Journal of Network and Computer Applications 193 (2021) 103210

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

# A selective ensemble model for cognitive cybersecurity analysis

Yuning Jiang, Yacine Atif *

*School of Informatics, University of Skövde, Sweden*

ARTICLE INFO

*Keywords:*
Information security
Vulnerability analysis
Data correlation
Machine learning
Ensemble
Data mining
Database management

ABSTRACT

Dynamic data-driven vulnerability assessments face massive heterogeneous data contained in, and produced by SOCs (Security Operations Centres). Manual vulnerability assessment practices result in inaccurate data and induce complex analytical reasoning. Contemporary security repositories' diversity, incompleteness and redundancy contribute to such security concerns. These issues are typical characteristics of public and manufacturer vulnerability reports, which exacerbate direct analysis to root out security deficiencies. Recent advances in machine learning techniques promise novel approaches to overcome these notorious diversity and incompleteness issues across massively increasing vulnerability reports corpora. Yet, these techniques themselves exhibit varying degrees of performance as a result of their diverse methods. We propose a cognitive cybersecurity approach that empowers human cognitive capital along two dimensions. We first resolve conflicting vulnerability reports and preprocess embedded security indicators into reliable data sets. Then, we use these data sets as a base for our proposed ensemble meta-classifier methods that fuse machine learning techniques to improve the predictive accuracy over individual machine learning algorithms. The application and implication of this methodology in the context of vulnerability analysis of computer systems are yet to unfold the full extent of its potential. The proposed cognitive security methodology in this paper is shown to improve performances when addressing the above-mentioned incompleteness and diversity issues across cybersecurity alert repositories. The experimental analysis conducted on actual cybersecurity data sources reveals interesting tradeoffs of our proposed selective ensemble methodology, to infer patterns of computer system vulnerabilities.

## 1. Introduction

Current vulnerability assessment practices of critical infrastructures (CIs) rely on the central role of security operations centres (SOCs) staff (Feng et al., 2017). Yet, conventional manual and subjective audits do not meet the dynamic CIs characteristics and the expected attack surface assessment. For example, some standard vulnerability-severity scoring mechanisms like CVSS[1] calculator require manual input, which is based on qualitative judgements of vulnerability properties such as exploitability, scope and impacts (Joh and Malaiya, 2011). However, on average, SOC can quickly generate several gigabytes of security events per day, creating significant stress on human responders (Russo et al., 2019; Bhatt et al., 2014). Meanwhile, most existing vulnerability assessment techniques are based on scheduled analysis lifecycle, which can be weekly, or monthly. Such vulnerability scans can turn obsolete, leaving gaps between vulnerability exploit occurrences and the deployment of available patches (Shahzad et al., 2012).

Security practices involve checking the latest vulnerability reports from online public cybersecurity data sources like the Common Vulnerabilities and Exposures (CVE)[2] and the National Vulnerability Database (NVD).[3] However, the data quality of these online cybersecurity databases is affected by diversity, incompleteness and inconsistency issues (Dong et al., 2019), which hampers accurate vulnerability assessment practices (Jo et al., 2020). CVE is used as the primary source of vulnerability instances. However, making vulnerability-remediation decisions based solely on CVE records can be biased by missing or incomplete data (Christey and Martin, 2013; Anwar et al., 2020). The information regarding the same vulnerability instance is integrated into a common cross-linked structure to support vulnerability analysis. Mavroeidis and Bromander (2017). However, relevant cybersecurity items are stored as standalone traces in several separated enumeration lists, which increases the challenge to correlate heterogeneous vulnerability data (Dong et al., 2019). It has already been reported that some CVE reports are not scored, nor classified, which limit their mapping to

* Corresponding author.
*E-mail addresses:* yuning.jiang@his.se (Y. Jiang), yacine.atif@his.se (Y. Atif).

[1] https://www.first.org/cvss/
[2] https://cve.mitre.org/index.html
[3] https://nvd.nist.gov/

https://doi.org/10.1016/j.jnca.2021.103210
Received 7 June 2020; Received in revised form 5 June 2021; Accepted 27 August 2021
Available online 4 September 2021

threat models and reduce their mitigation effectiveness (Ladd, 2017). Recently reported instances are notably lacking such scoring valuation, yet these valuations are crucial to compute severity scores that would otherwise result in poor patching decisions (Householder et al., 2017). Dong et al. conducted quantitative analysis studies on vulnerable software version inconsistencies between CVE and NVD. Their experimental data include vulnerabilities reported from January 1999 to March 2018, which indicates that only 59.82% of the CVE summaries strictly match with the standardised NVD entries (Dong et al., 2019). Therefore, vulnerability disclosure in these repositories can be misleading in cybersecurity decisions about patching prioritisation (Ruohonen, 2019).

