IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 25 October 2022, accepted 14 December 2022, date of publication 21 December 2022,
date of current version 28 December 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3231384

RESEARCH ARTICLE

Efficient Detection of Spam Over Internet Telephony by Machine Learning Algorithms

LADISLAV BEHAN[1], JAN ROZHON[1], JAKUB SAFARIK[2], FILIP REZAC[1], AND MIROSLAV VOZNAK[1], (Senior Member, IEEE)

[1]Department of Telecommunications, Faculty of Electrical Engineering and Computer Science, VSB—Technical University of Ostrava, 70800 Ostrava, Czechia
[2]IT4Innovations National Supercomputing Center, VSB—Technical University of Ostrava, 70800 Ostrava, Czechia

Corresponding author: Ladislav Behan (ladislav.behan@vsb.cz)

This work was supported in part by the European Regional Development Fund under the project AI and Reasoning under Grant CZ.02.1.01/0.0/0.0/15 003/0000466; in part by the Czech Ministry of Education, Youth and Sports within project under Grant SP2022/5; and in part by the VSB—Technical University of Ostrava through the Large Infrastructures for Research, Experimental Development and Innovations Project ''e-Infrastructure CZ'' under Grant LM2018140.

ABSTRACT Recent trends show a growing interest in VoIP services and indicate that guaranteeing security in VoIP services and preventing hacker communities from attacking telecommunication solutions is a challenging task. Spam over Internet Telephony (SPIT) is a type of attack which is a significant detriment to the user's experience. A number of techniques have been produced to detect SPIT calls. We reviewed these techniques and have proposed a new approach for quick, efficient and highly accurate detection of SPIT calls using neural networks and novel call parameters. The performance of this system was compared to other state-of-art machine learning algorithms on a real-world dataset, which has been published online and is publicly available. The results of the study demonstrated that new parameters may help improve the effectiveness and accuracy of applied machine learning algorithms. The study explored the entire process of designing a SPIT detection algorithm, including data collection and processing, defining suitable parameters, and final evaluation of machine learning models.

INDEX TERMS Data mining, machine learning, neural network, SIP, spam, SPIT, VoIP.

I. INTRODUCTION

As network technology progresses, internal IP telephony services are becoming increasingly used in many commercial operations. The growing interest in VoIP services also has a parallel increase in interest from hackers. Although researchers are persistently exploring different security tools and improving security systems to protect VoIP infrastructure, occasionally services are not protected from attackers, who are often one step ahead in detecting security vulnerabilities.

With low calling prices and VoIP technology which often includes free open-source tools (e.g. Asterisk), Spam over Internet Telephony (SPIT) is an increasingly attractive VoIP attack for spammers. Numerous methods have been proposed to recognise SPIT calls, some which distinguish the spam call even before the called party has answered. These methods are based on statistical analysis of call parameters such as the originating IP address or data from signalling and media messages. However, few of these methods have been researched from the perspective of applying modern artificial intelligence algorithms to SPIT detection.

The main aim of the current study was to apply algorithms to aid in detecting SPIT users. Many studies which investigated similar methods substantiated their research on artificially generated data ([19], [23], [24], [36], [41]) rather than analysis of real-world data ([26], [27]). The current study applied only real-world data collected from multiple sources (VoIP providers, honeypots, mobile devices and online data services) [21], [22], [33].

A key advantage of applying artificial neural networks (ANN) in SPIT detection is the capability of the ANN to learn and model non-linear and complex associations from a diverse range of inputs and outputs. After initial learning,

The associate editor coordinating the review of this manuscript and approving it for publication was Jose Saldana.

ANNs can make assumptions about hidden relationships from unseen data. The ANN is also able to generalise the model and make predictions from unseen data. In contrast to many other prediction methods, neural networks do not limit the input attributes or how they should be distributed.

By applying various statistical procedures and processing real-world datasets, we searched for suitable SPIT parameters that potentially improved the SPIT detection algorithm's accuracy and performance efficiency. The current study presents new parameters which aid in distinguishing telecommunications chain users with a high degree of accuracy. Applying various proven methods, we then identified the most suitable and computationally efficient machine learning (ML) SPIT detection model. The study's innovation is in processing data from genuine traffic and its investigation of different algorithms from a performance perspective with a variety of input parameters.

The paper is organized as follows: Chapter 2 describes the state of the art in the current research and contains an overview of the approaches used to distinguish SPIT calls from regular calls; Chapter 3 briefly describes the motivation for the study; Chapter 4 explores the main aim of applying innovative artificial intelligence methods to detect SPIT callers in Internet telephony; Chapter 5 concludes the paper.

TABLE 1. List of abbreviations.

Abbreviation	Definition
AI	Artificial Intelligence
ANN	Artificial Neural Networks
AUC	Area Under The Curve
CDR	Call Detail Records
CV	Cross-validation
EPMG	Enhanced Progressive Multi Grey-Levelling
ISFC	Incoming Single Failed Calls
ISSC	Incoming Single Successful Calls
LTD	Loose Tie Detection
MFIF	Multiple calls (First Incoming Failed)
MFIS	Multiple calls (First Incoming Successful)
MFOF	Multiple calls (First Outgoing Failed)
MFOS	Multiple calls (First Outgoing Successful)
ML	Machine Learning
NN	Neural Networks
OSFC	Outgoing Single Failed Calls
OSSC	Outgoing Single Successful Calls
PMG	Progressive Multi Grey-levelling
RTP	Real-time Transport Protocol
SHAP	Shapley Addi- tive explanation
SPIT	Spam Over Internet Telephony
SVM	Support Vector Machines
VoIP	Voice over Internet Protocol

II. STATE OF THE ART

Detecting SPIT with the expectation that VoIP users demand quick replies to their call requests is a challenging task. Researchers have presented different techniques to detect and distinguish voice spam calls from regular calls. Chikha et al. [1] classified these techniques according to list-based filtering, reputation-based filtering, Turing test, and pattern-based filtering.

A. LIST-BASED FILTERING

List-based filtering identifies SPIT by applying blacklist, whitelist and greylist filtering lists. Shin et al. [3] proposed progressive multi grey-levelling (PMG), an algorithm which calculates a caller's grey level. The algorithm continuously computes the 'grey level' of a caller by assessing previous call patterns and decides whether the call is established or blocked. Jinyoung et al. [4] investigated the PMG algorithm's operation in different call patterns and fine tuned the detection algorithm to block SPIT more reliably. Falomi et al. [6] defined a simulator which classifies SPIT callers according to their call rate, identity and call history details. Mathieu et al. [5] proposed a framework for classifying traffic analytically by calculating call quantity and duration.

Blacklists cannot be viewed as a SPIT countermeasure. Both server-side and client-side blacklists are very inefficient with Direct IP SPIT for many reasons. Server-side blacklists can be bypassed by Direct IP SPIT because SIP messages are forwarded directly to the client. Client-side blacklists are circumvented by Direct IP SPIT because the caller is able to use numerous identities for calling [7].

B. REPUTATION-BASED FILTERING

Reputation-based filtering allows users to rate other users according to assumed patterns to estimate a trust value for detecting potential SPIT calls [8]. The resulting score is determined from a buddy list and the user's evaluations of buddies. In related research, Patankar et al. [9] suggested using buddy lists in VoIP infrastructure to build trust chains between call participants. The authors claimed that call duration has a crucial role in classifying SPIT calls. Bokharaei et al. [12] proposed and integrated two new techniques: Loose Tie Detection (LTD), which creates a list based on SPIT parameters such as the duration of outgoing calls, and Enhanced Progressive Multi Grey-Leveling (EPMG), which creates a list that classifies outliers based on call density.

A potential attacker can bypass reputation-based detection mechanism systems in the same manner as blacklists [13]. A caller with a negative reputation may be seen as globally blacklisted when their call attempts are blocked. However, a regular blacklisted user needs to increase their 'plus points' to obtain a new 'clean' shield.

C. TURING TEST

The Turing test is able to identify intelligent behaviour equivalent to or indistinguishable from human behaviour. A reverse

Turing test known as CAPTCHA is a procedure managed by the PC to recognise human speech from the generated SPIT voice recording. Quittek et al. [14] applied two Turing test methods which used modelled human conversation patterns to distinguish calls originating from humans and classes of calls generated by bots. Turing tests are very useful, but they also have some weaknesses. A SPIT caller may pay some cheaper operators to break the Audio CAPTCHA. By using Turing tests, user interaction is required in every call session, and therefore, different capabilities in computational power are required at the VoIP endpoint to solve computational puzzles. If the task is difficult to solve, it will delay establishing the call session. This method is inefficient and counter-productive and frustrates regular users in the telecommunications chain.

D. BEHAVIOUR-BASED FILTERING

Other SPIT detection mechanisms analyse SIP message patterns or voice data processed through the the Real-time Transport Protocol (RTP). MacIntosh and Vinokurov [17] applied a signalling analysis technique which depends on two statistical aspects, one being that spam calls are unidirectional, the other being that most of the time legitimate users terminate call sessions. Wu et al. [18] proposed a mitigation scheme by processing different call variables parsed from the SIP protocol. They improved a simplified MPCK-Means algorithm for semi-supervised learning by adapting it to a massive number of calls. Barghi et al. [19] applied a model which is dependent on cooperation with the service provider and the called party, and thus the call lists update themselves over time. Hongchang et al. [20] recommended a combined three-layered SPIT detection scheme which collects signal characteristics; the proposed mechanism gathers signal data before the call is answered.

SIP messages carry caller information crucial to detecting SPIT. Successfully revealing spam before establishing the call improves the user experience and reduces the network resources used to manage SPIT calls. However, this method is able to only correctly classify SPIT calls after at least ten calls. In VoIP systems, changing the number is trivial and gives potential attackers an easy way to circumvent this detection method.

