SPY-BOT: Machine Learning-Enabled Post Filtering for Social Network-Integrated Industrial Internet of Things

Md Arafatur Rahman[a,*], Nafees Zaman[b], A. Taufiq Asyhari[c], SM Nazmus Sadat[b], Prashant Pillai[a], Ruzaini Abdullah Arshah[b]

[a] *University of Wolverhampton, School of Mathematics and Computer Science, Wulfruna Street, Wolverhampton, WV1 1LY, United Kingdom*
[b] *Faculty of Computing, University Malaysia Pahang, Tun Abdul Razak Highway, 26300 Gambang, Kuantan, Pahang, Malaysia*
[c] *School of Computing and Digital Technology, Birmingham City University, Millenium Point, Birmingham, B4 7XG, UK*

Abstract

A far-reaching expansion of advanced information technology enables ease and seamless communications over online social networks, which have been a *de facto* premium correspondents in the current cyber world. The ever-growing social network data has gained attention in recent years and can be handy for industrial revolution 4.0. With the integration of social networks with the Internet of Things being noticed in different industries to enhance human involvement and increase their productivity, security in such networks is increasingly alarming. Vulnerabilities can be characterized in the form of privacy invasion, leading to hazardous contents, which can be detrimental to social media actors and in turn impact the processes of the overall Social Network-Integrated Industrial Internet of Things (SN-IIoT) ecosystem. Despite this prevalence, the current platforms do not have any significant level of functionality to capture, process, and reveal unhealthy content among the social media actors. To address those challenges by detecting hazardous contents and create a stable social internet environment within IIoT, a statistical learning-enabled trustworthy analytic tool for human behaviors has been developed in this paper. More specifically, this paper proposes a machine learning (ML)-enabled scheme SPY-BOT, which incorporates a hybrid data extraction algorithm to perform post-filtering that arbitrates the users' behavior polarity. The scheme creates class labels based on the featured keywords from the decision user and classifies suspicious contacts through the aid of ML. The results suggest the potential of the proposed approach to classify the users' behavior in SN-IIoT.

Keywords: Behavioral analysis, computational method, machine learning, natural language processing, Social Network, IIoT, post-filtering.

*Corresponding authors

Email addresses: arafatur@ump.edu.my (Md Arafatur Rahman), zamannafees@gmail.com (Nafees Zaman), taufiq-a@ieee.org (A. Taufiq Asyhari), sohan.sadat@gmail.com (SM Nazmus Sadat), p.pillai@wlv.ac.uk (Prashant Pillai), ruzaini@ump.edu.my (Ruzaini Abdullah Arshah)

1. Introduction

1.1. Background

With the increase of information technology adoption in this modern era, people have changed their conventional way of communications. Sending emails, participating in video calls, or sharing opinions, images on social media are some of the examples of the modern way of interaction. This desire to connect has given rise to technological networks linking people, data, and things. Using these resources efficiently has led to the idea of linking smart objects while utilizing the data in the same network. Traditionally, this interaction is called the Internet of Things (IoT). IoT is vastly utilized in the industrial sectors nowadays, popularly coined as Industrial IoT (IIoT), as it improves productivity and builds meaningful relationships with technologies. People always prefer the seamless connection to interact and thus, social networking platforms aka virtual communication platforms are more popular and convenient rather than physical. Hence, integrating social networks with IoT would improve the overall productivity of an industry correspond to time, resources, etc.

Nonetheless, handling sensitive data and interacting with smart objects at the same time within the IIoT ecosystem are highly challenging. Instant and effortless access to these technologies has often caused security concerns. The immensely amplified use of cyber-based connectivity often results in undesirable side impacts. The availability of susceptible settings and user-friendly features can bring vulnerability in many stages for an organization. For instance, consider customer-supplier relationships in a common industrial shared platform. A customer can have more than one supplier, a supplier can have more than one customer and they share personal information, opinions on that platform. It is fairly possible that there might be some fraudulent customers or suppliers in that platform to scam.

Apart from the customer-supplier relationship, an industry can go through a similar situation where co-workers are connected with each other through an intra-communication platform or social network. The behavioral deviation is common among those users, which can lead to harmful incidents and misunderstandings, which can result in self-dejection and other bitter incidents in daily life. This might manifest into misbehaving attitudes like the form of threatening, pornographic, profanity, harassment, racist, vulgar contents [2, 3, 4, 5]. In the aftermath of exposing these sorts of offensive contents in Social Network-Integrated IIoT (SN-IIoT), the situation can easily bring hazardous impacts on both cybernetics and industrial productivity.

Given the inherent structural disarray of existing frameworks in SN-IIoT, users can easily expose their negative intents by posting obscene contents. Therefore, it is imperative to mitigate this situation so that the terrifying types of behavioral divergence in SN-IIoT cannot be swelled further. In this direction, content post-filtering in SN-IIoT is crucial in order to maintain collateral relation among the contacts [6]. The capability of a vast majority of existing SN-IIoT platforms is still inferior to cope with the challenges due to a lack of user behavioral analytic and precise features that identify the irregularity of conducts among the users. As a consequence, our previous work [7] encouraged us to design a framework for mitigating this kind of cyber hazards and suggested two filtering phases for the remedies. The first filtering phase is denoted as automated pre-filtering, which can be further divided

into: i) a trustworthy selection method of receiving friend request [8], and ii) an automated system of a friend-to-be model by the derivation of likelihood metrics [9]. The second filtering phase of the framework is designed to supervise post-friendship behavioral activities, which is referred to as automated post-filtering. This latter phase is the primary motivation behind this work.

1.2. Related Work

Online Social networks and anonymous-behavior are paralleled used terms in the cyber world. Facebook, Twitter, YouTube, Instagram, LinkedIn, Pinterest, Snapchat, etc. are very frequently used platforms by OSN users, and for their huge engagement in social media, they cannot maintain the consistency of behavioral integrity. Sometimes having with the voluminous divergence of characteristics bring hazards and manifest unlawful incidents. In this regard, many references have been proposed with various techniques in terms of post-phase analysis of malicious activities of the friends' behavior.

Sahay *et al.* [10] consider an approach to perceive and classify bullying of texts through some study and experiment. They run a substantial method for extracting text, user, and online-based attributes, to understand the intimidate users and their characteristics and differences from conventional users. Reassessments of the NLP method and gadgets gaining knowledge are considered and evaluated for figuring out bullying content in a dataset. But not have a significant role to eliminate behavioral activities regarding bullying on OSN.

Sarna *et al.* [11] tried to assemble the bullying messages into two parts- direct and indirect, then figuring out the solution by working on the bullying activities with checking the integrity of the user. But this paradigm did not take into account the user's friend behavioral attitudes and diversities in a complete manner.

Chen *et al.* [12] come up with the Lexical Syntactic Feature (LSF) method to identify abusive material along with potential unpleasant activities in the field of an online social network. The influence of vulgar and offensive language has been individualized by figuring out unpleasant content for this proposed scheme. They introduce the hand-authoring syntactic policies to classify the name-calling harassments.

