Multi-granularity relabeled under-sampling algorithm for imbalanced data. *Applied Soft Computing* 124: 109083.

Dalal, R. & Khari, M. 2022. Peculiar effectual approach: Q-routing in opportunistic network.S.Bhaumik,S. Chattopadhyay, T. Chattopadhyay &S. Bhattacharya (eds.) *Proceedings of International Conference on Industrial Instrumentation and Control*, pp. 609–615. Singapore: Springer Nature Singapore.

Ding, H., Chen, L., Dong, L., Fu, Z. & Cui, X. 2022. Imbalanced data classification: A knn and generative adversarial networks-based hybrid approach for intrusion detection. *Future Generation Computer Systems* 131: 240–254.

Doering, M., Lahde, S., Morgenroth, J. & Wolf, L. 2008. IBR-DTN: An efficient implementation for embedded systems. *Proceedings of the Annual International Conference on Mobile Computing and Networking, MOBICOM*, pp. 117–119. New York, New York, USA: ACM Press.

Douzas, G., Bacao, F. & Last, F. 2018. Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. *INFORMATION SCIENCES* 465: 1–20.

Dudukovich, R., Clark, G. & Papachristou, C. 2019. Evaluation of classifier complexity for delay tolerant network routing. *2019 IEEE Cognitive Communications for Aerospace Applications Workshop (CCAAW)*, pp. 1–7. IEEE.

Dudukovich, R., Hylton, A. & Papachristou, C. 2017. A machine learning concept for dtn routing. *2017 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, pp. 110–115.

Faris, H., Abukhurma, R., Almanaseer, W., Saadeh, M., Mora, A.M., Castillo, P.A. & Aljarah, I. 2020. Improving financial bankruptcy prediction in a highly imbalanced class distribution using oversampling and ensemble learning: a case from the spanish market. *Progress in Artificial Intelligence* 9(1): 31–53.

Friedman, J.H. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* pp. 1189–1232.

Galán-Jiménez, J., Berrocal, J., Garcia-Alonso, J. & Azabal, M.J. 2019. A novel routing scheme for creating opportunistic context-virtual networks in iot scenarios. *Sensors* 19(8).

García, V., Sánchez, J., Marqués, A., Florencia, R. & Rivera, G. 2020. Understanding the apparent superiority of over-sampling through an analysis of local information for class-imbalanced data. *Expert Systems with Applications* 158: 113026.

Garg, P., Dixit, A. & Sethi, P. 2021. Ml-fresh: Novel routing protocol in opportunistic networks using machine learning. *Computer Systems Science and Engineering* 40: 703–717.

Geurts, P., Ernst, D. & Wehenkel, L. 2006. Extremely randomized trees. *Machine learning* 63(1): 3–42.

G.S., T., Hariprasad, Y., Iyengar, S., Sunitha, N., Badrinath, P. & Chennupati, S. 2022a. An extension of synthetic minority oversampling technique based on kalman filter for imbalanced datasets. *Machine Learning with Applications* 8: 100267.

G.S., T., Hariprasad, Y., Iyengar, S., Sunitha, N., Badrinath, P. & Chennupati, S. 2022b. An extension of synthetic minority oversampling technique based on kalman filter for imbalanced datasets. *Machine Learning with Applications* 8: 100267.

Gu, Q., Tian, J., Li, X. & Jiang, S. 2022. A novel random forest integrated model for imbalanced data classification problem. *Knowledge-Based Systems* 250: 109050.

Guo, Y., Gou, G., Xiong, G., Jiang, M., Shi, J. & Xia, W. 2021. Let imbalance have nowhere to hide: Class-sensitive feature extraction for imbalanced traffic classification. *2021 INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS (IJCNN),* IEEE International Joint Conference on Neural Networks (IJCNN). Int Neural Network Soc; IEEE Computat Intelligence Soc. International Joint Conference on Neural Networks (IJCNN), ELECTR NETWORK, JUL 18-22, 2021.

Harkavy, E. & Net, M.S. 2020. Utilizing reinforcement learning to autonomously mange buffers in a delay tolerant network node. *2020 IEEE Aerospace Conference*, pp. 1–8.