Even though many researchers have proposed highly accurate and efficient SPIT detection methods, as technologies and algorithms develop, the SPIT problem still appears to need more research to improve detection mechanisms. The number of SPIT calls is continuing to rise rapidly, and the need for innovative, high-performance detection mechanisms is even greater.

According to social network spam filtering surveys, Support Vector Machine, Decision Tree, Naive Bayes, and Random Forest are algorithms which perform well in the spam detection process [38]. To achieve high accuracy levels, certain ensemble algorithms such as Random Forest, integrate the predictions of other ML algorithms. Yet surprisingly, an examination of the current literature on SPIT reveals that little research has been done in relation to NN algorithms. The current evidence suggests that NN algorithms are able to achieve high accuracy in detecting e-mail and SMS spam [39]. Improved optimization convergence and resistance to overfitting may be credited to this phenomenon [40]. In the current study, we propose neural network models which take advantage of these qualities for SPIT filtering.

III. MOTIVATION

The previous chapter reviewed the different techniques which can be applied to detect SPIT calls. In an investigation of the literature, we recognised that the public community lacks two essential factors: first, no tool is currently available for detecting attacks on VoIP infrastructure, a threat which will continue to develop and adapt to the dynamic needs of network security; second, no detailed studies on the use of modern neural networks in SPIT detection are available, and to the best of our knowledge, it also appears that no recent scientific attention has been given to this issue. A point in case is that most of the published research has worked with artificially generated data. The current study processed only real-world data collected from users around the world.

Our research has investigated all of the above-mentioned features. The following section describes how we collected and processed datasets from different sources. Suitable parameters for SPIT detection were defined to recognise the behavioural patterns of both SPIT callers and regular users. The paper guides through the entire process of integrating the resulting parameters into ML algorithms up to the final evaluation of the performance and effectiveness of each algorithm and the different types and numbers of input parameters (neurons).

The current study differs from other recently published papers in four main areas:

- The study applies only real-world data.
- A practical analysis is given for determining the NN hyper-parameters which produce a robust and reliable SPIT detection model.
- The application of neural networks is examined in detail in relation to the issue, and the effectiveness against other ML algorithms is analysed.
- Innovative parameters for SPIT detection are delivered.

The next chapter provides details of the datasets and data collection process for the study. Statistical methods were used to identify different network user characteristics and the parameters to distinguish them.

IV. PROPOSED SPIT DETECTION MECHANISM USING AI

SPIT calls and regular user behavioural patterns were analysed by processing the data with data mining methods and statistical procedures. From this data, various inputs for ML algorithms were obtained, tested and compared to determine the most suitable parameters for identifying SPIT callers.

A. STATISTICAL ANALYSIS OF THE COLLECTED DATA

Because users in the telecommunications chain operate according to specific behavioural patterns, it is essential to examine various datasets in detail and distinguish their features. Call detail records (CDR) contain useful information which aid in defining a particular user's behaviour. A report containing CDR records displays data which is listed according to user or phone number. This type of file is generated for a single user (number) and exhibits certain metrics such as call volume, call duration, timestamp, source and destination numbers.

Storing or collecting CDR records today, however, is not a simple matter, and in some circumstances, nearly impossible because of the GDPR. We did not want to study and process randomly generated data, therefore we needed a means to collect real-life data. The solution was to directly request call logs from friends, family members, co-workers and other people enthusiastic about supporting our research. Every mobile device stores call logs which can be extracted with suitable software. The mobile phone operating systems on the range of devices available today store phone records in different formats.

TABLE 2. Android call log items.

Key	Type	Meaning
number	String	Phone number as the user entered it.
date	Date	Date of call.
time	Timestamp	Time of call
type	Integer	Call type (incoming, outgoing, missed).
name	String	Name in the contact list
new	Binary	1 – new number, 0 – in contact list
dur	Integer	Call duration

Android phones store their records in XML, whereas iPhones record data as CSV. An overview of each type of CDR record is given in Table 2 (Android) and Table 3 (iOS). Although mobile phone records contain up to seven items, the CDR records from iOS phones contain only five items. All of these records were pre-processed and converted individually into JSON format.

TABLE 3. iOS call log items.

Key	Type	Meaning
Call type	String	Incoming or Outgoing
Date	Date	Date of call
Number	String	Phone number as the user entered it
Contact	Boolean	Name in the contact list (True, False)
Duration	Time	Call duration

Overall, we succeeded in obtaining 36 anonymous datasets containing 26,328 phone calls from people in eight countries.

We also acquired Some extensive datasets from a source online and processed data from the research project conducted at the University of Strathclyde (UoS). The CDR records database contains call logs that were pulled from 27 devices, totalling 13,035 records. Finally, we succeeded in processing a set of CDR records from 342 users who made a total of 33,726 calls over one month. These datasets are summarised in Table 4,

TABLE 4. Datasets used in the study.

Dataset	Number of Users	Call Count
Mobile users (self-gathered)	36	26,328
University of Strathclyde [21]	27	13,035
Telefónica [22]	342	33,726
VoIP provider 1	36	109,776
VoIP provider 2	125	596,855
Total	566	779,720

Despite the complications of obtaining a VoIP provider's data, we acquired a dataset which contains CDR records of 109,776 telephone calls from 47 telemarketers for May 2018, and another dataset which contains 596,855 records from 125 operators logged from February to May in 2019.

All data were processed into individual user profiles and subjected to a statistical analysis. Table 5 shows a summary of the user behaviour patterns in the telecommunications chain over a period of one day.

The analysis revealed that the average telemarketer called 43.41 times more often than an average user. In terms of average incoming call count, this number represents only 5% of a telemarketer's calls.

Generally, telemarketers attempt to sell as many products as possible, therefore the number of their outgoing calls represents such a high proportion of total calls (95%) in the dataset. For regular phone users, the ratio of incoming/outgoing calls was relatively balanced, incoming calls constituting 49% of the total number of calls and the remainder of the calls representing outgoing calls (51%).

Another difference between the behaviours of telemarketers and regular users was in the ratio of answered and missed calls. The chance that a phone marketer succeeded with an outgoing call was only 51%, whereas a regular user's chance was 93.5%. Considering this difference in accepting incoming calls, we can conclude that regular users had a higher likelihood of answering the phone (94.1%) than a SPIT user (37.5%). Call recipients were more amenable to responding to a regular user (92.9%) than a telemarketer (52%).

Another critical parameter in an individual user's behaviour was the number of unique phone numbers each user called. The difference in the processed values is clear, and SPIT users were characterized by a large number (93.5%) of unique numbers called. Regarding call length, the average telemarketer's call duration was 1.42 times shorter (93.45 s) than the average user's call time.

We then defined the number of calls according to certain time intervals of the day, assuming that specific user calls in the call centre data would have occurred at certain times of the day according to working hours.

TABLE 5. Daily average calling behaviour in users.

Investigated parameter	SPIT	Regular
Call count	180.58	4.16
Incoming calls	9.62 (5%)	2.04 (49%)
Outgoing calls	171.94 (95%)	2.12 (51%)
Answered calls	92.25 (51%)	3.89 (93.5%)
Not answered calls	88.33 (49%)	0.27 (6.5%)
Incoming answered	3.61 (37.5%)	1.92 (94.1%)
Outgoing answered	89.06 (52%)	1.97 (92.9%)
Incoming not answered	6.01 (62.5%)	0.12 (0.06%)
Outgoing not answered	82.89 (48.2%)	0.15 (0.07%)
Average duration	93.45 s	132.40 s
Incoming calls average duration	89.71 s	135.27 s
Outgoing calls average duration	97.19 s	129.53 s
Incoming calls unique numbers	122.17 (6.5%)	18.25 (49.6%)
Outgoing calls unique numbers	1768.22 (93.5%)	18.57 (50.4%)

The resulting statistics (Figure 1) clearly show that the average telemarketer initiated the majority of calls from 6 AM to 6 PM, implying a long, twelve-hour working shift. After 6 PM, the call count converged to zero, indicating the end of the shift. A slight decrease in the call count occurred around noon, possibly representing a lunch break. The data also indicates that individual employee profiles differ. By processing the user's working hours, we can conclude whether the data represents a part-time or a full-time worker.

As in the above example, we assumed that callers behaved differently during weekdays and weekends. Figure 2 indicates the resulting statistical analysis for all weekdays. The proportion of calls did not fluctuate on weekdays, and the proportion of calls placed by a phone vendor at the weekend was just 0.4%. An analysis of a regular user's processed data indicated that the weekend call count represented 17% of all calls.

Figure 3 shows that an increase in the number of calls not answered at weekends, suggesting that call centre employees did not attend work. Throughout the week, the likelihood of an answered call was 49.19%, whereas at the weekend, the likelihood was just 8.3%.

By including the statistics for incoming and outgoing calls, the behaviour in SPIT users highlighted a sharp contrast between initiating calls and a low number of receiving calls. The proportions in regular users, however, were generally equal.

Another important parameter for analysing the behaviour of telephone users is the duration of calls. Regular users generally called the same numbers several times, receiving callbacks from the same numbers or for the purpose of creating a telecommunications relationship. In these parameters, a regular user's call duration should be longer than that of a telemarketer.

SPIT users tended to call large numbers of users but did not create telecommunications relationships. The exception was acquired customers, in which case repeated calls were made and the average call duration was shorter. Call length was also governed by many called parties immediately hanging up the phone down after learning that a telemarketer was calling them.

The statistical analysis in Table 6 summarises the average call duration at each hourly interval. Regular users were active throughout the day. The period of the day with the shortest calls was between 6 AM and 9 AM, with an average call lasting 77.87 seconds. The period of the day with the longest calls was in the evening between 9 PM and 12 AM, with an average call time of 187.77 seconds.