Das *et al.* [13] designed a framework, to explain and examine a large quantity of social community user behavior evaluation methods with some classifications, unique emphasis on behavior characterization, behavior recognition and behavior prediction under two large perspectives -chronic consumer behavior & non-continual person behavior evaluation. It is partially highlighted a few research guidelines in the social community on personal behavior evaluation.

Wang *et al.* [16] suggested the internal measurement policy depend on a communal filtering method for the top-N hashtag assumption rate forecasting, which allows the coherence for the chronological dynamics. This scheme makes the connection in user likings and impact on their communication combining with linear threshold paradigm and improved CF version to establish behavior interiors activities. This method is applied solitary on the microblog Twitter.

Benevenuto *et al.* characterizes [17] characterized a evaluation process of consumer capabilities in cyber-space. The evaluation demonstrates the energy of using clickstream

statistics in identifying styles in social network capacities and social interactions. This kind of technique is generally used to navigate the propagation of consumer abilities.

This paper [19] deployed a framework for identifying harmful online inter-relations in the context of abusive contents achieved via textual content messages as well as pictures. The permutation of textual content and image evaluation strategies are taken into consideration of proper platform for the finding of possible cyberbullying hazards.

Buettner [21] endorse a framework named persona-based total product recommender (PBPR) to investigate social media facts as a way to predict a person's personality and the preference of product based on personality. They outlined the PBRS system to assess an IT-object to get an exclusive XING dataset from social media. It is a kind of technique to determine the product based on individual personality.

Yu *et al.* [23] contributed a partially supervised online spammer detection technique, by conveying on the highest use of textual content and user activities, and maintaining the record of social connectivity. By modeling the concept, they attempted to adjust a unique controlled NMF-based semi-supervised set of rules on gaining knowledge of CNMF. To analyse social media spammer detection, they tried to implement collaborative itemization on the message content, consumer behavior, and social relation by using rules of matrix. However, it is not an entire solution to regulate behavioral indecency.

Yan *et al.* [24] described the users' 'following' characteristics and attended as out-degree and in-degree with a microblog social community. They proposed a social network based human dynamics version and indicated the convincing power and instinctive force in terms of behavior for posting microblogs. Yousukkee [25] targeted the technique of classifying online social-based consumers and reviews numerous strategies to investigate consumer behavior in online social networks.

Several NLP techniques have been applied on social network behavioural Analysis. This paper [10] has developed code switch text using sub-word level LSTM to improve the efficiency of the marketplace in Facebook.

Sun *et al.* [**?**] has established an investigation platform using NLP techniques which incorporates the online social network data and identifies criminal offenses.

As NLP can be incorporated with online social network platforms, it provides motivation to work with NLP techniques in order to make online social network more secure for its users.

Unlike the aforementioned works, in this paper, we propose a machine learning-enabled post-filtering scheme that creates class labels based on the featured keywords from the decision user, and according to those labels, it detects indecent activities and classifies suspicious contacts that brings an advantageous to make the affairs more secure.

1.3. Contribution of this Work

In this paper, we investigate a post-filtering approach, wherein a user can take prosecutions against the offensive behaviors of existing contacts. This is driven by the fact that any deviating behavioral activities can likely transform into cyber hazards in SN-IIoT. By exploiting text processing, our proposed scheme relies upon generating class labels and an automated filtering technique with the aid of machine learning (ML). The purpose is to

identify behavioral divergence and classify the users' characteristics based on normal and diverging behaviors. More specifically, our contributions include the following.

- We propose a novel framework called **SPY-BOT** that analyzes social network behavior of user's SN-IIoT contacts and notifies the user accordingly. This framework utilizes text processing that is invoked to identify patterns relevant for classifying accepted (normal) behaviors and deviating behaviors.
- We introduce a novel **hybrid SN-IIoT data extraction algorithm** to retrieve users' data from the common shared platform. The extracted data could be used as features that distinguish the behaviors' polarity.
- Next, we develop a **friends matching metric**, a weighted data labeling scheme, to evaluate the user friends' behavioral polarity. While the current work leverages a predefined set of weight values, which is based on fine-tuning experimental data, this might be optimized depending upon the performance requirement.
- Finally, we demonstrate the effectiveness of our proposed scheme in providing the overall content classification and predicting a huge selection of obscene posts as part of the behavioral engagements. After obtaining the system-generated notification from the classified value, the SN-IIoT user is able to take actions against the distrustful contacts.

The analysis of the ML algorithms and their performance reveals that with tuned hyperparameters, our proposed technique acquires up to 92.7% accuracy on the validation dataset and up to 90.1% on the unseen test dataset. This performance is well accompanied by the desirable values of other metrics, namely high precision, recall and F-1 score. These results demonstrate the promising direction of this proposal in embedding computational intelligence for post-filtering content selection in the online social network platform to promote secure and hazard-free cyber environments.

The rest of the paper is sorted in the following order. Related works are reviewed in Section 1.2. We then present our proposed SPY-BOT framework for social network post-filtering analysis in Section 2. This is then followed by Section 3 where we provide the delineation of the previous works and the currently proposed method. Evaluation of the ML techniques and findings on the real world extracted dataset are described in Section 4. Section 5 concludes the paper and explains possible progressions of this work.

2. SPY-BOT: Machine Learning-Enabled Post Filtering for Social Network-Integrated Industrial Internet of Things

Machine Learning is a subset of Artificial Intelligence (AI) that caters to a system the capacity to understand the patterns of the data naturally without being explicitly instructed. The main substance of machine learning is data, which are often referred to as observations. Finding the pattern of those observations, a machine learning model predicts an outcome.