He, H., Bai, Y., Garcia, E.A. & Li, S. 2008. Adasyn: Adaptive synthetic sampling approach for imbalanced learning. *2008 IEEE INTERNATIONAL JOINT CONFERENCE ON NEURAL NETWORKS,VOLS 1-8*, IEEE International Joint Conference on Neural Networks (IJCNN), pp. 1322–1328. IEEE. International Joint Conference on Neural Networks, Hong Kong, PEOPLES R CHINA, JUN 01-08, 2008.

Ijaz, M.F., Attique, M. & Son, Y. 2020. Data-driven cervical cancer prediction model with outlier detection and over-sampling methods. *SENSORS* 20(10).

Islam, M.T. & Mustafa, H.A. 2023. Multi-layer hybrid (mlh) balancing technique: A combined approach to remove data imbalance. *Data Knowledge Engineering* 143: 102105.

Jindal, A., Dua, A., Kaur, K., Singh, M., Kumar, N. & Mishra, S. 2016. Decision tree and svm-based data analytics for theft detection in smart grid. *IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS* 12(3): 1005–1016.

Kandhoul, N. & Dhurandher, S.K. 2022. Deep q learning based secure routing approach for oppiot networks. *Internet of Things (Netherlands)* 20.

Karami, A. & Derakhshanfard, N. 2020. Rprtd: Routing protocol based on remaining time to encounter nodes with destination node in delay tolerant network using artificial neural network. *Peer-to-Peer Networking and Applications* pp. 1–17.

Kim, K. 2021. Normalized class coherence change-based knn for classification of imbalanced data. *Pattern Recognition* 120: 108126.

Kotsiantis, S.B. 2007. Supervised machine learning: A review of classification techniques. *Informatica* 31(3): 249–268.

Kumar, B.S., Vishnubhatla, S., Babu, C.M. & Shetty, S.P. 2023. Performance analysis of prophet routing protocol in delay tolerant network by using machine learning models. *IJACSA) International Journal of Advanced Computer Science and Applications* 14.

Kumar, S.P.A., Banyal, S., Bhardwaj, K.K., Thakur, H.K. & Sharma, D.K. 2022. Distributed probability density based multi-objective routing for opp-iot networks enabled by machine learning. *Journal of Intelligent and Fuzzy Systems* 42: 1199–1211.

Le, T. 2021. Multi-hop routing under short contact in delay tolerant networks. *Computer Communications* 165(September 2020): 1–8.

Li, W., Galluccio, L., Bassi, F. & Kieffer, M. 2017. Distributed faulty node detection in delay tolerant networks: Design and analysis. *IEEE Transactions on Mobile Computing* 17(4): 831–844.

Liang, H., Shang, Y. & Wang, S. 2021. Study on dtn routing protocol of vehicle ad hoc network based on machine learning. *Wireless Communications and Mobile Computing* 2021: 7965093.

Lim, L.P. & Mahinderjit Singh, M. 2020. Resolving the imbalance issue in short messaging service spam dataset using cost-sensitive techniques. *Journal of Information Security and Applications* 54: 102558.

Lindgren, A., Doria, A. & Schelen, O. 2004. Probabilistic routing in intermittently connected networks. *Service assurance with partial and intermittent resources*, pp. 239–254. Springer.

Ling, Y., Zhang, R., Cen, M., Wang, X. & Jiang, M. 2021. Cost-sensitive heterogeneous integration for credit card fraud detection. pp. 750–757. IEEE; IEEE Comp Soc; Shenyang Aerosp Univ.

Mahbooba, B., Timilsina, M., Sahal, R. & Serrano,M. 2021. Explainable artificial intelligence (xai) to enhance trust management in intrusion detection systems using decision tree model. *COMPLEXITY* 2021.

Mao, Y., Zhou, C., Ling, Y. & Lloret, J. 2019. An optimized probabilistic delay tolerant network (dtn) routing protocol based on scheduling mechanism for internet of things (iot). *Sensors* 19(2): 243.

Mochizuki, D., Abiko, Y., Saito, T., Ikeda, D. & Mineno, H. 2019. Delay-tolerance-based mobile data offloading using deep reinforcement learning. *Sensors (Switzerland)* 19(7).

More, A.S. & Rana, D.P. 2022. Performance enrichment through parameter tuning of random forest classification for imbalanced data applications. *Materials Today: Proceedings* pp. 3585–3593.

Nguyen, H.M., Cooper, E.W. & Kamei, K. 2011. Borderline over-sampling for imbalanced data classification. *International Journal of Knowledge Engineering and Soft Data Paradigms* doi:10.1504/IJKESDP.2011.039875.