TABLE 6. Comparison of average duration in hours.

	SPIT			Regular		
Time	Dur.	Inc.	Out.	Dur	Inc.	Out.
00:00–05:59	0	0	0	119.2	145.4	93.1
06:00–08:59	66.8	52.7	80.9	77.8	96.6	59.1
09:00–11:59	84.5	80.8	88.1	125.9	132.7	119.3
12:00–14:59	83.7	75.8	91.7	117.3	124.1	110.5
15:00–17:59	82.8	82.8	82.7	127.9	144.7	111.2
18:00–20:59	164.5	220.4	108.5	164.9	183.5	146.4
21:00–23:59	0	0	0	187.8	204.2	171.3

In terms of SPIT users, the table illustrates a different story. As mentioned above, the data reflect user working hours. From 9 PM to 6 AM, SPIT users were not active. The shortest average call duration (66.77 seconds) occurred between 6 and 9 AM, and the average call time peaked between 6 PM and 9 PM, mainly as a result of the low frequency of calls made during this part of the day.

B. NEW SPIT CLASSIFICATION PARAMETERS

We also statistically analysed the behaviour of both user types according to the adage "Tell me who your friends are, and I will tell you who you are" and highlighted the top ten persons with whom the user maintained frequent contact.

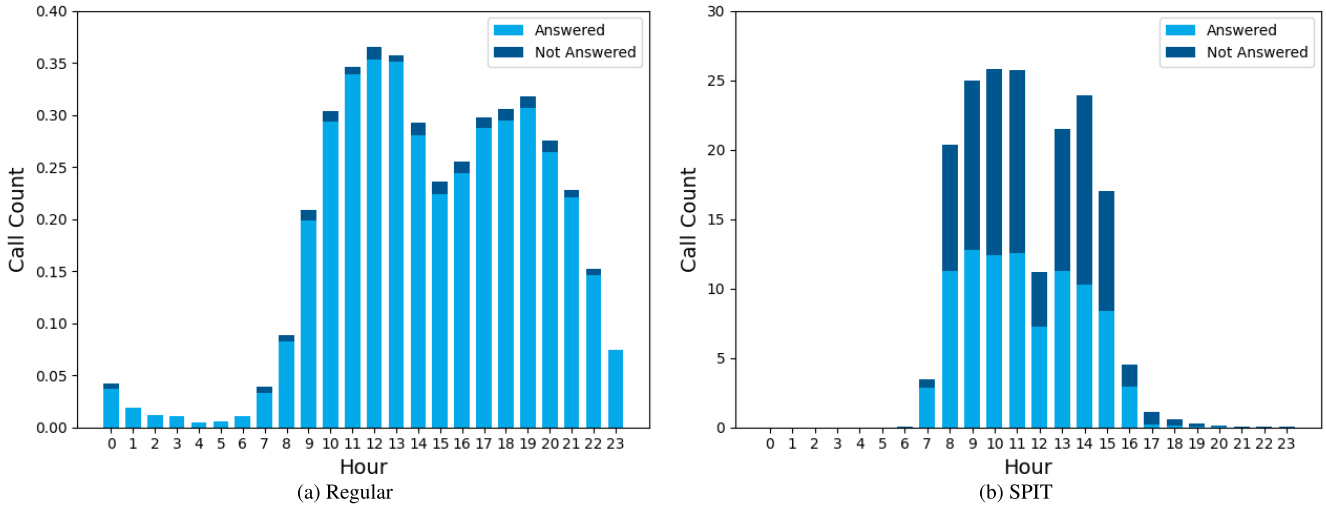
The processed data in Table 6 indicates that incoming calls received by telemarketers accounted for 6.3% of all calls from their top ten contacts. The remaining majority (93.7%) consisted of outgoing calls. Although SPIT users called these contacts most often, the proportion of answered calls was only 45.3%, lasting on average 103.03 seconds.

The results for regular users showed major differences from the results for SPIT users: the respective proportions of incoming and outgoing calls accounted for 45.6% and 54.4% of all calls, and the answered call rate was 82.6%, with an average call duration of 135.34 seconds.

TABLE 7. Top ten contacts.

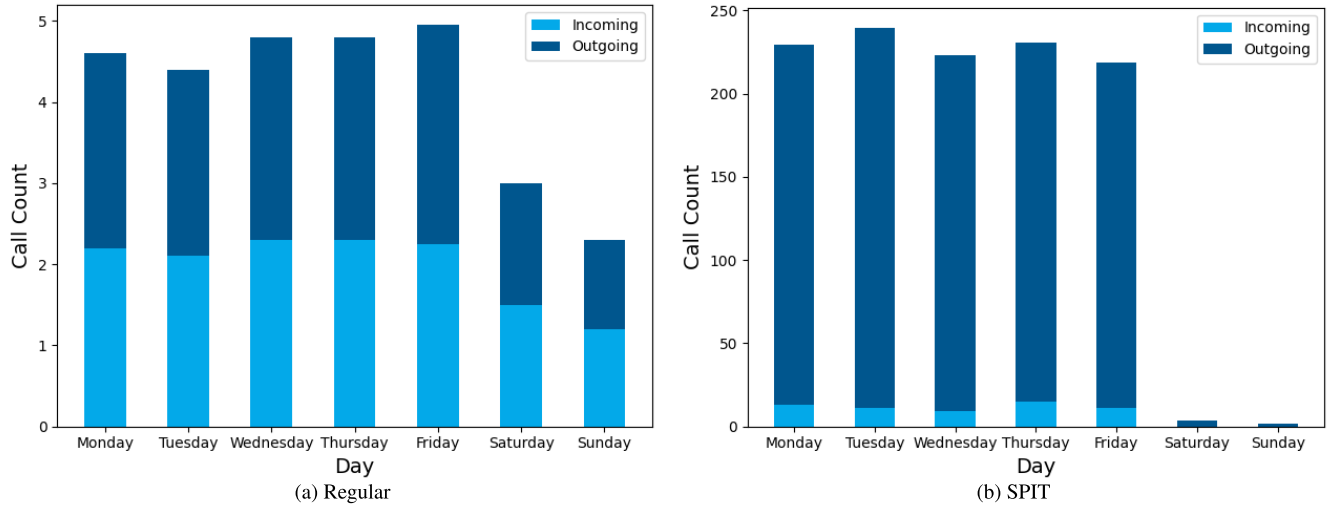
Investigated variable	SPIT	Regular
Total Calls	344.65	141.92
Incoming calls	20.74 (6.3%)	64.70 (45.6%)
Outgoing calls	323.91 (93.7%)	77.22 (54.4%)
Answered calls	155.97 (45.3%)	117.29 (82.6%)
Not answered calls	188.68 (55.7%)	24.63 (17.4%)
Average duration	103.03 s	135.34 s
Relationship index	0.09	0.82

The final item (i.e., Relations index) in Table 7 represents the proportion of calls to the top ten contacts in the total calls



(a) Regular (b) SPIT

FIGURE 1. Average call counts for specific time intervals.



(a) Regular (b) SPIT

FIGURE 2. Incoming/Outgoing calls.

per user. The results indicate that telemarketers maintained telecommunications relationships with only 9% of contacts, while regular users maintained relationships with 82% of contacts.

Table 8 lists new parameters which describe how the relationships between users commenced and how their contact continued according to the processed data. We identified and defined eight ways how these relationships developed or terminated after the first call:

- Outgoing single failed calls (OSFC) – a call was made but it was not answered. Neither party initiated another call.

$$osfc = \frac{\sum outgoing\_single\_failure\_calls}{\sum unique\_numbers} \quad (1)$$

- Outgoing single successful calls (OSSC) – a call was made and it succeeded. Neither party initiated another call.

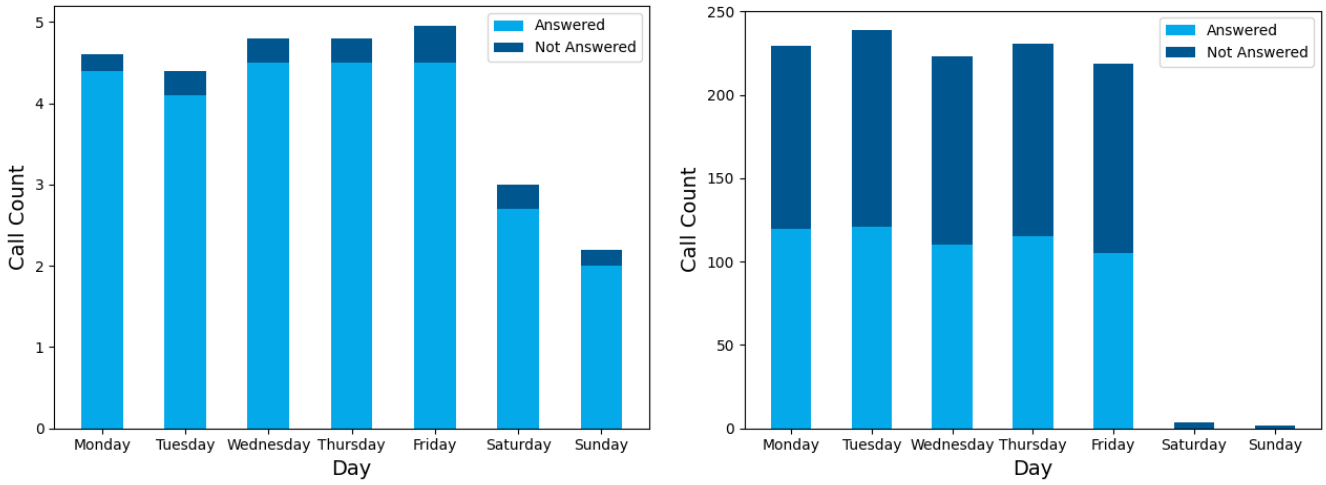
$$ossc = \frac{\sum outgoing\_single\_success\_calls}{\sum unique\_numbers} \quad (2)$$

- Multiple calls (first outgoing failed) (MFOF) – the first outgoing call failed but several other calls were made later.

$$mfof = \frac{\sum first\_outgoing\_failure}{\sum unique\_numbers} \quad (3)$$

- Multiple calls (first outgoing successful) (MFOS) – the first outgoing call succeeded and several other calls were made later.

$$mfos = \frac{\sum first\_outgoing\_success}{\sum unique\_numbers} \quad (4)$$



(a) Regular (b) SPIT

FIGURE 3. Answered/Not Answered calls.