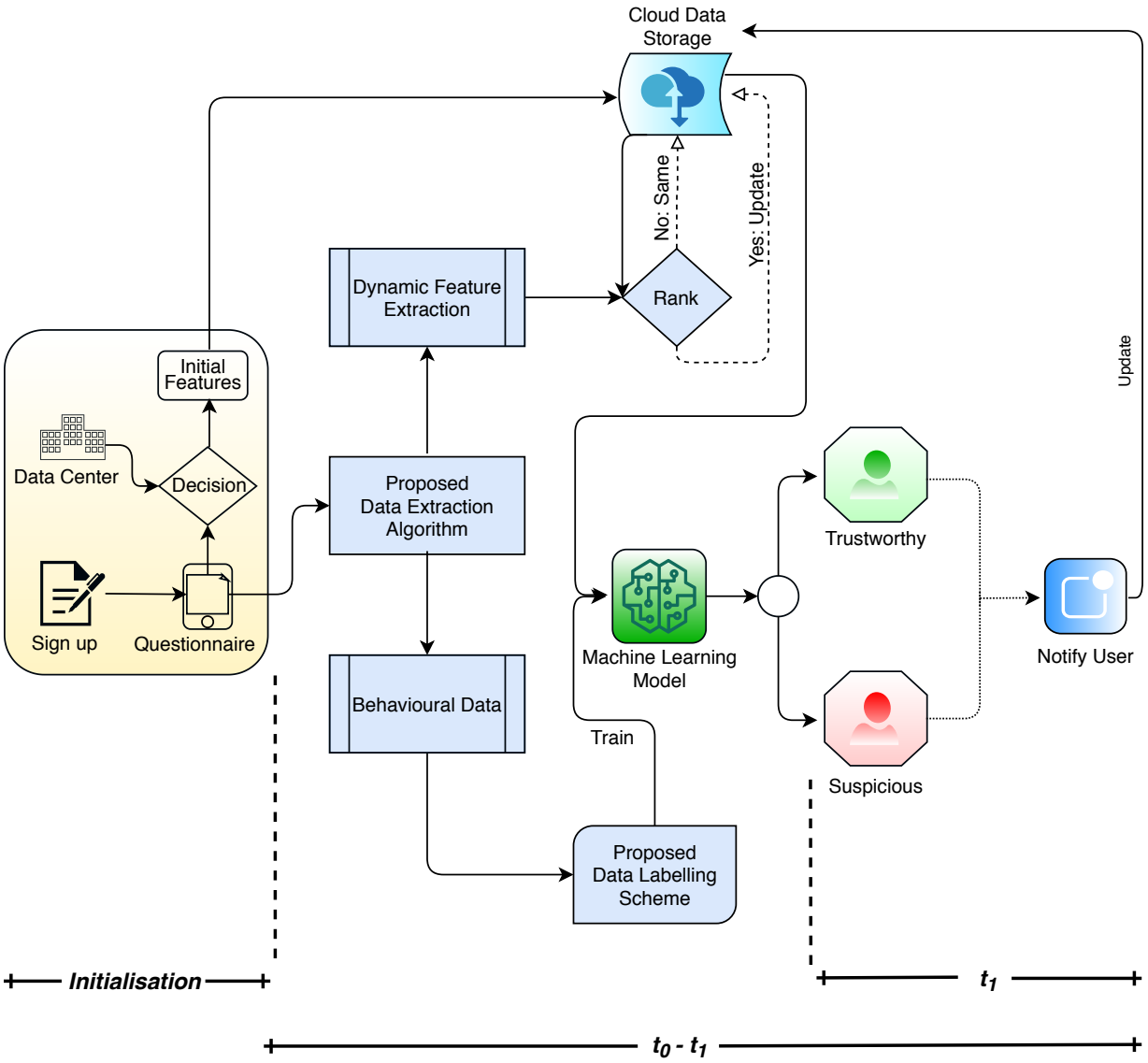

Figure 1: Proposed functionality of the SPY-Bot

In this paper, we propose a functional bot that can capture, process, reveal undesirable contents among SN-IIoT contacts, and notify the decision user with the help of machine learning. The functionalities (as depicted in Figure 1) of the bot are detailed in this section.

2.1. Initialization

To use an online social media platform, one must perform an account creation task at the very beginning. Once it is done, he can start surfing that platform. As our bot aims to deliver better recommendations, it poses the user some applicable inquiries to get a rudimentary thought of his inclinations. After extracting that, it compares and ranks

features with another huge set of features that are previously stored in a company database. From there, our bot provides a certain amount of features that are relevant to the user and stores them in a database. The selection of features is dynamic and will be updated periodically, which is discussed in the next subsection.

2.2. Data Retrieval

As we aim to deliver a better recommendation of friends according to the user's preference, we plan to retrieve data concerning two different parameters namely dynamic feature selection and behavioral data.

- **Dynamic Feature Selection:** A machine learning model is trained through input features and it provides outputs according to that. In order to keep track of user's preferences, our bot intends to train the machine learning model with different features at separate periods which we are calling dynamic feature selection. Based on the features that our bot retrieves only from the user's post, it compares and ranks those with a previous set of features and updates if necessary. Consequently, if a user changes his preferences, the SPY-Bot will act and recommend correspond to that.

- **Behavioral Data:** The major concern of our bot is to break down appropriately the behavioral actions of the user's friends. To perform that, it retrieves data from the user's news feed. This information incorporates posts, shares, images, posted by the user's friend. Once it is done, the processed data are fed to the machine learning model for analyzing the behavior of the user's friends concerning the user's choice. For example, if a decision user does not like certain things, a person of opposite choices would not be suggested by our bot. Rather it notifies the user to check either the person of opposite preferences is okay to have space in the user's friends' list.

2.3. Proposed Data Extraction Algorithm

To fetch behavioral data from SN-IIoT we propose a hybrid algorithm (as depicted in Algorithm 1) which incorporates some of the techniques of natural language processing(NLP). This hybrid algorithm will collect and process data from the user's news feed.

To start with, we initialize a variable *dict* as a dictionary inside the Main function (as depicted at Line #27) where we take f_{name} as dictionary keys and $p_{f_{name}}$ as the corresponding values using a loop at Line #31. Here, f_{name} refers to the friends' names, and $p_{f_{name}}$ refers to the posts shared by the friends. We also take another variable called p_{count} to retrieve the number of posts available on the decision user's news feed in a single day. The loop takes inputs for *dict*'s key-value pair until p_{count} reaches to 0. After that, we call a function Preprocessing that is defined at Line #7. Accessing the value of *dict*'s, it processes the text with help of the TextProcessing function, defined at Line#1. TextProcessing function is responsible for applying various natural language processing techniques, including stemming, normalization and tf-idf on the given text. These processed texts are updated as key-values at Line #10 and return an updated dictionary. This updated dictionary is further called

Algorithm 1 Data Extraction

```
1: function TextProcessing(v_n):
2:     v_sn ← Stemming(v_n)
3:     v_nn ← Normalization(v_sn)
4:     v_tn ← TfIdf(v_nn)
5:     return v_tn
6: end function

7: function Preprocessing(dict):
8:     for k_n,v_n in dict do
9:         for x in v_n do
10:            v_n ← TextProcessing(v_n)
11:        end for
12:    end for
13:    return dict
14: end function

15: function Search(dict, u_feature):
16:    u_f ← DataFrame
17:    for k_n, v_n in dict do
18:        for y in v_n do
19:            if v_n in u_feature then
20:                uf.append(k_n, v_n)
21:            end if
22:        end for
23:    end for
24:    return u_f
25: end function

26: function Main:
27:    dict ← dictionary()
28:    flag ← True
29:    p_count ← max(t_now-1)
30:    u_feature ← DataFrame
31:    while flag do
32:        f_name ← input(friends name)
33:        p_fname ← input(friends post)
34:        if f_name in dict then
35:            dict[f_name].append(p_fname)
36:        else
37:            dict[f_name] = p_fname
38:            p_count−−
39:        end if
40:        if p_count == 0 then
41:            flag ← false
42:        end if
43:    end while
44:    P_process ← Preprocessing(dict)
45:    P_data ← Search(P_process, u_feature)
46:    return P_data
47: end function
```

for matching with a data-frame named $u_{feature}$ using the Search function at Line #45. $u_{feature}$ contains the initial features and values preferred by the decision user. The matching features and values are returned as a data-frame named u_f which is the tabular form of friends' behavioral data.