### 1.1. Problem statement

There is an increasing consensus among cybersecurity experts that machine learning (ML) techniques can support a continuous assessment of vulnerability data sources and hence alleviate some of the afore-mentioned cybersecurity challenges unfolding from the cyber threats' changing landscape (Almukaynizi et al., 2017). Advances in ML-induced security solutions are expected to shift the burden of managing large volumes of vulnerability information away from security experts and onto some digital alternatives. Recently published findings (Edkrantz and Said, 2015; Spanos et al., 2017; Bullough et al., 2017; Fang et al., 2020) report moderate successes in applying ML techniques to improve efficiency and productivity in security activities like cybersecurity alert management and related incident analysis. Nevertheless, ML techniques remain under-utilised to streamline various cybersecurity operations like vulnerability categorisation and related severity evaluation. For example, the multi-label and multi-class classification may require different ML techniques to improve prediction performances, and use different validation metrics (Aly, 2005; Tsoumakas and Katakis, 2007). Since there are no single-size-that-fits-all ML-based solutions to automate cybersecurity operations (Heelan, 2011), the selection of a suitable option becomes an overwhelming decision due to potential cybersecurity risks that may result from adopting the wrong ML model. The dilemma of choosing the most appropriate ML model that automatically alleviates the increasingly sophisticated threats is the problem addressed in this paper. The goal is to enhance situation awareness of cybersecurity specialists with up-to-date, diverse, and precise security indicators.

We adopt a data-centric methodology that blends security knowledge from several security data sources, in order to enable computers to learn from data in a similar way to humans. Hence our proposed cybersecurity framework integrates research thrusts from natural language processing, data mining, and machine learning, to streamline cybersecurity analysis while keeping security specialists in the analysis loop. To realise our framework, we embrace IBM's[4] cognitive cybersecurity inspirational definition to approach the cognitive perspective of cybersecurity. Thus, cognitive security augments human intelligence with artificial intelligence, to take advantage of the increasingly massive corpus of vulnerability information and reveal their hidden correlations (Andrade et al., 2019; Osifeko et al., 2020). Furthermore, our approach connects ML advances to cognitive sciences by considering a human agent in the cybersecurity analysis loop, to improve the performance of automation methods with human knowledge (Holzinger et al., 2019). For example, cybersecurity experts are solicited in the process of feature selection and validation-metric selection so that the generated ML models are tailored to problems in specific cybersecurity domains. Our approach empowers cybersecurity operators with ML techniques. This approach scouts constantly and optimises the utilisation of online cybersecurity data repositories. In doing so, we monitor the ICT (referring to Information and Communications Technology) assets of a given infrastructure to elicit their vulnerabilities' severity before damage occurs. The problem addressed in this paper can be formally specified as follows:

$$\mathcal{ML} = \underset{0<i\leq N}{Arg\,Max}(ML_i^D, S_p) \tag{1}$$

Given a set of vulnerability instances $D$ and a set of $N$ available ML baseline models $ML_i$ $(0 < i \leq N)$ (e.g. a support-vector-machine, or SVM model), our goal is to find an optimised model or ensemble of several baseline models $\mathcal{ML} = Arg\,Max(ML_i, 1 \rightarrow N, D, S_p)$, from an input of given classifiers $ML_i$ trained using $D$ labels, for analysing multiple targeted cybersecurity-analysis purposes $S_p$, such as categorising vulnerability severity or threat profile.

### 1.2. Research contributions

Fig. 1 illustrates the overall architecture and purpose of our proposed cognitive vulnerability analysis framework. The process starts from a web crawler module that continuously gathers data from heterogeneous online public vulnerability-alert repositories, and fuses retrieved data into a local cross-linked database. Subsequently, a pipeline of selective ML ensemble models produces classifiers according to the entered cybersecurity-issue metric sets such as vulnerability-severity metrics (e.g. confidentiality impact), or threat profile metric (e.g. denial of service), and so forth. This ML pipeline includes modules for data pre-processing and cleaning, feature engineering, and ML model selection. Our classifiers are trained using historical data to correlate new vulnerability instances into distinctive patterns. This approach addresses the aforementioned diversity issues and infers missing security information. Furthermore, our method embodies a ML ensemble construction built on top of trained classifiers to optimise the classification of security indicators. Instead of using a single ML model, our ensemble model combines multiple algorithms within a common knowledge structure that subsumes vulnerability patterns, as well as their correlations. Security operators are involved in the validation process to select and prioritise classification-performance metrics like accuracy and precision. End-to-end optimisation throughout the entire pipeline is challenging, yet we address the whole process realises the workflows of our approach to cognitive cybersecurity illustrated in Fig. 1. The process breakdown into modules features an optimisation opportunity to find out the best model for different tasks.

Subsequently, the main contributions of this paper include a novel cognitive cybersecurity framework based on (i) A localised database that synchronously integrates multiple online and heterogeneous cybersecurity repositories, (ii) A ML pipeline that correlates data together to optimise predictive analytics, and (iii) An application of ensemble methods to select the best classification model, as depicted further in Fig. 1, and elaborated further later in Fig. 2. The outcomes of our proposed cognitive security approach enable automated processes that provide a timely vulnerability detection and analysis.