- Incoming single failed calls (ISFC) – the first incoming call failed. Neither party initiated another call.

$$isfc = \frac{\sum incoming\_single\_failure\_calls}{\sum unique\_numbers} \quad (5)$$

- Incoming single successful calls (ISSC) – the first incoming call succeeded. Neither party initiated another call.

$$issc = \frac{\sum incoming\_single\_success\_calls}{\sum unique\_numbers} \quad (6)$$

- Multiple calls (first incoming failed) (MFIF) – the first incoming call failed but several other calls were made later.

$$mfif = \frac{\sum first\_incoming\_failure}{\sum unique\_numbers} \quad (7)$$

- Multiple calls (first incoming successful) (MFIS) – the first incoming call succeeded and several other calls were made later.

$$mfis = \frac{\sum first\_incoming\_success}{\sum unique\_numbers} \quad (8)$$

Each of the calculated values shows differences, but the ISFC and ISSC values were zero for both user types and therefore not applied to the classification algorithms.

The other parameters were applied if they were statistically significant in identifying SPIT users in the telecommunications VoIP network.

The subsequently modelled ML algorithms generally filter any redundant features by applying statistical hypothesis testing methods, e.g., Gini Index or the Chi-Squared test.

With some insight into the character of the data, we applied statistical methods to evaluate the significance of the input parameters. Gini Index indicated the parameter weights; the results of this test are summarised in Figure 4.

TABLE 8. Initial call relationships.

Investigated variable	SPIT	Regular
Outgoing		
Outgoing single failed calls	0.2505	0.0214
Outgoing single successful calls	0.3498	0.2546
Multiple calls (first outgoing failed)	0.2208	0.0301
Multiple calls (first outgoing successful)	0.1641	0.3241
Incoming		
Incoming single failed calls	0	0
Incoming single successful calls	0	0
Multiple calls (first incoming failed)	0.0084	0.3512
Multiple calls (first incoming successful)	0.0064	0.0190

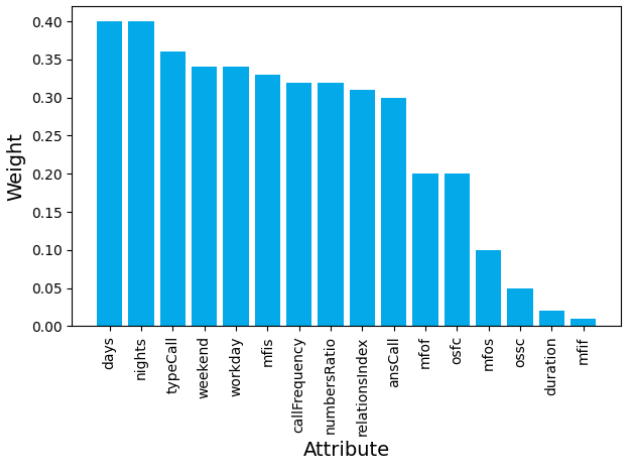


FIGURE 4. Significance of features according to Gini Index.

This chapter discussed several behavioural parameters obtained by applying statistical methods to real-world datasets. These parameters allow regular users and SPIT users to be distinguished in the telecommunications chain. In addition to the parameters studied in other publications, several

innovative parameters were also investigated (Fig. 4). These are discussed in detail in the next chapter. Whether inputting these parameters into various machine learning algorithms and tuning their hyper-parameters improves the efficiency and performance of the detection models is also examined.

C. APPLICATION OF ML ALGORITHMS TO DETECT SPIT

Once they were defined, the relevant parameters were input into previously researched classification algorithms [2], [4], [17], [27] which are able to detect anomalies and threats in telecommunications networks.

The algorithms employed in the current study were selected for their classification accuracy, training and testing overheads, and overall effectiveness. Based on a thorough empirical study, the following algorithms proved suitable candidates for SPIT detection:

- Naïve Bayes, Logistic Regression (Linear ML algorithms)
- K-nearest neighbour, Decision Tree (Non-linear ML algorithms)
- Random Forest (Ensemble ML algorithms)
- Neural Networks (Deep Learning algorithms)

Training for each classification model used a validation dataset to verify the neural network's learning and to avoid overfitting. When ML algorithms learn and test the parameters of a prediction function on identical data, it is considered a methodological error; this type of model is extremely accurate, but its predictions would fail with new unseen data.

The size of the dataset required the use of a cross-validation (CV) statistical method to estimate the accuracies of the ML models on unseen data. This method separated the data into k independent folds, with $k - 1$ folds employed to train the network and the remaining fold used for testing. This process continued until all the folds were used at least once as a test set. The network's final output was then calculated by averaging the accuracy attained from each test set.

In the following sub-section, we examine the results of the ML algorithms using a naive approach, which means all parameters are applied as input to the individual algorithms without any pre-processing.

1) CONSTRUCTION OF A NAIVE ML MODEL

Before experimental application, the algorithms were optimized to deliver the best possible accuracy. This allowed the use of different algorithms to produce sets of results which were all comparable to the proposed neural network. The optimiser applied a grid search method similar to the neural network's hyper-parameter optimisation method described in Chapter 4.

A simple or random prediction is often made using a naive classifier model. The model is unsophisticated and is referred to as naive because it generates predictions without any previous learning or domain expertise.

The performance of a baseline classifier on a classification task determines the lower limit of predicted performance in all other models. A classification model possesses some skill if, for example, it outperforms a naive classifier. A classifier model lacks competence if it performs more poorly than the naive classifier.

Table 9 summarises the resultant accuracies of the ML models which used the naive dataset and applied the 16 parameters (Fig. 4) described in the previous chapter as input for the ML algorithms.

TABLE 9. Accuracies of ML algorithms applied to SPIT detection (naive approach in all parameters).

Models	TPR	FPR	PRE	SPE	ACC	AUC
SVM	0.98	0	1.00	1.00	0.98	1.00
Decision Tree	0.96	0.01	0.97	0.99	0.97	0.97
Random Forest	1.00	0	1.00	1.00	1.00	1.00
Naive Bayes	0.96	0.01	0.99	0.99	0.98	0.99
Logistic Reg.	0.99	0.01	0.99	0.99	0.99	0.99
Neural Network	0.99	0.01	0.99	0.99	0.98	0.99

TPR – True Positive Rate, FPR – False Positive Rate, PRE – Precision, SPE – Specificity, ACC – Accuracy, AUC – Area Under the Curve

Logistic Regression (ACC 99%) and Random Forest (ACC 100%) achieved the best accuracies. The neural network achieved a similar accuracy of 99%, while the other algorithms produced slightly lower accuracies of 98% and 96%. The neural network is discussed in the next chapter.

The time that the ML algorithm spends on training and scoring will determine whether it is applied to the dataset. If the algorithm's training process takes too long, remodelling the classification model by applying another dataset in combination with hyper-parameter optimisation may be complicated and time-consuming and lead to excluding that algorithm from the candidate list.

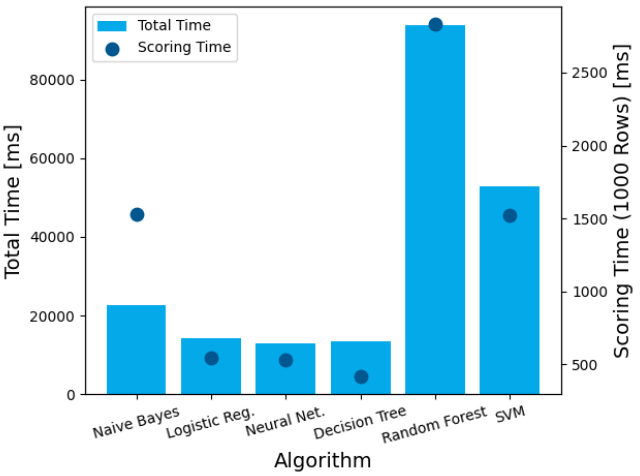


FIGURE 5. Naive approach ML vs. neural network.

Figure 5 presents a comparison of ML algorithm performance and indicates that Logistic Regression and Neural Network were the most powerful and efficient algorithms in applying the naive approach.

2) COMPARISON WITH OTHER PUBLISHED STUDIES

Nazih et al. [23] proposed a detection mechanism based on the SVM classifier. Using the SIP-Msg-Gen tool, the authors performed an experiment on generated data and achieved one hundred percent accuracy.

Chikha et al. [24] used several ML algorithms (Naive Bayes, Bagging, AdaBoost, and Multilayer Perceptron) to detect SPIT. However, the authors reported only the resulting AUC for each algorithm and did not examine the individual algorithms in detail. The results indicate AUC values of 0.93 for Naive Bayes and 1.0 for AdaBoost. The data applied in the study was simulated.

Azad et al. [26] analysed the anonymised CDRs of more than two million users in a real-world dataset gathered over a day by an anonymous telecommunications provider. This interesting work applied the Decision Tree algorithm and examined the features of user behaviour in individual telecommunications networks. However, little can be learned from the work about the classifier's resulting success or the metrics that the study applied. The study identified 250 SPIT users from the processed data, 50 call centre users, and the remainder as legitimate users.