2.4. Proposed Data Labelling Scheme

The novelty of this paper comes into play in this subsection. This subsection defines the process to evaluate the user friends' behavioral polarity by using friends matching metrics. Initially, the weightage value of features is inputted by the decision (DU) and formulated by the friends matching metrics; afterward, the value [0; 1] of the decision evaluator is yielded to decide or classify the friends' behavioral state by complying with a predefined threshold.

The initial stage of the method evaluates the sum of the weight of friends matching

metrics is given by

$$FMM_u = \sum_{i=1}^{N^p} f_i^p w_i^p + \sum_{j=1}^{N^n} f_j^n w_j^n \tag{1}$$

Where, N^p and N^n are the total number of positive and negative features respectively used for evaluation of $i \in 1, ..., N^p$ and $j \in 1, ..., N^n$.

$$f_i^p = \begin{cases} 1, \text{if agree with decision user} \\ 2, \text{if there is no comments} \\ 3, \text{if disagree with decision user} \end{cases}$$

$$f_j^n = \begin{cases} 1, \text{if agree with decision user} \\ 2, \text{if there is no comments} \\ 3, \text{if disagree with decision user} \end{cases}$$

w_i^p and w_j^n are the weightage of the positive and negative features, which will represent in the scale between -5 to 5.

$$Decision = \begin{cases} \text{Good, if } \dfrac{FMM_{u_d} + FMM_{u_f}}{2FMM_{u_d}} \geq Th \\ \text{Bad, if } \dfrac{FMM_{u_d} + FMM_{u_f}}{2FMM_{u_d}} < Th \end{cases} \tag{2}$$

Here, FMM_{u_d} and FMM_{u_f} are the friends matching matrices for user and friend respectively. Note that, the weights of FMM are calculated individually user and friends respectively. Finally, the result [0; 1] that underpins the potential to classify a friend is "Good" or "Bad" from the equation 2.

2.5. Machine Learning Model

In furtherance of identifying suspicious individuals, a machine learning model is developed. As suggested by NFL(No Free Lunch) [26] theorem, there is no reason to choose one machine learning algorithm over another unless we make assumptions on the dataset. Moreover, concerning time and cost constraints, it is not feasible to try every single machine learning algorithm on a dataset. Thus, we decide to use algorithms that are performed well in binary classifications before such as support vector machine, logistic regression. However, basic machine learning models may not perform very well initially. Hence, we further tune those models to achieve higher performance. The input of the machine learning model is the behavioral data with respect to dynamic features retrieved from the cloud data storage. Following that, the final model predicts either the friend is trustworthy or suspicious, notifies the decision user, and updates to the cloud data storage.

All the data extraction procedures to notify the user and update cloud data storage are executed during $t_0 - t_1$ time where notifying and update takes t_1 time. This repeats further for $t_2, t_3, ...t_n$ times.

3. Model Development and Implementation

Section 2 sets the direction of our proposed approach. Nonetheless, in this paper, we break-down our methodology into 4 phases (as depicted in Figure 2), which are delineated in the accompanying sub-sections.

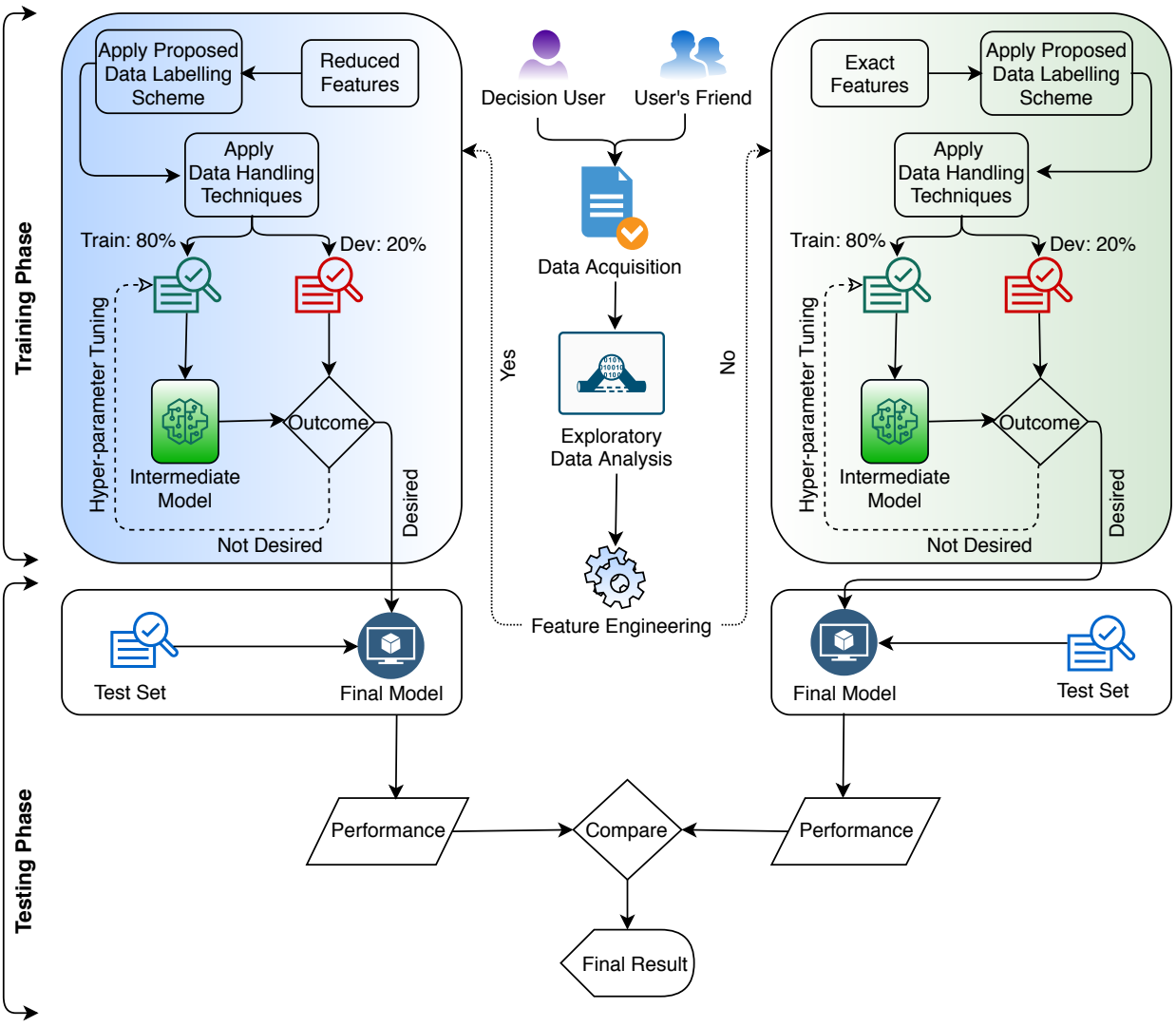

Figure 2: Our methodology in designing a machine learning-enabled post filtering approach for OSN.