Pereira, P.R., Casaca, A., Rodrigues, J.J.P.C., Soares, V.N.G.J., Triay, J. & Cervello-Pastor, C. 2012. From delay-tolerant networks to vehicular delay-tolerant networks. *IEEE Communications Surveys Tutorials* 14(4): 1166–1182.

Petrides, G. & Verbeke, W. 2022. Cost-sensitive ensemble learning: a unifying framework. *DATA MINING AND KNOWLEDGE DISCOVERY* 36(1): 1–28.

Ren, J., Wang, Y., Cheung, Y.M., Gao, X.Z. & Guo,X. 2023. Grouping-based oversampling in kernel space for imbalanced data classification. *Pattern Recognition* 133: 108992.

Roy, A., Cruz, R.M., Sabourin, R. & Cavalcanti, G.D. 2018. A study on combining dynamic selection and data preprocessing for imbalance learning. *Neurocomputing* 286: 179–192.

Saez, J.A., Luengo, J., Stefanowski, J. & Herrera, F. 2015. Smote-ipf: Addressing the noisy and borderline examples problem in imbalanced

classification by a re-sampling method with filtering. *INFORMATION SCIENCES* 291: 184–203.

Schonlau, M. & Zou, R.Y. 2020. The random forest algorithm for statistical learning. *The Stata Journal* doi:10.1177/1536867X20909688.

Segundo, F.R., e Silva, E.S. & Farines, J.M. 2016. A dtn routing strategy based on neural networks for urban bus transportation system. *Journal of Network and Computer Applications* 64: 216–228.

Seyhan, K., Nguyen, T.N., Akleylek, S., Cengiz, K. & Islam, S.K. 2021. Bi-GISIS KE: Modified key exchange protocol with reusable keys for IoT security. *Journal of Information Security and Applications* 58(March): 102788.

Shaikhina, T., Lowe, D., Daga, S., Briggs, D., Higgins, R. & Khovanova, N. 2019. Decision tree and random forest models for outcome prediction in antibody incompatible kidney transplantation. *BIOMEDICAL SIGNAL PROCESSING AND CONTROL* 52: 456–462.

Sharma, D.K., Dhurandher, S.K., Woungang, I., Srivastava, R.K., Mohananey, A. & Rodrigues, J.J. 2016. A machine learning-based protocol for efficient routing in opportunistic networks. *IEEE Systems Journal* 12(3): 2207–2213.

Sharma, D.K., Kukreja, D., Aggarwal, P., Kaur, M. & Sachan, A. 2018. Poisson's probability-based Q-Routing techniques for message forwarding in opportunistic networks. *International Journal of Communication Systems* 31(11): 1–23.

Sharma, D.K., Rodrigues, J.J., Vashishth, V., Khanna, A. & Chhabra, A. 2020. RLProph: a dynamic programming based reinforcement learning approach for optimal routing in opportunistic IoT networks. *Wireless Networks* 26(6): 4319–4338.

Singh, A.K., Pamula, R., Jain, P.K. & Srivastava, G. 2021. An efficient vehicular-relay selection scheme for vehicular communication. *Soft Computing* .

Singh, A.V., Juyal, V. & Saggar, R. 2017. Trust based intelligent routing algorithm for delay tolerant network using artificial neural network. *Wireless Networks* 23(3): 693–702.

Soares, V.N., Rodrigues, J.J. & Farahmand, F. 2014. Geospray: A geographic routing protocol for vehicular delay-tolerant networks. *Information Fusion* 15: 102–113. Special Issue: Resource Constrained Networks.

Spyropoulos, T., Psounis, K. & Raghavendra, C.S. 2005. Spray and wait: an efficient routing scheme for intermittently connected mobile networks. *Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking*, pp. 252–259. ACM.

Sun, J., Lang, J., Fujita, H. & Li, H. 2018. Imbalanced enterprise credit evaluation with dte-sbd: Decision tree ensemble based on smote and bagging with differentiated sampling rates. *INFORMATION SCIENCES* 425: 76–91.

Sun, L., Li, M., Ding, W., Zhang, E., Mu, X. & Xu, J. 2022. Afnfs: Adaptive fuzzy neighborhood-based feature selection with adaptive synthetic over-sampling for imbalanced data. *Information Sciences* 612: 724–744.