More precisely, we suggest an optimisation algorithm that selects the best ML base algorithm(s) to construct effective ensemble models for diverse cybersecurity mission targets. We evaluate this approach through a comparative experimental study that contrasts our approach against multiple commonly used text-mining techniques in the context of two cognitive cybersecurity scenarios, namely vulnerability analysis and threat modelling, through a set of standard performance metrics. The evaluation benchmark of our proposed cognitive cybersecurity analysis approach involves an extensive database that comprises over 130 000 samples stemming from 8 actual online cybersecurity repositories and other corresponding manufacturer websites, as well. Our results show improved prediction performance compared to individual text-mining techniques, commonly employed in security management when analysing vulnerability reports. For example, we may choose LSTM-ANN (Long Short-Term Memory based Artificial Neural Network) (Zhou et al., 2016) and MLP (Multi-layer Perception Classifier) (Zanaty, 2012) as base learners for threat classification ensemble,

[4] https://www.ibm.com/se-en/security/artificial-intelligence

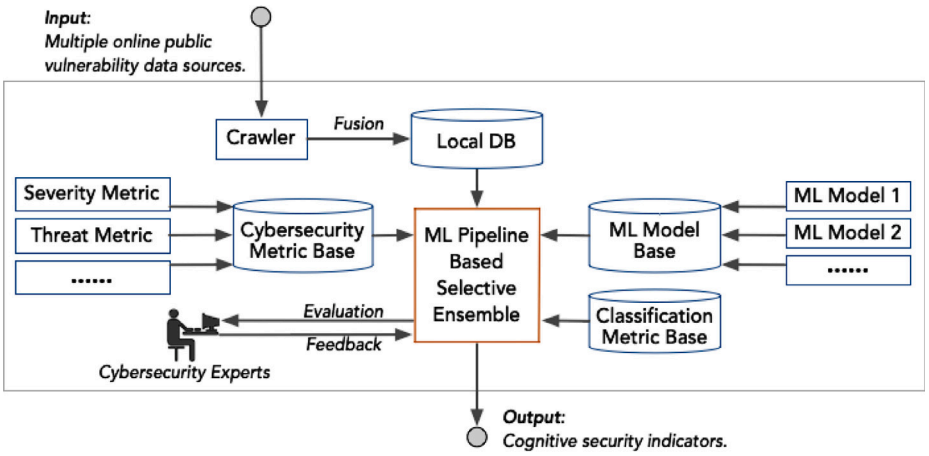


**Fig. 1.** Cognitive cybersecurity.

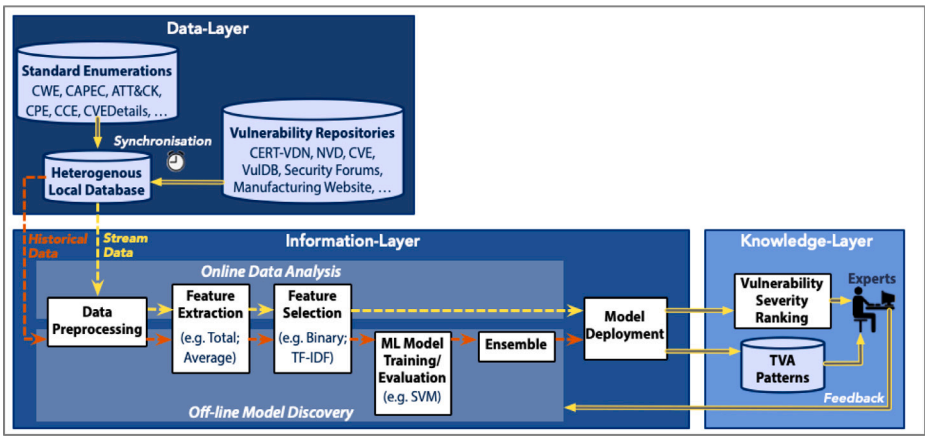


**Fig. 2.** Machine-learning pipeline based cybersecurity knowledge generation.

while we may choose LSTM-ANN, NBSVM (Naïve Bayes, Wang and Manning, 2012, based Support Vector Machines) (Joachims, 2001) and MLP as base learners for vulnerability-severity score computation ensemble. The optimisation algorithm explores all combination schemes for selecting best-performing ML-based instances.

The remainder of this paper is organised as follows. Section 2 provides some relevant background and outlines some related works in data-driven methods, information-fusion techniques and ML models applied to cyber-security problems. We also discuss and contrast in this section similar ensemble techniques with our proposed approach used in this paper. The proposed ML pipeline architecture and the related ensemble optimisation algorithm are revealed in Section 3. Section 4 rolls out the experiments settings and associated parameters to explain the design and methodology of our experiments. We also show in this section the performance tradeoffs induced by our selective ensemble technique to validate our proposed cognitive cybersecurity approach. Finally, Section 5 concludes the paper with some final remarks summarising this study and some future work suggestions to extend it further.

## 2. Background and related works

Assessing cybersecurity vulnerabilities supports analytics-based decision-making processes to protect computing infrastructures. The goal is to focus attention and investment on specific acute risks arising from threat-exploitability with varying degrees of impact-severity.