3.1. Phase One: Problem Definition

Machine learning models are widely used nowadays to provide business solutions. Starting from MNC's to SME's, stakeholders are largely inclined to machine learning for availing a fair amount of revenue in quick time. The purpose of machine learning is not finite to business solutions, rather it serves a wide region of modern life. Different sectors include healthcare, education, transportation, are benefited by digging up meaningful insights and

solutions to diverse problems through machine learning. Often these industries use machine learning to yield recommendation systems by which they can anticipate the characteristics of users and prescribe things corresponding to that. Several works in literature have considered on online social networking platforms as well, such as mitigating hazardous activity [7], pre-filtering approach [9] etc. However, there have not been introduced any post-filtering approaches yet in online social network platforms. Thus OSN users are deprived of this emerging technology. Like many other approaches, user's information and preferences are extracted to design a functional bot in this paper. Other than recommend friends behavior is out of the scope of this paper.

3.2. Phase Two: Data Extraction

This phase is responsible for storing relevant data and dispatching those to the training phase. In this phase, there are three sub-phases such as Data Acquisition, Exploratory Data Analysis and Feature Engineering that are described below:

3.2.1. Data Acquisition

We start with acquiring relevant data from the users in this phase. As part of our experiment, one hundred users are distinguished randomly, fulfilling the criteria of having more than five hundred friends on social media. In the case of data collection from any person or entity, it is of course an ethical obligation to look after the autonomy of that individual or organization. Any survey should be conducted in such a way that the ethical features are observed and only then the best practice of research can be ensured. The privacy of respondents must be respected and comply with all legal requirements for data protection as well [27] and finally, the aims of the survey need to be informed. For that purpose, after randomly opting for one user to volunteer for this research, we take all the endorsement from him to use his data from social media and maintain all the confidentiality and legitimacy. Along with that, he also agrees to share his friends' behavioral data as our proposed model needs both the user's and his friends' behavioral data. Due to the security of private data and the unavailability of data extraction resources (e.g. Twitter public API only gives access to extract extremely limited historical tweets), we follow a questionnaire-based survey method to extract information from the user.

3.2.2. Exploratory Data Analysis

Exploratory Data Analysis aids in understanding the summary and distribution of data. It additionally assists to choose which algorithm we should use to achieve an optimal solution. In our dataset, we deal with eleven features with 600 training examples with two class labels 'Good' and 'Bad' where 450 training examples are from class 'Good', and the rest of the 150 observations are from class 'Bad'. This delineation indicates that the dissemination of classes is not adjusted as the ratio of two classes 'Good' and 'Bad' is 3:1. Even though the class imbalanced characteristic of our dataset is not extreme, it is pretty much common in real-life datasets. According to this analysis, we decide to choose evaluation matrices that suit such an imbalanced dataset which will be clarified in Section 3.3.5 and Section 3.3.6.

3.2.3. Feature Engineering

The performance of machine learning models is highly reliable on the representation of feature vectors. Along with improving the predictive power, it also mitigates the complexity and makes a model simple which is easier to execute in terms of time and memory constraints[28].

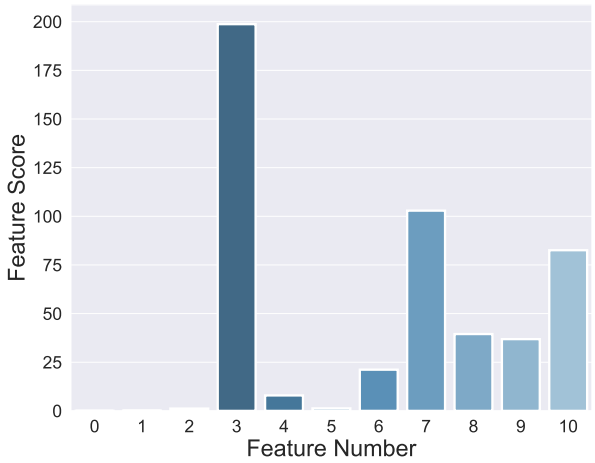

Figure 3: Feature scoring

In our case, we use a Chi-squared statistic to test the relevancy of input features with respect to the outcome. To add to that, chi-squared is applied due to categorical variables such as Gender, Age Level, Relation, Bonding, Origin, Religion, Political Views, Social Groups, Followers, Racism Terms, Liquor Item, Abusive Words in our dataset. From Figure 3 we notice the predominant features like Bonding, Social Groups, Abusive Words with a high score.

3.3. Phase Three: Training

This phase comprises six steps. From the data labeling scheme to finalize the end model, all are executed in this entire phase. These steps are analogous for both either using feature engineering or not except the number of features. While not using the feature engineering mechanism, we take the exact number of features from the dataset to train a model. However, using the feature engineering technique, we reduce the dimension of the dataset and consider the five most predominant features that have high scores depicted in Figure 3. Each of these six steps of this phase is discussed in detail in the followings:

3.3.1. Step One

Our data labeling scheme is implemented in this step. Below is a sample demonstration of the calculation of our data labeling scheme.

- We are to get the friend matching metrics FMM_u, from 'sample one', where, we have N=11, then,

  $FMM_{u_d} + FMM_{u_k} = ((11*5) + (2+1+1+0+2+1+3+2+1+5+5)) = 78$,

  where, the agreed (i.e., selected by '1') attributed values of corresponding feature by the DU gets the maximum assigned weight, unexpressed or no comments (i.e., selected by '2') attributed value gets null (0) value and disagreed (i.e., selected by '3') attributed value gets maximum negative assigned weight.

  Finally, the decision can be regulated from the function by paralleling with the predefined threshold (Th) value. e.g., if the assessed denomination is 'equal or above' to user predefined threshold (Th) value, then it returns the behavioral status as "good" otherwise "bad".

  Let, the user assigned threshold $(Th) = 0.60 \approx \mathbf{60\%}$,

  now for the user-friend 'sample one', we procure the formulated score is $(78)/2 * 55 = 0.71 \approx 71\%$.

  The relative amount of **71%** means that the friend for 'sample one' potentially to be a 'good' on behavioral status.

- For the user 'sample two' the procured value is $(55 + 33)/2 * 55 = 0.80 \approx 80\%$.

  The result, **81%** processes for the friend 'sample two' potentially to be 'good'.

- For the user-friend 'sample three' we gain the quantity is $(55 + 2)/2 * 55 = 0.52 \approx 52\%$.

  The outcome **52%** means, 'sample three' potentially to be a 'bad'.

3.3.2. Step Two

As mentioned earlier, we are dealing with categorical data. Categorical data are often described as nominal and ordinal in statistics. Although machine learning algorithms such as decision tree can handle categorical data, other algorithms are not capable enough to do so. As we pick, Support Vector Machine and Logistic Regression to train our model, we must use encoding techniques to feed the data properly into the model. We use a popular machine learning package 'scikit-learn' in python, from there we choose the 'preprocessing' library and import 'OrdinalEncoder' and 'LabelEncoder' to encode the categorical variables.

Ordinal values are natural, ordered and the range between the categories is unknown[29]. Our proposed model defines weightage to different observations; thus, the categorical values of the dataset are considered as ordinal. For this reason, we pick ordinal encoding over one-hot encoding. The OrdinalEncoder function takes data having the shape of (number of samples, number of features) while LabelEncoder only fits data that has the shape of (number of samples,), often known as a rank-one array[30]. Hence, we use LabelEncoder to encode the target feature only.