Sun, X., Yu, Q., Wang, R., Zhang, Q., Wei, Z., Hu, J. & Vasilakos, A.V. 2013. Performance of dtn protocols in space communications. *Wireless networks* 19(8): 2029–2047.

Tang, Y., Zhang, Y.Q., Chawla, N.V. & Krasser,S. 2009. Svms modeling for highly imbalanced classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 39(1): 281–288.

Tibshirani, R.J. & Efron, B. 1993. An introduction to the bootstrap. *Monographs on statistics and applied probability* 57(1).

Tripathi, A., Mekathoti, V.K., Waghulde, I., Patel, K. & Balasubramanian, N. 2021. Neural network aided optimal routing with node classification for adhoc wireless network. *CEUR Workshop Proceedings* 2889: 43–54.

Uchida, N., Hashimoto, R., Sato, G. & Shibata, Y. 2019. *Adaptive Array Antenna Controls with Machine Learning Based Image Recognition for Vehicle to Vehicle Networks*, vol. 22. Springer International Publishing.

Vahdat, A., Becker, D. et al. 2000. Epidemic routing for partially connected ad hoc networks.

Vashishth, V., Chhabra, A. & Sharma, D.K. 2019. A Machine Learning Approach Using Classifier Cascades for Optimal Routing in Opportunistic Internet of Things Networks. *Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks workshops* 2019-June: 1–9.

Wongvorachan, T., He, S. & Bulut, O. 2023. A comparison of undersampling, oversampling, and smote methods for dealing with imbalanced classification in educational data mining. *Information (Switzerland)* 14: 54.

Xiaolong, X.U., Wen, C. & Yanfei, S. 2019. Over-sampling algorithm for imbalanced data classification. *Journal of Systems Engineering and Electronics* 30: 1182–1191.

Xie, X., Liu, H., Zeng, S., Lin, L. & Li, W. 2021.A novel progressively undersampling method based on the density peaks sequence for imbalanced data. *Knowledge-Based Systems* 213: 106689.

Xu, Y., Zhang, Y., Zhao, J., Yang, Z. & Pan, X. 2019. Knn-based maximum margin and minimum volume hyper-sphere machine for imbalanced data classification. *International Journal of Machine Learning and Cybernetics* 10: 357–368.

Xu, Z., Shen, D., Nie, T. & Kou, Y. 2020. A hybrid sampling algorithm combining m-smote and enn based on random forest for medical imbalanced data. *Journal of Biomedical Informatics* 107.

Yen, S.J. & Lee, Y.S. 2009. Cluster-based under-sampling approaches for imbalanced data distributions. *Expert Systems with Applications* 36: 5718–5727.

Yu, L., Zhou, R., Tang, L. & Chen, R. 2018. A dbn-based resampling svm ensemble learning paradigm for credit classification with imbalanced data. *Applied Soft Computing* 69: 192–202.

Zang, B., Huang, R., Wang, L., Chen, J., Tian, F. & Wei, X. 2017. An improved knn algorithm based on minority class distribution for imbalanced dataset. pp. 696–700. Institute of Electrical and Electronics Engineers Inc.

Zeraatkar, S. & Afsari, F. 2021. Intervalvalued fuzzy and intuitionistic fuzzyknn for imbalanced data classification. *Expert Systems with Applications* 184: 957–4174.

Zguira, Y., Rivano, H. & Meddeb, A. 2018. Iob-dtn: A lightweight dtn protocol for mobile iot applications to smart bike sharing systems. *2018 Wireless Days (WD)*, pp. 131–136. IEEE.

Zhang, A., Yu, H., Huan, Z., Yang, X., Zheng, S. & Gao,S. 2022. Smote-rknn: A hybrid re-sampling method based on smote and reverse k-nearest neighbors. *Information Sciences* 595: 70–88.

Zhao, Z., Zhong, P. & Zhao, Y. 2011. Learning svm with weighted maximum margin criterion for classification of imbalanced data. *Mathematical and Computer Modelling* 54: 1093–1099.

Zheng, M., Li, T., Zheng, X., Yu, Q., Chen, C., Zhou, D., Lv, C. & Yang, W. 2021. Uffdfr: Undersampling framework with denoising, fuzzy c-means clustering, and representative sample selection for imbalanced data classification. *INFORMATION SCIENCES* 576: 658–680.