### 2.1. Cybersecurity data

Data collection and correlation provides a basis for further vulnerability analytics empowered by the application of ML techniques. Databases like CVE repository accumulate vulnerability reports for around 20 years that could provide a basis for vulnerability-trends analysis (Na et al., 2016; Russo et al., 2019). Tools like cve-search[5] (referring to a web interface and API, or application programming interface to CVE) and Open Vulnerability and Assessment Language (OVAL)[6] are some of the common techniques used to identify and manage organisation vulnerabilities in a vendor-independent environment. These tools led to streamlined approaches to retrieve vulnerabilities from public repositories in support of testbed systems (Siboni et al., 2019). Furthermore, correlation studies between multiple vulnerability data sources have been undertaken to combine different perspectives from security stakeholders (Sauerwein et al., 2019; Jiang et al., 2019), to connect related analysis into broader statistical associations. For instance, one study by Allodi and Massacci (2014) correlates NVD to data sources like ExploitDB,[7] and Symantec (SYM) AttackSignature and ThreatExplorer. Another study by Geer and Roytman (2013) also correlates the NVD database to ExploitDB and Metasploit to support penetration testers. They highlight that statistical interpretations of CVE datasets need to be combined with other live security-related data

[5] https://www.circl.lu/services/cve-search/
[6] https://oval.mitre.org/
[7] https://www.exploit-db.com/

sources, including product vendors across deployed infrastructures, to raise indicators' reliability and precision. However, further research that incorporates heterogeneous data from different databases while providing structured and simplified indicators is limited (Dong et al., 2019).

One of the aims of this paper is to enable context-aware data analysis that aids situation awareness. We address the aforementioned problem by categorising multiple online cybersecurity resources into semantically integrated cybersecurity instances and enumerations, using CVE-IDs as unique references. Data obtained from crawling these cybersecurity-alert repositories, as well as manufacturer websites, are condensed into relevant views. These mined indices support security operators' decision-making processes. Thus, we empower the cognitive assets of security operators involved at various SOC levels with a localised and synchronised database empowered by our proposed ML pipeline. We also reconcile inconsistent vulnerability-severity scores from different sources before adopting them as ground truths for the proposed ML pipeline to enhance the accuracy of further vulnerability analytics.

### 2.2. Artificial intelligence for cybersecurity

The retrieved textual information from cybersecurity data sources provides grounds for trend analysis and supports further pattern recognition, whereby Artificial Intelligence (AI) techniques harness a large amount of available open-source vulnerability data. Pattern-based methods are commonly used for vulnerability categorisation and assessment (Andrade et al., 2019; Torres et al., 2019). In these studies, feature-engineering approaches are used to extract textual descriptors that distinguish vulnerabilities. These methods are also employed to detect anomalies and to generate a vectorial- or a graphical representation of vulnerability attributes used in pattern recognition processes. Among them, Term Frequency-Inverse Document Frequency (TF-IDF) is widely used to identify important terms in vulnerability reports. In our approach, we combine these natural language processing (NLP) (Zhu and Dumitraş, 2016) techniques together with some features defined by cybersecurity experts. In doing so, we extract vulnerability features and transform unstructured text into normalised and structured data.

Text-mining techniques have been widely used to collect cyber threat intelligence from cybersecurity repositories and technical blogs (Liao et al., 2016), to retrieve attack patterns (Husari et al., 2017), as well as to assess vulnerability severity (Khazaei et al., 2016; Spanos et al., 2017). For instance, Neuhaus and Zimmermann apply a topic model based on LDA (referring to latent dirichlet allocation) to analyse vulnerability trends using reports from CVE (Neuhaus and Zimmermann, 2010). These models use different classification algorithms or clustering approaches, such as *Bayesian network* (Holm et al., 2015), *Support Vector Machines (SVM)* (Patil, 2017), etc,. For example, Scandariato et al. adopt bag-of-words for feature extraction and employ Naïve Bayes and Random Forest classifiers for software-vulnerability classification and prediction (Scandariato et al., 2014). Bozorgi et al. mine vulnerability disclosure reports from OSVDB (referring to Open Source Vulnerability Database) and CVE to rank these vulnerabilities in terms of exploitability. They train a SVM classifier using the exploit-classification status reported in OSVDB as ground truth (Bozorgi et al., 2010). Bullough et al. evaluate performances of several prior ML models, including the SVM model from Bozorgi et al. (2010), to quantify the influence of class imbalance, as well as how training and testing datasets are divided. They extract training data from NVD and use proof-of-concept (PoC) exploit in ExpoitDB as ground truths (Bullough et al., 2017).

However, existing AI-based methods are usually driven by specific cybersecurity problems like detecting exploits. Yet, a thorough vulnerability assessment needs to consider various cybersecurity aspects, such as vulnerability categorisation, severity evaluation, and exploit prediction, which requires different learning models. Different models perform diversely for various tasks. For example, LR (Logistic Regression) (Almukaynizi et al., 2017; Zhang et al., 2011) algorithm does not require heavy computing sources, but its prediction accuracy may lag behind compared to more CPU-intensive Neural Networks (Zhou et al., 2016) training. Moreover, there are no one-size-fits-all in computational intelligence methods whereby classification or clustering tasks could be handled uniformly (Torres et al., 2019; Oprea et al., 2018). This is especially true in vulnerability analysis, considering the difficulty of related properties' evaluation and subsequent classification imbalance or bias (Sommer and Paxson, 2010).