3.3.3. Step Three

Once the data are ready to feed into a model, we split it into two sets namely the training set and the development set (dev set in short). We suggest splitting the data proportion of 98% : 1% : 1% for the train, dev, and test set in a real-life scenario. This is logical because a lot of social media users will contribute to a huge amount of data. For example, if we have 1 million observations, 1% of that will be 10000 observations, which are more than enough for dev and test sets. A machine learning model performs according to the amount of data as well. If we take too many observations for dev and test sets, the model will not get enough data points to learn the representation causing underfitting[31]. However, in the case of small datasets, as the number of observations is less, it is better to use the conventional theoretical split of 60% : 20% : 20% for the train, dev, and test sets. We take 80% of 600 observations for training and 20% for the development set as we have a separate test set equivalent to the amount of dev set.

3.3.4. Step Four

In this step, we apply machine learning algorithms to train the models. As our classification task is a binary classification, support vector machine[32] and logistic regression[30] would perform efficiently[33, 34]. Thus, we stick to these two algorithms to develop our machine learning model. We use both algorithms with default parameters to see, how those perform without tuning in our dataset. The result will be demonstrated in Section 4.

3.3.5. Step Five

In this step, we evaluate our newly created machine learning model. As our task is binary classification exploiting an imbalanced dataset, we choose our evaluation metrics according to that. Those are log loss [35], and accuracy [36, 37]. Moreover, we draw a confusion matrix to get the precision, recall, specificity and F-1 score of our models.

3.3.6. Step Six

In this final step of the training phase, we decide either we should stick with the current model or not. If the current model satisfies the desired outcome, we assign that as our final model and the model is ready for testing which will be discussed in the next sub-section. However, if it does not satisfy the criteria of deployment, we jump back to 3.3.4 again. In this procedure of going back to 3.3.4, we tune hyperparameters, so that the new model can outperform the older one. Hyperparameter tuning is basically an optimization task. A regularization hyperparameter regulates the flexibility, degrees of freedom of a model which

prevents overfitting[**?**]. Therefore, it has a huge impact on prediction accuracy. Optimal hyperparameter settings may differ from dataset to dataset, thus the process of tuning hyperparameters may require more than once. Nowadays, a lot of existing works focused on the construction of algorithms for finding optimal hyperparameters[38, 39]. We use the 'Grid Search'[30] algorithm to find the optimal hyperparameter for our dataset. Once, that is done, the model is retrained with new parameters and goes through 3.3.5 and this step repeats until it satisfies the desired outcome.

3.4. Phase Four: Testing

In this phase, we demonstrate the outcome of the final machine learning models by making predictions on a test dataset. As we split our training phase into two parts, with feature engineering, and without feature engineering, the same goes for this phase as well. As soon as we get a final machine learning model for the feature engineering part and another model for the non-feature engineering part from Section 3.3.6, our models are tested on a dataset that is unseen to our models and contains 120 observations. Finally, we observe the effectiveness of those two models comparing their performances, and conclude a decision. Although the display result portion is included in this phase in Figure 2, for the coherence of writing, we will discuss it in the result and discussion section.

4. Validation, Results and Discussion

This section discusses, analyzes, and compares the performance evaluation of all the machine learning models mentioned above in Section 3.3.4. We divide our experiments into two stages. In the initial stage, we train a logistic regression model and a support vector machine model with default parameters and analyze their performances using the exact number of features from the original dataset. After that, we tune hyperparameters of those models and analyze their performances as well. Furthermore, we compare the outcomes and select the supreme one. Finally, we tested the chosen model on an unseen dataset to predict and analyze the performance.

In the second stage, we provide the five most dominant features that are found in Section 3.2.3, to train several machine learning models. The rest of the workflow is analogous to the procedures that are mentioned above at the initial stage in this section.

4.1. Experimental Setup

We use the same device to compute and execute all the ML models. The training dataset consists of 600 observations and 11 features and 2 class labels. Splitting those observations into the training set and the dev set is equal for all the created models. All the simulations are done almost in an analogous experimental setup. However, few changes are made in stage two due to the demand of the situation. Therefore, we will see how these changes affect the overall performance of ML models.

Table 1: Performance Evaluation on Different Measures in Training Phase

Algorithms	Measures	Precision	Recall	F1-score	Log Loss	AUC Score
Basic Logistic Regression	Macro Average	0.87	0.83	0.85	0.312	0.907
	Weighted Average	0.89	0.89	0.89		
Basic Support Vector Machine	Macro Average	0.87	0.86	0.86	0.315	0.905
	Weighted Average	0.90	0.90	0.90		
Tuned Logistic Regression	Macro Average	0.88	0.84	0.86	0.311	0.905
	Weighted Average	0.90	0.90	0.90		
Tuned Support Vector Machine	Macro Average	0.92	0.90	0.91	0.280	0.927
	Weighted Average	0.93	0.93	0.93		

4.2. Stage One

We begin our experiment by constructing a logistic regression model and a linear support vector machine model. From Table 1 we can notice, the logistic regression model has high precision, recall and F1-score for weighted average scores. This is reasonable because the predominant class is positive. However, it is best to consider the macro average score as it is not biased to any dominant classes and the score of macro average is not that satisfactory. The overall accuracy of the model is 89%. Moving on to the support vector machine model, we see from Table 1, even though the performance is slightly better than logistic regression, the results are not appeasing to choose this as the final model. To enhance the performance of the predefined models, we tune hyperparameters and evaluate the performance. We use the 'Grid Search' algorithm to find optimal hyperparameter values. From that search, for the logistic regression model, we find the best 'solver' parameter namely 'liblinear'[30]. The regularization we find otherwise known as 'penalty' parameter is 'L1' regularization aka Lasso regression[30]. However, these parameters have not been able to significantly change the performance of the model, which can be seen in Table 1. The basic support vector machine model has performed much better than the tuned logistic regression model which indicates not to select logistic regression models for our problem.

As the basic support vector machine model has shown promise to perform well in our dataset, we tune the hyperparameters of the support vector machine model. We use kernel 'RBF' known as radial basis function [30] which supports non-linear decision boundary and find the value of 'C', the regularization parameter, 1. This hyperparameter tells the SVM optimized model how much we want to avoid misclassifying training observations. After retraining the model with the best-chosen parameters, we observe from Table 1 that, precision, recall and F1-score are remarkably high. From Table 1, it is evident that the tuned SVM model has the lowest log loss value of 0.28, and the area under the curve is almost 93% while the others have 90% in their bags. From the above analysis, we decide to choose a tuned support vector machine as our final model.

Finally, we evaluate our model on an unseen dataset. We plot the accuracy scores on a visual diagram and as expected, our model has outperformed some state of art models with an accuracy of 90.1% (as depicted in Figure 4). We can also observe the high precision, recall, f1-score, area under curve value from Table 3.