Nassar et al. [27] presented a monitoring scheme which used the SVM algorithm for classification. The scheme continuously monitored 38 characteristics in signalled time slices and applied those attributes as input to the classification algorithms, achieving an accuracy of 98.9% in SPIT detection.

D. APPLICATION OF NEURAL NETWORK ALGORITHMS TO SPIT

This section introduces the neural network (NN) which used the given dataset. A neural network applying all the input parameters described above was constructed. We then attempted to reduce the number of input parameters to decrease the algorithm's load on computing power.

Various possible settings for the neural network's hyper-parameters and the least number of neurons required for highly accurate classification were explored.

First, we examined integration of the new input parameters into the NN algorithms and determined whether it could obtain similar accuracy in classifying users as the algorithms in the previous chapter with pre-processed data. The other parameters have already been sufficiently addressed by other studies, and an overview is given in Table 10.

The process of finding a suitable topology consists of the following steps:

- Selecting features (using Gini Index)
- Optimising hyper-parameters
- Training the NN
- Validating the NN
- Comparing the results with other solutions

1) HYPER-PARAMETER OPTIMIZATION

Optimising the hyper-parameters in a ML algorithm is a type of optimisation problem which first involves finding a set of hyper-parameters and then investigating and determining a suitable combination to either minimise the model's loss or maximise the model's accuracy.

TABLE 10. SPIT parameters applied in other studies.

Feature	P1	P2	P3	P4	P5	P6	P7	P8
Answered calls						•		
Failed calls		•						
Call frequency	•	•	•	•	•	•	•	•
Call duration	•	•			•	•	•	
Outgoing numbers	•		•			•	•	
Incoming numbers			•		•		•	•
Incoming/outgoing				•				
Days						•		•
Hours						•		•

P1 – Toyoda et al. [34], P2 – Chikha et al. [24], P3 – Su M.-Y. et al. [25], P4 – Azad et al. [26], P5 – Liu et al. [13], P6 – Batthalla et al. [36], P7 – Vennila et al. [37], P8 – Barghi et al. [19]

To achieve an optimal classification result, a suitable topology must be defined in terms of the number of hidden layers and their neurons. An insufficient number of neurons in individual layers may lead to inadequate training of the network. Too many neurons, however, may lead to overfitting the network and generalising the neural network model to its patterns so that it is unable to predict new values [2].

For the purposes of the NN model presented here, minimum batch size (MBS), learning coefficients, momenta, L2 regularisation effects, error function types, input data pre-processing, neural network initialisation, number of learning epochs, and minimum classification errors all required specification. These factors interact, and altering any one of them may result in an inevitable change in others. Generally, many different optimum solutions with various parameter combinations can be found.

A random selection of internal weights between neurons and neuron biases complicates any assessment of the effectiveness of changing each parameter. The aim in this part of the study was to identify the best parameters which approximate an ideal classifier for training the neural network while consuming minimal system resources.

Gradient search, population-based search, and racing algorithms are some of the methods used to optimise hyper-parameters [31]. Grid search, which is a method of examining all potential hyper-parameter configurations within a range specified by the user [31], is the current standard for hyper-parameter optimization in ML.

Table 11 presents an overview of all parameters used as input for the Grid Search algorithm.

From the resulting 100,000+ neural network combinations, we selected the top five which achieved the best accuracy (98.2–98.4) (Table 12).

The next step was an analysis of the resulting NN models. We again applied the cross-validation (CV) method to estimate the NN models' accuracies on unseen data. If the result from only one iteration of the k-fold CV is considered, it is rather skewed and repeated observations are therefore necessary. Repeated k-fold cross-validation can help minimise the

TABLE 11. Grid Search hyper-parameter settings.

Hyper-parameters	Settings
Number of hidden layers	0–3 (Step 1)
Number of neurons	1–6 (Step 1)
Momentum (MMT)	0.0–1.0 (Step 0.1)
Batch sizes (BS)	8–64 (Step 8)
Number of epochs	150–250 (Step 10)
Learning Rate (LR)	0.001–0.9 (Step 0.001)
Loss functions (LF)	Mean Squared Error (MSE) Crossentropy (CE) Hinge
Activation functions (AF)	Sigmoid Tanh ReLu

TABLE 12. Best performing neural network models and their configurations.

ID	MMT	BS	AF	L1	L2	LF	LR	ACC
1	0.4	8	tanh	5	3	MSE	0.04	98.4
2	0.2	32	relu	6	5	MSE	0.007	98.4
3	0.3	8	sigm	5	-	MSE	0.2	98.4
4	0.3	64	sigm	4	-	MSE	0.2	98.4
5	0.2	16	sigm	6	3	MSE	0.1	98.2

ID – Neural Network ID, MMT – Momentum, BS – Batch size, AF – Activation function, L1 – Layer 1 number of neurons, L2 – Layer 2 number of neurons, LF – Loss function, LR – Learning rate, ACC – Accuracy

inaccuracy in the mean model performance estimate. Each observation was therefore repeated two to ten times, and the resulting accuracy (ACC) means were calculated across all folds from all runs.

The results in Table 13 clearly indicate that the most promising model was the third (ID 3), which achieved the highest and most stable accuracy in all the k-fold CV results.

TABLE 13. Repeated k-Fold accuracy results.

	Number of folds						
ID	2	3	4	5	10	15	20
1	95.37	95.81	95.79	95.53	96.30	96.17	96.22
2	89.41	87.48	88.45	85.29	90.73	89.29	92.83
3	**95.39**	**95.53**	**96.07**	**95.93**	**96.34**	**96.52**	**96.57**
4	71.61	76.89	83.14	86.25	88.78	86.86	91.55
5	94.62	94.85	95.07	95.21	95.35	95.39	95.44

We then investigated how the resulting NN model performed and compared its effectiveness to other ML algorithms whose input layers also consisted of only the seven new parameters introduced in the current study.

The results (Fig. 6) indicate that the NN model was one of the best performing and most efficient. The next step involved reducing the number of input parameters with the aim of achieving similar or even better accuracy/performance results.

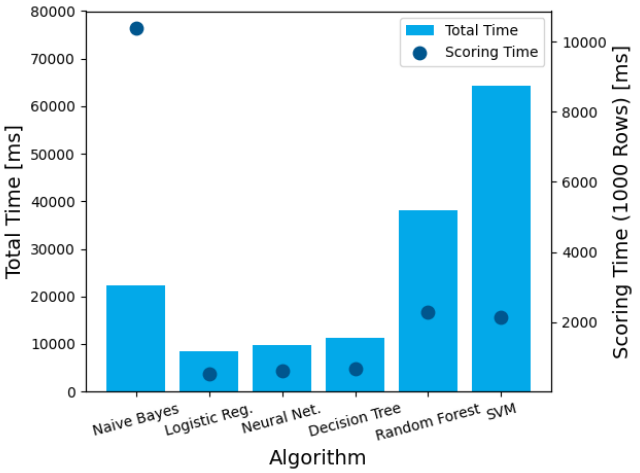


FIGURE 6. ML algorithm comparison (reduced set with seven parameters).

2) REDUCTION OF INPUT PARAMETERS

The first step in obtaining a suitable NN network model was to examine the different models and reduce the number of input parameters according to the weights obtained from Gini Index weight calculation. A higher weight represents a more relevant attribute for neural network classification.

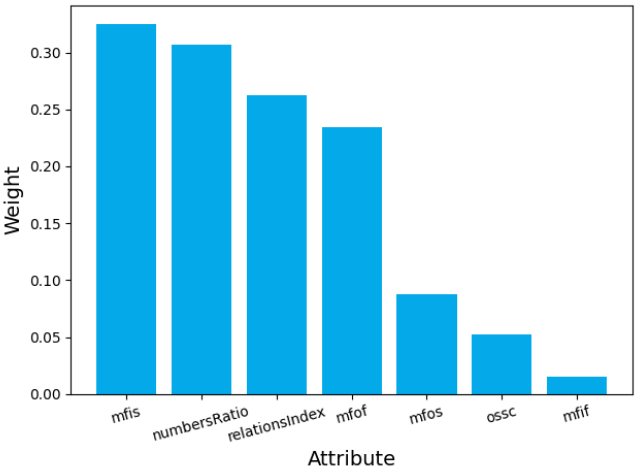


FIGURE 7. Gini index of selected features in the NN.

The number of input parameters in the naive model could be reduced to seven parameters by dropping redundancies and keeping the parameters with high Gini impurity values (Fig. 7). When all seven input parameters were applied to the neural network input, it was able to retain high accuracy. The number of input parameters was also reduced to five and still achieved high accuracy.

In the real world, certain situations may occur when not all parameters can be used in practice; for example, when the trained model is deployed over a limited time frame or limited

hardware is available for training, or the model needs to run quickly on older mobile devices without internet connections. Other technical factors which depend on the individual situation must therefore be considered concurrently with the resulting accuracy. In these situations, it would be preferable to have a model which has fewer input neurons and produces highly accurate results at the required performance level.

According to the weights obtained from the Gini Index calculation (Fig. 7), we selected five of the parameters used as input for the ML algorithms. The k-fold CV method was then reapplied to estimate the model's accuracy.

The accuracy values in Table 14 resulted from applying the k-fold CV method; each ML algorithm was executed ten times, and the resulting accuracy means were calculated across all folds in all runs.

TABLE 14. Evaluation of ML algorithms using five input parameters.