Ensemble methodology achieves better performance at classifying malicious and benign connections compared to linear ML models, and provides a potential solution to address the challenges in individual ML approaches (Lower and Zhan, 2020; Tong et al., 2018). Besides, ensemble approaches have built-in techniques that can handle class imbalance as well. Naturally, the ensemble ML domain has a long-standing tradition, and many interesting references are worth mentioning when considering the application and evaluation of ensemble ML models (Resende and Drummond, 2018). Vanerio and Casas suggest a supervised learning method named Super Learner to find the optimal ensemble model for anomaly detection (Vanerio and Casas, 2017). Li et al. developed a hybrid intrusion detection technique using binary classification and k-nearest neighbour (KNN) classifiers (Li et al., 2017). Rajagopal et al. provide an ensemble paradigm based on meta classification and stacking generalisation, while targeting an enhanced predictive accuracy of network intrusions (Rajagopal et al., 2020). Fang et al. employ LightGBM algorithm as the ensemble technique to predict exploits of vulnerabilities (Fang et al., 2020). These studies have demonstrated the application of ensemble approaches to improve model performances like prediction accuracy. However, there are limited efforts in utilising ensemble methods to fulfil multiple cybersecurity-analysis purposes.

Identifying base ML algorithms that can perform the classification task is crucial to construct effective ensemble models (Dietterich, 2000). It is also critical to piecing together an appropriate combination scheme for selected base ML algorithms (Onan et al., 2016). The most common methods for ensemble construction are voting (Kittler et al., 1998), bagging (Breiman, 1996), boosting (Freund and Schapire, 1995) and stacking (Wolpert, 1992) methods. Normally, both bagging and boosting techniques require the initial dataset to be large enough to capture most of the complexity of the underlying distribution, so that sampling from the dataset is a good approximation of the real distribution. Larcher et al. apply adaptive ensemble methods to enhance the trade-off between cybersecurity requirements and system efficiency (Larcher et al., 2019). These adaptive methods usually use weighted voting techniques which diversify attribute weights across classifiers depending on selected metrics. This process can be further improved by taking into considerations outperforming metrics for more adaptive predictions (Quintal et al., 2020).

Our ensemble approach combines independent classifiers that use information from different sources. This approach yields better performances for cybersecurity classification tasks, while improving flexibility in handling diverse cybersecurity analysis missions. A variant of cross-validation is adopted to minimise possible over-fitting across classification tasks (Van der Laan et al., 2007). Our cybersecurity framework combines different sets of evaluation metrics to select ensemble ML that optimises vulnerability severity-score computation. The classification validity for these selected ensemble ML classifiers is asserted in the subsequent performance evaluation section.

### 2.3. Cognitive cybersecurity

In recent years, cognitive modelling has been employed in cybersecurity analysis, experiments, and simulations to address human participation in effective decision-making when keeping computational infrastructures secure (Veksler et al., 2020; Andrade and Yoo, 2019).

The application of cognitive science in cybersecurity investigates relationships between human security experts' experience with related procedures and practices, that involve the analysis of security data sources like alert reports and related blogs.

In this paper, cognitive methods are used to set up a framework involving and empowering security actors. Some relevant related works have been contributed by Andrade and Yoo (2019), who developed a three-layer (knowledge, information and cognitive) framework considering four basic cognitive-security components, that are process, technology, knowledge and cognitive skills (Andrade and Yoo, 2019). Their goal was to integrate multiple cognitive theories and models to support cybersecurity situation awareness. In our works, however, we elaborate a cognitive framework to enhance the process of perception and comprehension of AI-based vulnerability-analysis models. In doing so, we utilise security experts' feedbacks in our proposed ML pipeline loop to improve predictive vulnerability categorisation outcomes.

## 3. Cognitive cybersecurity framework

Next, we discuss our approach that employs ML techniques to streamline cybersecurity knowledge transfer. In this section, we describe the design of our framework and the related methods in a step-by-step manner.

### 3.1. Overview

Our approach is composed of three steps that connect unstructured data collections to meaningful information leading to cognitively meaningful indicators and know-how at the knowledge layer level, as illustrated in Fig. 2. First, we periodically collect heterogeneous data from online cybersecurity repositories. As part of the process, we label related enumerations based on standard cybersecurity-related categorisations and fuse them into a common localised database. Secondly, we employ a ML pipeline to utilise human-interpretable text-classification models (Liao et al., 2020) to produce relevant threat, vulnerability and attack (TVA) patterns as well as risk severity ranking, based on data retrieved from the constructed localised database. Finally, we empower security experts at the next knowledge layer level to evaluate the machine-generated indexes and consider their feedback back into the offline training process. We focus on the first two steps in this paper.