Table 2: Performance Evaluation on Different Measures in Training Phase with Feature Engineering

Algorithms	Measures	Precision	Recall	F1-score	Log Loss	AUC Score
Basic Logistic Regression	Macro Average	0.86	0.81	0.83	0.301	0.897
	Weighted Average	0.88	0.88	0.88		
Basic Support Vector Machine	Macro Average	0.86	0.81	0.83	0.321	0.89
	Weighted Average	0.88	0.88	0.88		
Tuned Logistic Regression	Macro Average	0.87	0.83	0.85	0.301	0.895
	Weighted Average	0.89	0.89	0.89		
Tuned Support Vector Machine	Macro Average	0.89	0.86	0.87	0.332	0.898
	Weighted Average	0.91	0.91	0.91		

Table 3: Performance Evaluation on Test Dataset

Algorithms	Classes	Precision	Recall	F1-score	Log Loss	AUC Score
Final Model (Tuned Support Vector Machine)	Macro Average	0.92	0.89	0.89	0.20	0.99
	Weighted Average	0.91	0.90	0.90		

4.3. Stage two

We formulate the same machine learning models as Stage one, however, instead of taking all features, we use only five features to train appropriate machine learning models. The reason behind this is to find if feature engineering excels in the overall performance or not. The findings and results of this experiment are similar to our previous experiment without feature engineering. From Table 2, we can see, tuned support vector machine performs best amongst all the other models with high precision, recall, AUC score, and low log loss. Thus, we choose this model to predict on an unseen dataset. From Table 4, we notice the model performs satisfactorily on the test dataset as well. The accuracy hits almost 90% as depicted in Figure 4.

4.4. Trade off

The dimension of data is an important factor to consider on choosing a machine learning model. The high dimensionality of data increases the complexity of a model. Thus, feature engineering plays a vital role to overcome this issue. However, in some cases, feature engineering performs poorly and costs accuracy. Analysing Tables 1, 2, 3 and 4, we observe, there is no such difference on performances either we exploit feature engineering or not. There is a slight reduction of accuracy (as depicted in Figure 4 in feature engineering which is negligible. Thus, due to the capacity of dimension reduction, we decide to follow the feature engineering technique, which eventually keeps our model simple, accurate, and curtails the problem of overfitting.

Table 4: Performance Evaluation on Test Dataset (FE)

Algorithms	Classes	Precision	Recall	F1-score	Log Loss	AUC Score
Final Model (Tuned Support Vector Machine)	Macro Average	0.91	0.89	0.89	0.204	0.982
	Weighted Average	0.91	0.90	0.90		

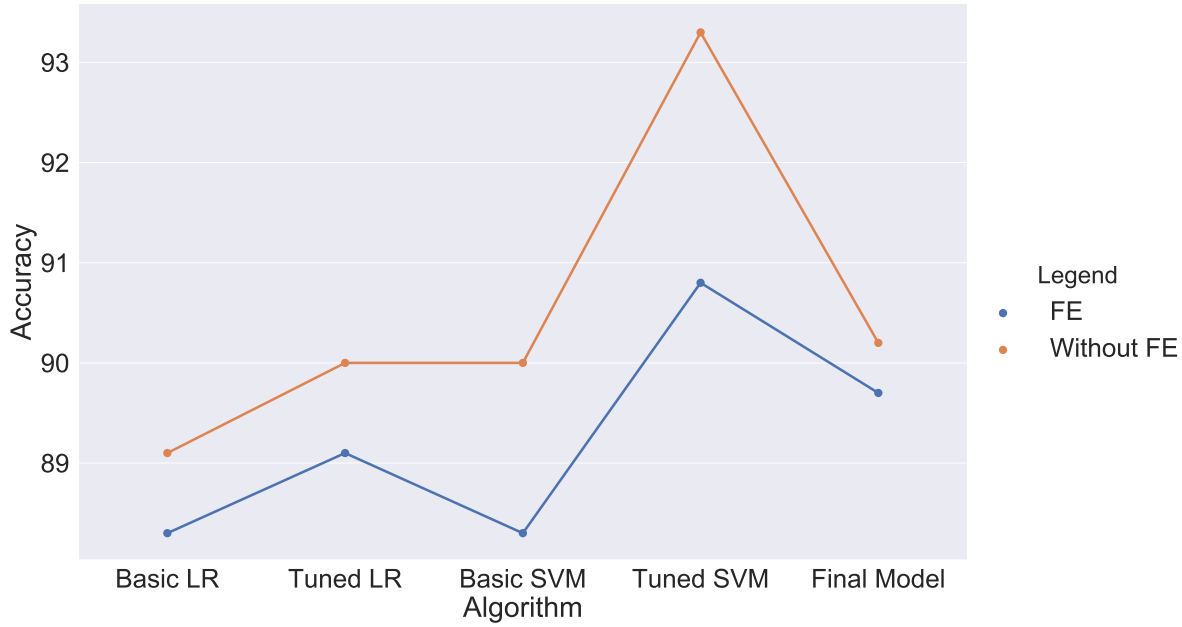

Figure 4: Accuracy.

5. Conclusion

Driven by the ease of communications in SN-IIoT platforms and the lack of any significant guidance to identify unhealthy contents among the contacts, we have introduced SPY-BOT as an ML-based post-filtering approach for social network behavioral analysis to mitigate the cyber hazards. More specifically, we have considered an automated post-filtering technique where the DU is able to regulate the indecent contents from the user's friends. Our most significant contribution has been the analyses of the behavioral polarity of users' contacts based on attributing contents. Herein we have mainly focused on converting an unlabeled dataset into a labeled dataset considering the behavioral analysis and attributions, and how machine learning models perform on the labeled dataset. It is worth mentioning that, all the experiments have been performed on a relatively small dataset. Exploration of different ML models, 'Big Data' concept and large datasets can potentially improve the post-filtering result and will serve as a future direction of this study.

References

[1] S. Siddiqui, T. Singh *et al.*, "Social media its impact with positive and negative aspects," *International Journal of Computer Applications Technology and Research*, vol. 5, no. 2, pp. 71–75, 2016.

[2] M. Fire, R. Goldschmidt, and Y. Elovici, "Online social networks: threats and solutions," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 4, pp. 2019–2036, 2014.

[3] S. Rathore, P. K. Sharma, V. Loia, Y.-S. Jeong, and J. H. Park, "Social network security: Issues, challenges, threats, and solutions," *Information sciences*, vol. 421, pp. 43–69, 2017.

[4] M. B. Yassein, S. Aljawarneh, and Y. A. Wahsheh, "Survey of online social networks threats and solutions," in *2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT)*. IEEE, 2019, pp. 375–380.

[5] S. Hopkins and J. Ostini, "Domestic violence and facebook: harassment takes new forms in the social media age," *The Conversation*, vol. 30, pp. 1–3, 2015.

[6] A. S. Vairagade and R. A. Fadnavis, "Automated content based short text classification for filtering undesired posts on facebook," in *2016 World Conference on Futuristic Trends in Research and Innovation for Social Welfare (Startup Conclave)*. IEEE, 2016, pp. 1–5.