Models	TPR	FPR	PRE	SPE	ACC	AUC
SVM	0.95	0.05	0.92	0.96	0.96	0.97
Decision Tree	0.94	0.06	0.96	0.98	0.97	0.96
Random Forest	0.94	0.06	0.98	0.99	0.97	0.99
Naive Bayes	0.85	0.15	0.93	0.97	0.94	0.98
Logistic Reg.	0.94	0.06	0.94	0.97	0.96	0.98
Neural network	0.95	0.02	0.97	0.98	0.97	0.97

TPR – True Positive Rate, FPR – False Positive Rate, PRE – Precision, SPE – Specificity, ACC – Accuracy, AUC – Area Under the Curve

The results clearly indicate that using the top five (based on the Gini Index) input parameters produced the best results, with an accuracy of 97%.

To determine the most suitable NN model, we used the Grid Search algorithm with the same settings as in Table 11. The results are summarised in Table 15 and revealed that certain NN models were highly accurate even with input layers consisting of only five neurons.

TABLE 15. Results for the top five NN models.

ID	NP	MMT	BS	AF	L1	L2	LF	LR	ACC
1	5	0.9	16	tanh	3	2	MSE	0.05	0.97
2	5	0.8	25	tanh	3	3	MSE	0.1	0.97
3	5	0.5	25	tanh	3	1	MSE	0.1	0.97
4	3	0.8	16	tanh	1	1	MSE	0.1	0.96
5	3	0.5	16	tanh	2	1	MSE	0.3	0.96

PN – Number of parameters, MMT – Momentum, BS – Batch size, AF – Activation function, L1 – Layer 1 neurons number, L2 – Layer 2 neurons number, LF – Loss function, LR – Learning rate, ACC – Accuracy

From these NN models, we once again needed to select the most stable model which produced the most stable accuracies from all k-fold cross-validations. The results summarised in Table 16 indicate that all the models obtained highly accurate, stable values, but on average the best results were achieved by the model with ID 1. All the results hereinafter indicate the performance of this particular model.

The CV results in Table 13 indicate that the model with ID 1 produced the the highest and most stable accuracy of all the k-fold cross-validations.

TABLE 16. Accuracy of the neural network model according to the number of k-fold cross-validations.

	Number of folds						
ID	2	3	4	5	10	15	20
1	**95.84**	**95.66**	**95.84**	**96.02**	**96.03**	**95.84**	**96.01**
2	95.30	95.12	95.66	95.12	95.47	95.49	95.12
3	95.48	94.94	96.20	96.02	96.03	94.94	95.30
4	95.48	95.12	95.66	95.12	95.12	96.03	95.84
5	95.66	95.30	95.12	95.48	95.66	95.48	95.48

The NN model's performance and effectiveness were then compared to other ML algorithms. Table 17 summarises the performance results of the ML algorithms and various time metrics in milliseconds.

TABLE 17. ML algorithm performance comparison (time in milliseconds).

Model	ACC	Total Time	Train. Time	Scor. Time
SVM	0.96	49206	989.2	1954.8
Decision Tree	0.96	6874	403.3	502.3
Random Forest	**0.99**	27911	824.6	1932.1
Naive Bayes	0.94	4177	**260.4**	**176.5**
Logistic Reg.	0.94	**4003**	453.9	208.1
Neural Net.	0.97	5288	534.9	210.9

Training time and Scoring time indicate values per 1000 items.

The resulting graph (Figure 8) revealed that algorithms such as SVM or Random Forest achieved reasonably high accuracy but at the expense of requiring much more time to build than other ML algorithms. In the case of Random Forest, this was because up to 30 trees must be built to obtain the ability to correctly classify. SVM algorithms are generally considered slower in runtime than other ML algorithms techniques which produce similar generalisation performance [35]. Taking into account the considerably longer runtimes and the similar accuracies produced by the algorithms, other classification models may, in this case, be preferable for application to our dataset.

The best or similar performance results were achieved by using the Naive Bayes, Logistic Regression and Neural Network algorithms. The accuracies achieved with the proposed neural network model improved on the other two models, and therefore it proved most suitable ML model for the investigated dataset.

Figure 9 graphically depicts the resulting neural network model, which consists of three layers. The input layer consisted of the five selected parameters. These inputs were processed by a pair of hidden layers leading to the output layer, which classified calls according to regular or SPIT users.

Finally, we compared the total time and the scoring time performance of the algorithms (Fig. 10). The graph indicates the quickest performing algorithms for each input parameter

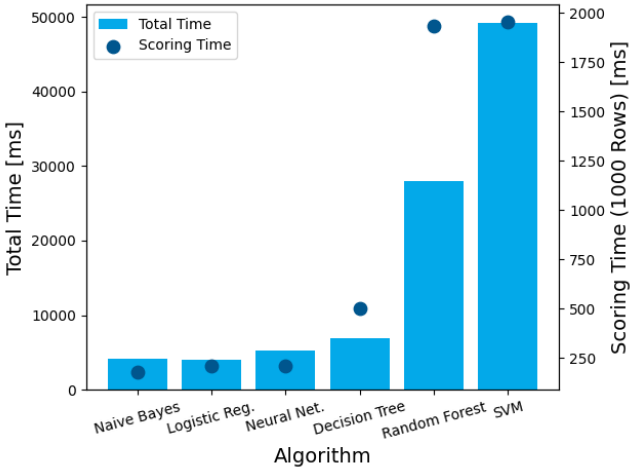


FIGURE 8. Performance of ML algorithms (5 Parameters).

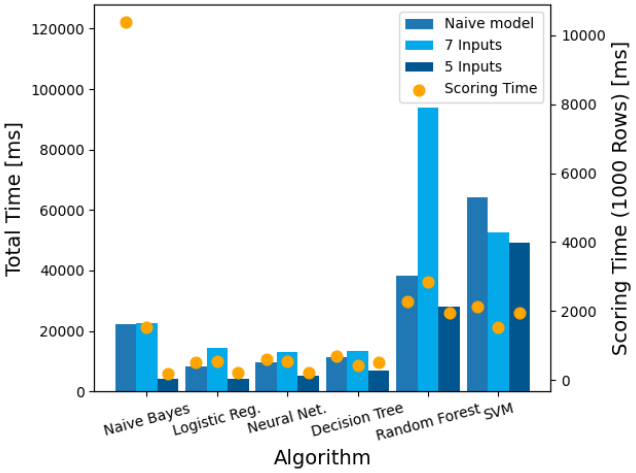


FIGURE 10. Runtime comparison of the algorithms for various input sets.

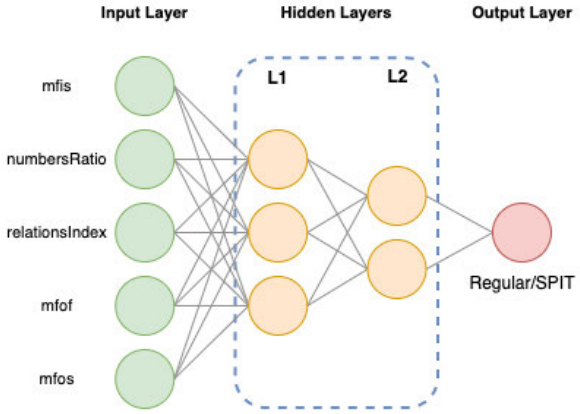


FIGURE 9. Architecture of the best performing neural network.

set (naive model containing all data and models with reduced sets of five and seven parameters). Reducing the number of inputs decreased classification time in each algorithm, and similar times were required for logistic regression, neural network and decision tree. The neural network was also able to also maintain high accuracy (Table 17).

The integration of new parameters, application of suitable ML algorithm configurations and comparison with a naive ML model produced an overall improvement in neural network performance, with scoring time up to 2.9 times quicker and almost no loss in accuracy. Applying the proposed NN model decreased the overall runtime by 2.5 times.

Compared to the model with seven input neurons, the final model with five input neurons produced a total runtime and scoring time 1.9 times and 2.5 times shorter, respectively.

The resulting confusion matrix (Table 18) indicates that six of the 170 predicted regular users were misclassified as SPIT users. The results for classifying SPIT users were similar, five users being incorrectly classified as regular users.

The precision and recall quantities were then defined for each category. Precision indicates the proportion of correctly

TABLE 18. Confusion matrix.

	True Regular	True SPIT	Class Precision
Prediction Regular	164	5	97.04%
Prediction SPIT	6	165	96.49%
Class Recall	96.47%	97.06%	

classified users from all classifications, and recall describes how many of the reference values the classifier determined correctly. Each of these metrics exceeded 95% in both classes, verifying the suitability of the proposed classifier for this dataset.

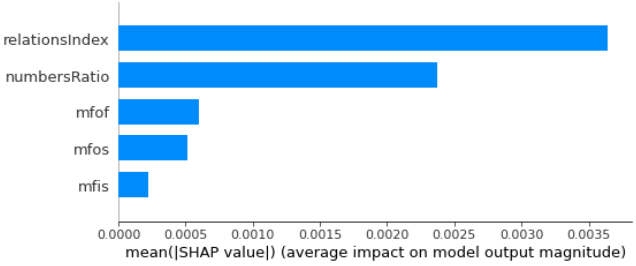


FIGURE 11. Variable importance plot.