### 3.2. Data acquisition and integration

Security-related data can usually be obtained in two ways: (1) direct access using software like vulnerability scanner, and (2) indirect access using existing online public datasets. Direct data collection approaches are suitable for short-term analytics or a comparatively small amount of collected data. In contrast, indirect data collection works better for a long-term endeavour or a comparatively large amount of data saved by acquisition time and storage costs (Xin et al., 2018). There exist other indirect offline accesses to security datasets. However, these operations usually require stakeholders' authorisation or may have ethical concerns. Therefore in this paper, we employ online public datasets. However, these information items are separated, stored and published in different data formats. Furthermore, they obey different standards, each of which has its own syntax and semantics. Therefore, the different data sources employed in our work can be categorised into two semantic categories considering their co-occurrence:

- Vulnerability Repositories: CVE, NVD, and CERT databases have been recognised as standard resources for security analysis that publish vulnerability reports regularly. Vulnerability documents could be found typically with a formal identifier, namely CVE ID. Additional available sources of security-related data could be gathered in online forums such as ExploitDB, SecurityFocus, SecurityTracker, manufacture websites, as well as third party cybersecurity analysers like ICS CERT.
- Standard Enumerations: These standards include respectively a taxonomy from Common Weakness Enumeration (CWE), threat categorisation from the website *cvedetails.com* and attack patterns from Common Attack Pattern Enumeration and Classification (CAPEC), as well as Adversary Tactics, Techniques and Common Knowledge (ATT&CK)). Besides, security analysts need to match a current system configuration with retrieved vulnerability reports from enumerations like vulnerable product dictionary from Common Platform Enumeration (CPE) and recommended secure system configurations from Common Configuration Enumeration (CCE). Additional system-configuration information like component version and package build-number can be gathered by crawling manufacturer websites.

To tackle the heterogeneity of such data, we employ data-fusion techniques to capture data from multiple cybersecurity repositories and enumerations. Then we fuse condensed information into one localised NoSQL database based on MongoDB[8] documents structure, as illustrated in the Data Layer of Fig. 2. We directly downloaded data seeds from CVE and NVD repositories. Then we crawled vulnerability reports from other cybersecurity data sources like SecurityFocus. In addition, we also define higher-level object classes by extracting keywords with the highest occurrence in CVE reports using open-source LDA-based topic-modelling techniques (Merrouni et al., 2019). In doing so, we match retrieved vulnerability instances to the selected object classes with attributes, specialisations and instantiations using an ontology (Angelini et al., 2018). For example, we extract top keywords from all the vulnerability reports recorded for a desired period, e.g. 2002–2019, by sorting keywords in descending order of occurrence. For instance, we pick the ones with highest occurrence, such as *allow*, *remote*, *execute*, *script*, etc., among around 2000 automatically generated keywords. Then we further categorise them into object-classes to represent these keywords and their correlations using ML techniques described next, in order to recognise security patterns between different directions such as *vulnerability*, *threat*, *attack*, *remediation*, and *system*.

### 3.3. Information alignment pipeline

From the end of the previous step, we could identify data poised to trigger correlation instances. Then we employ a ML pipeline that classifies the retrieved instances considering threat-, vulnerability-, and attack-labels. As the word "pipeline" suggests, it is a series of steps chained together by integrating ML cycles, where each cycle involves obtaining the data, pre-processing it, training/testing it on a ML algorithm and finally obtaining some output (in the form of a prediction). Our proposed machine pipeline apply text-mining (Kowsari et al., 2019) and NLP techniques (Zhu and Dumitraş, 2016) to fill up missing information that is prevalent in vulnerability reports, as well as predicts cybersecurity trends.

To classify new vulnerability instances, we first conduct some data pre-processing like tokenisation and feature extractions on the descriptions of extracted vulnerability instances, as illustrated in Fig. 3.

*Data preprocessing*

The process starts by preprocessing and normalisation of retrieved raw textual reports from the abovementioned repositories to clean them. This includes traditional text processing steps such as tokenisation, removing unnecessary punctuations and stop-words, word-stemming and word-lemmatisation (or grouping together different forms of a word into a common item using some dictionary).