[7] A. Rahman, S. N. Sadat, A. T. Asyhari, N. Refat, M. N. Kabir, and R. A. Arshah, "A secure and sustainable framework to mitigate hazardous activities in online social networks," *IEEE Transactions on Sustainable Computing*, 2019.

[8] M. A. Rahman, V. Mezhuyev, M. Z. A. Bhuiyan, S. N. Sadat, S. A. B. Zakaria, and N. Refat, "Reliable decision making of accepting friend request on online social networks," *IEEE Access*, vol. 6, pp. 9484–9491, 2018.

[9] V. Mezhuyev, S. N. Sadat, M. A. Rahman, N. Refat, and A. T. Asyhari, "Evaluation of the likelihood of friend request acceptance in online social networks," *IEEE Access*, vol. 7, pp. 75 318–75 329, 2019.

[10] K. Sahay, H. S. Khaira, P. Kukreja, and N. Shukla, "Detecting cyberbullying and aggression in social commentary using nlp and machine learning," *International Journal of Engineering Technology Science and Research*, vol. 5, no. 1, 2018.

[11] G. Sarna and M. Bhatia, "Content based approach to find the credibility of user in social networks: an application of cyberbullying," *International Journal Of Machine Learning and Cybernetics*, vol. 8, no. 2, pp. 677–689, 2017.

[12] Y. Chen, Y. Zhou, S. Zhu, and H. Xu, "Detecting offensive language in social media to protect adolescent online safety," in *2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*. IEEE, 2012, pp. 71–80.

[13] K. Das and S. K. Sinha, "A survey on user behaviour analysis in social networks," *International Journal of Computer Science and Information Security*, vol. 14, no. 11, p. 895, 2016.

[14] H. Mubarak, K. Darwish, and W. Magdy, "Abusive language detection on arabic social media," in *Proceedings of the First Workshop on Abusive Language Online*, 2017, pp. 52–56.

[15] M. Maia, J. Almeida, and V. Almeida, "Identifying user behavior in online social networks," in *Proceedings of the 1st workshop on Social network systems*. ACM, 2008, pp. 1–6.

[16] C. Wang, T. Bo, Y. W. Zhao, C.-H. Chi, K.-Y. Lam, S. Wang, and M. Shu, "Behavior-interior-aware user preference analysis based on social networks," *Complexity*, vol. 2018, 2018.

[17] F. Benevenuto, T. Rodrigues, M. Cha, and V. Almeida, "Characterizing user behavior in online social networks," in *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*. ACM, 2009, pp. 49–62.

[18] V. S. Chavan and S. Shylaja, "Machine learning approach for detection of cyber-aggressive comments by peers on social media network," in *2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*. IEEE, 2015, pp. 2354–2358.

[19] K. B. Kansara and N. M. Shekokar, "A framework for cyberbullying detection in social network," *International Journal of Current Engineering and Technology*, vol. 5, no. 1, pp. 494–498, 2015.

[20] H. Waheed, M. Anjum, M. Rehman, and A. Khawaja, "Investigation of user behavior on social networking sites," *PloS one*, vol. 12, no. 2, p. e0169693, 2017.

[21] R. Buettner, "Predicting user behavior in electronic markets based on personality-mining in large online social networks," *Electronic Markets*, vol. 27, no. 3, pp. 247–265, 2017.

[22] V. M. Deshpande and M. K. Nair, "A novel framework for privacy preserving ad-free social networking," in *2017 2nd International Conference for Convergence in Technology (I2CT)*. IEEE, 2017, pp. 139–145.

[23] D. Yu, N. Chen, F. Jiang, B. Fu, and A. Qin, "Constrained nmf-based semi-supervised learning for social media spammer detection," *Knowledge-Based Systems*, vol. 125, pp. 64–73, 2017.

[24] Q. Yan, L. Wu, and L. Zheng, "Social network based microblog user behavior analysis," *Physica A: Statistical Mechanics and Its Applications*, vol. 392, no. 7, pp. 1712–1723, 2013.

[25] S. Yousukkee, "Survey of analysis of user behavior in online social network," in *2016 Management and Innovation Technology International Conference (MITicon)*. IEEE, 2016, pp. MIT–128.
[26] D. H. Wolpert, "The lack of a priori distinctions between learning algorithms," *Neural computation*, vol. 8, no. 7, pp. 1341–1390, 1996.
[27] K. Kelley, B. Clark, V. Brown, and J. Sitzia, "Good practice in the conduct and reporting of survey research," *International Journal for Quality in health care*, vol. 15, no. 3, pp. 261–266, 2003.
[28] C. R. Turner, A. Fuggetta, L. Lavazza, and A. L. Wolf, "A conceptual basis for feature engineering," *Journal of Systems and Software*, vol. 49, no. 1, pp. 3–15, 1999.
[29] "Ordinaldata." [Online]. Available: https://stats.idre.ucla.edu/other/mult-pkg/whatstat/what-is-the-difference-between-categorical-ordinal-and-numerical-variables/
[30] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, 2019.
[31] H. Zhang, L. Zhang, and Y. Jiang, "Overfitting and underfitting analysis for deep learning based end-to-end communication systems," in *2019 11th International Conference on Wireless Communications and Signal Processing (WCSP)*. IEEE, 2019, pp. 1–6.
[32] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proceedings of the fifth annual workshop on Computational learning theory*, 1992, pp. 144–152.
[33] Y. Tian, M. Sun, Z. Deng, J. Luo, and Y. Li, "A new fuzzy set and nonkernel svm approach for mislabeled binary classification with applications," *IEEE Transactions on Fuzzy Systems*, vol. 25, no. 6, pp. 1536–1545, 2017.
[34] Y. Wang and J. L. Priestley, "Binary classification on past due of service accounts using logistic regression and decision tree," 2017.
[35] C. Ferri, J. Hernández-Orallo, and R. Modroiu, "An experimental comparison of performance measures for classification," *Pattern Recognition Letters*, vol. 30, no. 1, pp. 27–38, 2009.
[36] S. J. Lee, P. D. Yoo, A. T. Asyhari, Y. Jhi, L. Chermak, C. Y. Yeun, and K. Taha, "Impact: Impersonation attack detection via edge computing using deep autoencoder and feature abstraction," *IEEE Access*, vol. 8, pp. 65 520–65 529, 2020.
[37] M. A. Rahman, A. T. Asyhari, L. Leong, G. Satrya, M. Hai Tao, and M. Zolkipli, "Scalable machine learning-based intrusion detection system for iot-enabled smart cities," *Sustainable Cities and Society*, vol. 61, p. 102324, 2020.
[38] R. McGibbon, C. Hernández, M. Harrigan, S. Kearnes, M. Sultan, S. Jastrzebski, B. Husic, and V. Pande, "Osprey: Hyperparameter optimization for machine learning," *Journal of Open Source Software*, vol. 1, no. 5, p. 34, 2016.
[39] X. Dong, J. Shen, W. Wang, Y. Liu, L. Shao, and F. Porikli, "Hyperparameter optimization for tracking with continuous deep q-learning," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 518–527.