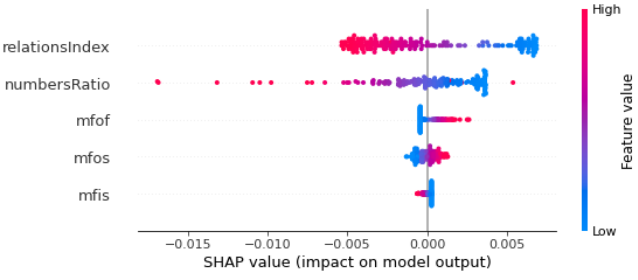
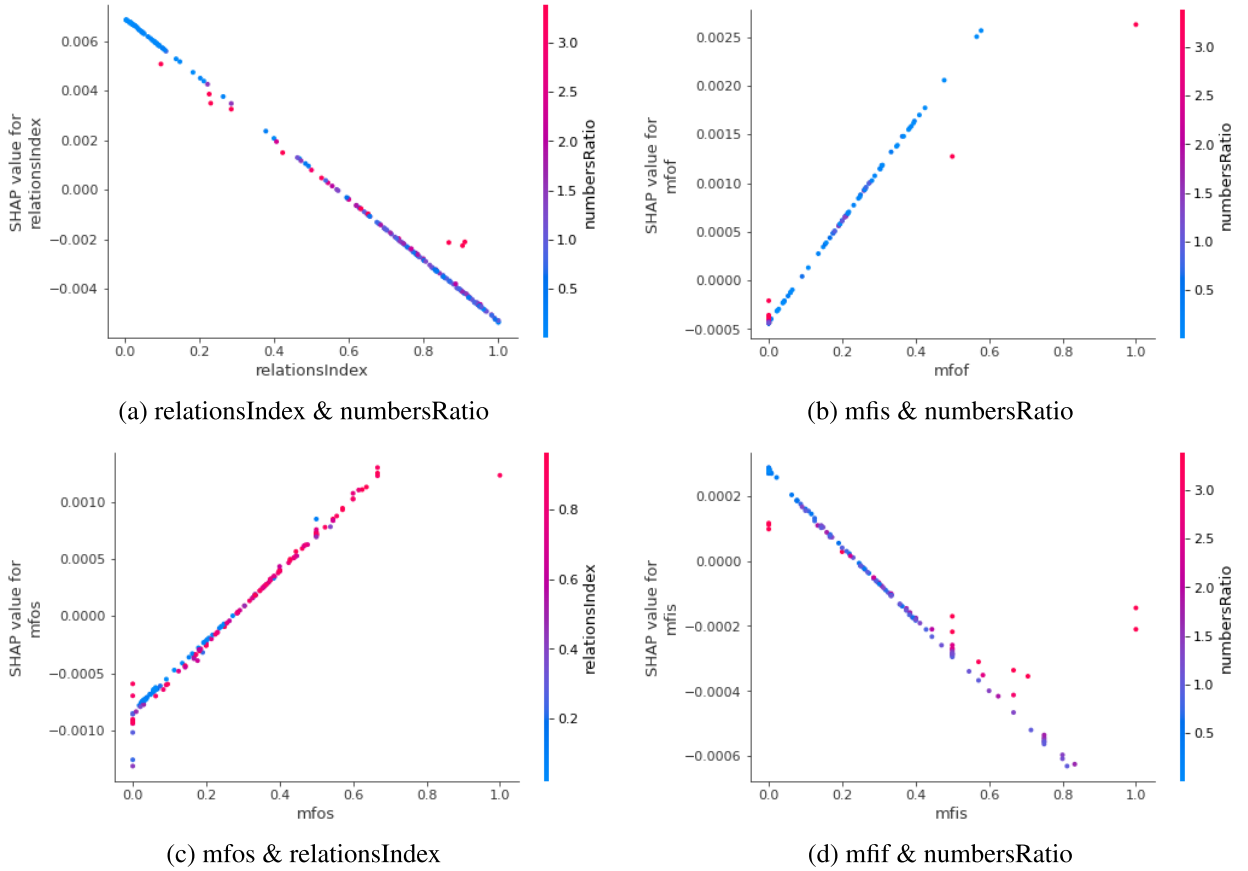


FIGURE 12. SHAP Variable importance plot.

3) LOCAL AND GLOBAL EXPLANATIONS USING SHAP

Lundberg and Lee [29] developed SHAP (SHapley Additive exPlanations) as a mechanism for explaining individual



(a) relationsIndex & numbersRatio

(b) mfis & numbersRatio

(c) mfos & relationsIndex

(d) mfif & numbersRatio

FIGURE 13. SHAP dependency plots for the SPIT detection features.

predictions. SHAP is based on Shapley Values, which are the theoretically ideal values in cooperative game theory. The concept underlying the importance of SHAP features is straightforward: using features which have high absolute Shapley values is crucial. Therefore, we summed the absolute Shapley values for each feature throughout the entire dataset to obtain their global importance.

Let us denote the NN model which requires explanation as f, the explanatory model as g, and the input variable and simplified input as z and z', respectively. SHAP uses an additive feature attribution to comprehend the ML model's output:

$$f(z) = g(z') = \phi_0 + \sum_{k=1}^{N} \phi_k z'_k \tag{9}$$

where N stands for the number of attributes, ϕ_k denotes the SHAP value of a feature k, and ϕ_0 denotes the constant value when all input variables are absent.

The local interpretations were aggregated by averaging the absolute Shapley values per attribute throughout the entire dataset to obtain a global interpretation of the NN model's predictions. The SHAP feature importance plot (Fig. 11) depicts the global effect of each feature on SPIT classification. The most significant variables are listed in descending order on a variable importance plot. The upper variables have a higher predictive ability than the lower variables because they contribute more to the model. According to the SHAP technique, the *relationsIndex* and *numbersRatio* parameters are the most essential features, whereas *mfis* is the least effective feature for this NN model.

Figure 12 shows a SHAP summary plotting the magnitude, direction and impact of each parameter on the output of the proposed model.

The graph indicates the following information:

- **Feature importance** – The parameters of the neural network are sorted according to relevance in descending order.
- **Feature impact** – The x-axis indicates whether the impact of a particular value is linked to a higher or lower prediction.
- **Original value** – The SHAP variable importance plot uses red points to indicate high values in the variable and blue points for low values.
- **Correlation** – Red points indicate a positive effect on the quality rating. Positive or negative feature impacts are shown on the x-axis.

An interesting question is what the partial dependency graph looks like. A partial dependency graph depicts the marginal impact of one or two variables on a ML model's

predictions [30]. We plotted SHAP partial dependence for the seven variables to gain some insight into the NN SPIT detection model's behaviour (Fig. 13). The figure depicts the influence of a feature (x-axis) on each instance's prediction (y-axis). The plot also depicts the properties which the variables interact with most and illustrates whether the connection between a target and a feature is linear, monotonic or complex.

For example, the accompanying graph (Fig. 13a) illustrates that *relationsIndex* and *numbersRatio* have an almost linear and negative trend. Figure 13c suggests an approximately linear and positive relationship between the *mfos* and *relationsIndex* parameters.

Chapter 4 discussed innovative parameters which were obtained from real-world data and applied as input to various machine learning algorithms. The most suitable detection model for the datasets was also identified, and an analysis of the neural networks designed for SPIT detection was presented as a contribution to reduce the lack of available literature on the topic. The novel parameters presented in the current study demonstrated an important role in SPIT detection, and a gradual reduction of these parameters aided in decreasing the computational and performance requirements of individual algorithms while maintaining high accuracy.

V. CONCLUSION

Through successive steps, the current study produced an innovative and highly effective SPIT detection mechanism which uses neural networks. The novel input parameters described in the study played a significant role in the proposed algorithm by increasing the performance of the classifier while maintaining high accuracy, which is especially important in real-time scenarios.

By applying different ML algorithm configurations, testing and gradually reducing the number of parameters, the study produced a neural network classifier which is able to distinguish SPIT users from regular users with a high degree of accuracy (97%) and has significantly lower performance requirements than other ML algorithms which apply the same input parameters.

With all 16 parameters as input, the neural network model achieved an accuracy of 99%, which is an improvement on other studies with real-world datasets [26], [27]. Artificial datasets (e.g. [23], [24], [41]) are also able to produce similar SPIT detection accuracy, but in working with real data, we expect lower accuracy. The approach in the current study demonstrates the effectiveness of the novel call parameters and the capability of the proposed neural network.

The study's comparison of algorithms provides a repeatable framework for future studies. The study also contributes significantly with its method of creating a real-world dataset from various telecommunications sources. This dataset has been published and is available online [33], and future studies by other authors might benefit from a similar approach in proposing new solutions and techniques to compare with the proposed neural network.

Of course, VoIP spammers come in many forms and detecting them is an important priority. SPIT users who send prerecorded messages to many end-users simultaneously can be easily identified. Through the methods applied in the current study, we identified a SPIT caller who was a call-centre employee attempting to reach as many network users as possible with the prospect of raising the chance to sell a product.

Given the nature of the data collected, future research could study the telecommunications relationships of individual users in more detail and perhaps define more differences which distinguish these users.

REFERENCES

[1] J. Zhang, J. Wang, Y. Zhang, J. Xu, and H. Wu, "A novel SPITters detection approach with unsupervised density-based clustering," in *Data Mining and Big Data*. Cham, Switzerland: Springer, 2018, pp. 314–324.

[2] F. Gorunescu, "Classification performance evaluation," in *Data Mining*. Berlin, Germany: Springer, 2011, pp. 319–330.

[3] D. Shin, J. Ahn, and C. Shim, "Progressive multi gray-leveling: A voice spam protection algorithm," *IEEE Netw.*, vol. 20, no. 5, pp. 18–24, Sep. 2006.

[4] J. Ahn, V. Shyamasundar, and Y.-T. Song, "Enhancing the blockage of spam over internet telephony (SPIT) using adaptive PMG algorithm," in *Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*. Berlin, Germany: Springer, 2008, pp. 15–26.

[5] B. Mathieu, S. Niccolini, and D. Sisalem, "SDRS: A voice-over-IP spam detection and reaction system," *IEEE Security Privacy*, vol. 6, no. 6, pp. 52–59, Nov. 2008.