[8] https://www.mongodb.com/

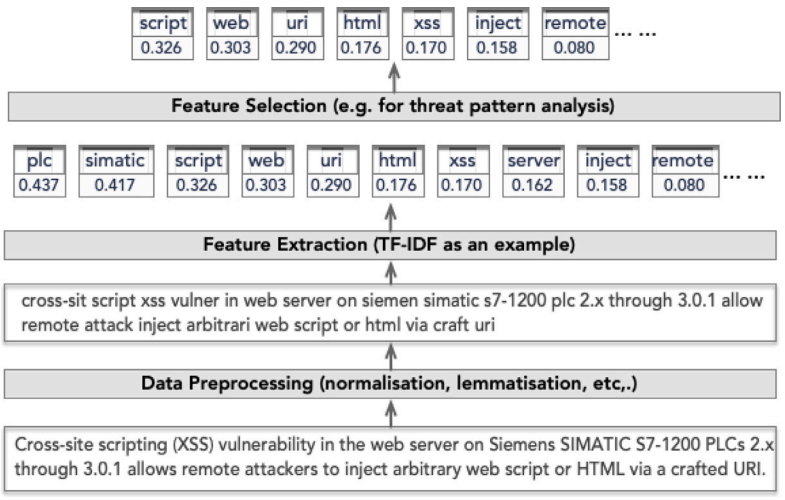


**Fig. 3.** Information alignment pipeline example.

**Table 1**
Data features, sources, types and modelling.

| Feature | Feature source | Feature Type | Feature modelling |
|---|---|---|---|
| Vulnerability description | CVE, SecurityFocus, etc. | Text | Word and N-gram character embedding |
| Vulnerable component name | *CVE, CPE* | Text | Word embedding into vectors |
| Vulnerable component version | *CVE, CPE* | Number | Numeric feature embedding |
| Vulnerable component type | CPE | Category | One-hot encoding |
| Vulnerable component vendor | *CVE, CPE* | Text | Word embedding into vectors |
| Weakness category ID | CWE | Number | One-hot encoding |
| Weakness category name | CWE | Text | Word and N-gram character embedding |
| Weakness category description | CWE | Text | Word and N-gram character embedding |
| Attack category ID | CAPEC | Number | One-hot encoding |
| Attack category name | CAPEC | Text | Word and N-gram character embedding |
| Attack category description | CAPEC | Text | Word and N-gram character embedding |
| Threat category | CVE Details | Category | One-hot encoding |
| Access vector (Exploitability) | CVSS V2 | Category | One-hot encoding |
| Access complexity (Exploitability) | CVSS V2 | Category | One-hot encoding |
| Authentication (Exploitability) | CVSS V2 | Category | One-hot encoding |
| Confidentiality impact | CVSS V2 | Category | One-hot encoding |
| Integrity impact | CVSS V2 | Category | One-hot encoding |
| Availability impact | CVSS V2 | Category | One-hot encoding |
| CVSS V2 base score | CVSS V2 | Number | Scaled to (0,10) |
| Extra reports | Manufacture websites; Security blogs; etc. | Text | Word and N-gram character embedding |

*Feature extraction*

From cleaned textual reports, we further extract indirect features and direct features. Indirect features include numerical counts of words and unique words. From these extracted features, we create a vocabulary and a term-document matrix. Each column in the matrix represents a word in the vocabulary, while each row represents a document in our dataset. The values, in this case, are the word counts (i.e. indirect feature). Direct features are the ones inherently derived from words or sentences, such as word frequency and vector distance mapping of words (e.g. Word2Vec). Three ways are adopted to extract direct features, namely Count-based, Binary-based, and TF-IDF based feature weighting approaches (Trstenjak et al., 2014). Count-based feature extraction assigns weights based on the term-frequency of words in a document. Binary-based weighting assigns '1' or '0' to a word depending on whether the word is present in a document or not. TF-IDF based weighting approach assigns more weight to the words that are unique to a particular document than the words that are commonly used across documents (Debole and Sebastiani, 2004). TF-IDF is usually employed to extract important keywords from a document to understand what characterises that document. We also summarise the data features, feature sources, feature types as well as feature modelling in Table 1. Taking the vulnerability description as an example, we capture textual features and translate them into vectors and embeddings that represent the similarity between words and n-gram characters. Another example is the numeric feature extracted from CPE to indicate vulnerable component type, namely hardware-, software-, and operating-system categories. We model such numeric feature through a one-hot encoded vector, used to distinguish each word in the document from every other word.

*Feature selection*

Our goal here is to select the most relevant features of our problem. Instead of evaluating all the available candidate features extracted from the previous step, we compare the candidate features through preprocessing the original feature space and rank these features based on metrics such as processing time. Furthermore, features are assigned with different weights according to the intended cybersecurity analysis. For example, *script, web, uri, html, xss* are given higher feature weights than *plc, simatic* when we target threat-pattern analysis, as shown in Fig. 3.

*Machine learning model training*

Previously, we proposed to utilise some NLP methods to convert cybersecurity reports' content into a numerical format (Zhu and Dumitraş, 2016). Simultaneously, we select and train ML algorithms with extracted features. The goal is to apply ML techniques to classify new incoming reports based on historical observations, in order to extract relevant TVA patterns and to derive confident predictions of vulnerability score severity. The main challenge is the appropriate