[6] M. Falomi, R. Garroppo, and S. Niccolini, "Simulation and optimization of SPIT detection frameworks," in *Proc. IEEE GLOBECOM Global Telecommun. Conf.*, Nov. 2007, pp. 2156–2161.

[7] Y. Soupionis and D. Gritzalis, "Audio CAPTCHA: Existing solutions assessment and a new implementation for VoIP telephony," *Comput. Secur.*, vol. 29, no. 5, pp. 603–618, Jul. 2010.

[8] D. Sisalem, J. Floroiu, J. Kuthan, U. Abend, and H. Schulzrinne, *SIP Security*. Chichester, U.K.: Wiley, 2009.

[9] P. Patankar, G. Nam, G. Kesidis, and C. R. Das, "Exploring anti-spam models in large scale VoIP systems," in *Proc. 28th Int. Conf. Distrib. Comput. Syst.*, Jun. 2008, pp. 85–92.

[10] P. Golik, P. Doetsch, and H. Ney, "Cross-entropy vs. squared error training: A theoretical and experimental comparison," in *Proc. Interspeech*, Aug. 2013, pp. 1756–1760.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[12] H. K. Bokharaei, A. Sahraei, Y. Ganjali, R. Keralapura, and A. Nucci, "You can SPIT, but you can't hide: Spammer identification in telephony networks," in *Proc. IEEE INFOCOM*, Apr. 2011, pp. 41–45.

[13] J. Liu, B. Rahbarinia, R. Perdisci, H. Du, and L. Su, "Augmenting telephone spam blacklists by mining large CDR datasets," in *Proc. Asia Conf. Comput. Commun. Secur.*, May 2018, pp. 273–284.

[14] J. Quittek, S. Niccolini, S. Tartarelli, M. Stiemerling, M. Brunner, and T. Ewald, "Detecting SPIT calls by checking human communication patterns," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2007, pp. 1979–1984.

[15] J. Quittek, S. Niccolini, S. Tartarelli, and R. Schlegel, "On spam over internet telephony (SPIT) prevention," *IEEE Commun. Mag.*, vol. 46, no. 8, pp. 80–86, Aug. 2008.

[16] M. A. Nielsen, *Neural Networks and Deep Learning*. Hoboken, NJ, USA: Determination Press, 2018. [Online]. Available: http://neuralnetworksanddeeplearning.com/

[17] R. MacIntosh and D. Vinokurov, "Detection and mitigation of spam in IP telephony networks using signaling protocol analysis," in *Proc. IEEE/Sarnoff Symp. Adv. Wired Wireless Commun.*, Apr. 2005, pp. 49–52.

[18] Y.-S. Wu, S. Bagchi, N. Singh, and R. Wita, "Spam detection in voice-over-IP calls through semi-supervised clustering," in *Proc. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2009, pp. 307–316.

[19] F. Barghi, M. Hossein, Y. Moghadam, and H. K. Roshtkhari, "A comprehensive SPIT detection and prevention framework based on reputation model on call communication patterns," in *Proc. Iranian Conf. Intell. Syst. (ICIS)*, Feb. 2014, pp. 1–5.

[20] C. Hongchang, C. Fucai, and L. Shaomei, "A multilayered fusion method for SPITs detection," in *Proc. 4th Int. Conf. Intell. Comput. Technol. Autom.*, Mar. 2011, pp. 30–33.
[21] A. McDiarmid, J. Irvine, S. Bell, and J. Banford. (2011). *CRAWDAD Dataset Strath/Nodobo*. [Online]. Available: https://crawdad.org/strath/nodobo/20110323
[22] M. Pielot. (2019). *CRAWDAD Dataset Telefonica/Mobilephoneuse*. [Online]. Available: https://crawdad.org/telefonica/mobilephoneuse/20190429
[23] W. Nazih, Y. Hifny, W. Elkilani, T. Abdelkader, and H. Faheem, "Efficient detection of attacks in SIP based VoIP networks using linear l1-SVM classifier," *Int. J. Comput. Commun. Control*, vol. 14, no. 4, pp. 518–529, Aug. 2019.
[24] R. J. B. Chikha, T. Abbes, W. B. Chikha, and A. Bouhoula, "Behavior-based approach to detect spam over IP telephony attacks," *Int. J. Inf. Secur.*, vol. 15, no. 2, pp. 131–143, Apr. 2016.
[25] M.-Y. Su and C.-H. Tsai, "Using data mining approaches to identify voice over IP spam," *Int. J. Commun. Syst.*, vol. 28, no. 1, pp. 187–200, Jan. 2015.
[26] M. A. Azad, M. Alazab, F. Riaz, J. Arshad, and T. Abullah, "Socioscope: I know who you are, a robo, human caller or service number," *Future Gener. Comput. Syst.*, vol. 105, pp. 297–307, Apr. 2020.
[27] M. Nassar, R. State, and O. Festor, "Monitoring SIP traffic using support vector machines," in *Recent Advances in Intrusion Detection*. Berlin, Germany: Springer, 2008, pp. 311–330.
[28] M. D. Zeiler, "ADADELTA: An adaptive learning rate method," 2012, *arXiv:1212.5701*.
[29] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," 2017, *arXiv:1705.07874*.
[30] J. H. Friedman, "Greedy function approximation: A gradient boosting machine," *Ann. Statist.*, vol. 29, no. 5, pp. 1189–1232, Oct. 2001.
[31] F. Hutter, J. Lücke, and L. Schmidt-Thieme, "Beyond manual tuning of hyperparameters," *KI-Künstliche Intell.*, vol. 29, no. 4, pp. 329–337, Jul. 2015.
[32] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *J. Mach. Learn. Res.*, vol. 13, pp. 281–305, Feb. 2012.
[33] L. Behán, *Telemarketer & Regular User CDR Phone Records*. Genève, Switzerland: Zenodo, 2022, doi: 10.5281/ZENODO.6637796.
[34] K. Toyoda, M. Park, N. Okazaki, and T. Ohtsuki, "Novel unsupervised SPITters detection scheme by automatically solving unbalanced situation," *IEEE Access*, vol. 5, pp. 6746–6756, 2017.
[35] E. E. Osuna and F. Girosi, "Reducing the run-time complexity in support vector machines," in *Advances in Kernel Methods: Support Vector Learning*. Cambridge, MA, USA: MIT Press, 1999, pp. 271–283.
[36] S. Batthalla, M. Swarnkar, N. Hubballi, and M. Natu, "VoIP profiler: Profiling voice over IP user communication behavior," in *Proc. 11th Int. Conf. Availability, Rel. Secur. (ARES)*, Aug. 2016, pp. 312–320.
[37] G. Vennila, N. S. Shalini, and M. Manikandan, "Performance analysis of VoIP spoofing attacks using classification algorithms," in *Proc. Appl. Innov. Mobile Comput. (AIMoC)*, Feb. 2014, pp. 193–198.
[38] P. Kaur, A. Singhal, and J. Kaur, "Spam detection on Twitter: A survey," in *Proc. 3rd Int. Conf. Comput. Sustain. Global Develop. (INDIACom)*, 2016, pp. 2570–2573.
[39] A. Barushka and P. Hájek, "Spam filtering using regularized neural networks with rectified linear units," in *AI*IA 2016 Advances in Artificial Intelligence*, Cham: Springer, 2016, pp. 65–75.
[40] A. Barushka and P. Hajek, "Spam filtering in social networks using regularized deep neural networks with ensemble learning," in *Artificial Intelligence Applications and Innovations*. Cham, Switzerland: Springer, 2018, pp. 38–49.
[41] M. A. Azad and R. Morla, "Rapid detection of spammers through collaborative information sharing across multiple service providers," *Future Gener. Comput. Syst.*, vol. 95, pp. 841–854, Jun. 2019.

LADISLAV BEHAN received the master's degree in telecommunications from the VSB—Technical University of Ostrava, in 2016, where he is currently pursuing the Ph.D. degree with the Department of Telecommunications. His research interests include performance evaluation of the telecommunication infrastructure and machine learning algorithms applications.

JAN ROZHON received the Ph.D. degree in telecommunications from the VSB—Technical University of Ostrava, in 2016. He is currently an Assistant Professor with the Department of Telecommunications, VSB—Technical University of Ostrava. His research interests include performance evaluation of the telecommunication infrastructure, speech and video quality, and artificial intelligence applications.

JAKUB SAFARIK received the master's and Ph.D. degrees in telecommunications from the VSB—Technical University of Ostrava, Czech Republic, in 2011 and 2017, respectively. He is currently a full-time Postdoctoral Researcher in machine learning and neural networks with the VSB—Technical University of Ostrava. His current research interests include computational intelligence, improvements for machine learning, neural networks, and hyperparameter optimization.

FILIP REZAC received the Ph.D. degree, in 2016, for a thesis which focused on the issues of dynamic disaster-resilient network systems and their security. He is currently a Postdoctoral Researcher with the Department of Telecommunications, VSB—Technical University of Ostrava, Czech Republic. His research interests include VoIP technology implementations, network modeling, disaster-resilient network systems, next generation networks, and network security. He is a Copy Editor of the *Advances in Electrical and Electronic Engineering* journal.

MIROSLAV VOZNAK (Senior Member, IEEE) received the Ph.D. degree in telecommunications from the Faculty of Electrical Engineering and Computer Science, VSB—Technical University of Ostrava, Czechia, in 2002, and the Habilitation degree, in 2009. He was appointed as a Full Professor of electronics and communication technologies, in 2017. He has authored or coauthored over 170 articles in SCI/SCIE journals. His research interests include information and communications technology, particularly on quality of service and experience, network security, wireless networks, and in the last couple years also on big data analytic. According to the Stanford University ranking released in 2020 and also in 2021, he is one of the World's Top 2% of Scientists in Networking and Telecommunications.