**Table 2**
Adopted Classification Performance Metrics.

| Metric | Binary-class evaluation | Multi-class single-label evaluation | Multi-label evaluation |
|---|---|---|---|
| | (TP is true positive; TN is true negative; FP is false positive; FN is false negative; $p_o$ is the observed agreement; $p_e$ is the expected agreement; C is the number of classes) | | (k is the number of labels; n is the number of instances; $Y_i$ is the ground truth vector of labels for instance $x_i$; $Z_i$ is the predicted label vector for instance $x_i$; $h$ is the multi-label classifier; $I$ is the indicator function) |
| Accuracy | $ACC = \frac{TP+TN}{TP+TN+FP+FN}$ | $maACC = \frac{\sum_{i=1}^{C} \frac{TP_c+TN_c}{TP_c+TN_c+FP_c+FN_c}}{C}$<br>$baACC = \frac{1}{C}\sum_{c=1}^{C} \frac{TP_c}{TP_c+FN_c}$ | $mlACC = \frac{1}{n}\sum_{i=1}^{n} \frac{\lvert Y_i \cap Z_i \rvert}{\lvert Y_i \cup Z_i \rvert}$ |
| Precision | $PRE = \frac{TP}{TP+FP}$ | $maPRE = \frac{\sum_{c=1}^{C} \frac{TP_c}{TP_c+FP_c}}{C}$<br>$miPRE = \frac{\sum_{c=1}^{C} TP_c}{\sum_{c=1}^{C}(TP_c+FP_c)}$ | $mlPRE = \frac{1}{n}\sum_{i=1}^{n} \frac{\lvert Y_i \cap Z_i \rvert}{\lvert Z_i \rvert}$ |
| Recall | $REC = \frac{TP}{TP+FN}$ | $maREC = \frac{\sum_{c=1}^{C} \frac{TP_c}{TP_c+FN_c}}{C}$<br>$miREC = \frac{\sum_{c=1}^{C} TP_c}{\sum_{c=1}^{C}(TP_c+FN_c)}$ | $mlREC = \frac{1}{n}\sum_{i=1}^{n} \frac{\lvert Y_i \cap Z_i \rvert}{\lvert Y_i \rvert}$ |
| F1 | $F1 = \frac{2*PRE*REC}{PRE+REC}$ | $maFS = \frac{2*maPRE*maREC}{maPRE+maREC}$<br>$miFS = \frac{2*miPRE*miREC}{miPRE+miREC}$ | $mlFS = \frac{1}{n}\sum_{i=1}^{n} \frac{2\lvert Y_i \cap Z_i \rvert}{\lvert Y_i \rvert + \lvert Z_i \rvert}$ |
| HammingLoss | HLS = 1-ACC | (Same as Hamming Distance) | $mlHLS = \frac{1}{kn}\sum_{i=1}^{n}\sum_{l=1}^{k}[I(l \in Z_i \wedge l \notin Y_i) + I(l \notin Z_i \wedge l \in Y_i)]$ |
| Cohen's Kappa | | $\frac{p_o-p_e}{1-p_e}$ | |

ML selection dilemma, as stated earlier. Thus, to solve this problem, we train a set of ML models, test and validate these ML models, and select well-performing models as component models for further ensemble construction. To do so, we apply a simple model of data distribution parallelism to balance statistical efficiency and hardware efficiency, following the framework suggested in Jiang et al. (2018). The training data (referring to historical vulnerability instances and reports) is partitioned. Each partition (or certain fields of the partition) is used for a ML model training. Meanwhile, model replicas are used to update and store parameters. This way, multiple ML models are trained and tested offline. Once the training and testing processes are complete, metadata and learnt models' configuration parameters (e.g. the embedding dimensions, word-occurrence threshold, etc.) are saved in the candidate model database.

### 3.4. Candidate model evaluation

We distinguish different validation metrics for various classification tasks, namely binary classification tasks, multi-class classification tasks, multi-label classification tasks, as well as CVSS score evaluation metrics. In binary classification, one instance can be classified into one of two categories. While in multi-class classification, one instance can be classified into one of multiple (more than two) categories. In multi-label classification, one instance can be classified with more than one label, referring to multiple categories (Tsoumakas and Katakis, 2007). The evaluation measures for single-label are usually different than for multi-label. In multi-label classification, a misclassification is no longer a hard wrong or right (Sorower, 2010). A prediction containing a subset of the actual classes should be considered better than a prediction that contains none of them, i.e., predicting two of the three labels correctly is better than predicting no labels at all. Therefore, prediction of multi-label instances can be fully correct, partially correct with different levels of correctness, or fully incorrect.

Here we choose the multi-label accuracy (mlACC) evaluation as an example to illustrate the validation process of the candidate ML models. The prediction performance $P^{mlACC}$ of the candidate ML model is generated through Algorithm 1. Let $D$ be a k-label dataset with $n$ instances $(x_i, Y_i)$ where $Y_i \in Y = \{0,1\}^k$ is the ground truth vector of labels for $i$th sample with a label-set $|L| = k$. $h$ is a multi-label classifier with $Z_i = h(x_i) = \{0,1\}^k$ be the set of predicted label memberships for instance $x_i$. Given a set of $M$ related evaluation metrics $m_j$ $(0 < j \leq M)$, we can further generate a performance vector $P=[P^{m_1}, \ldots, P^{m_j}, \ldots, P^{m_M}]$ for the candidate ML model.

**Algorithm 1** Performance Evaluation for Multi-Label Accuracy

**procedure** EVALUATEPERFORMANCE(ML, m)
▷ $ML$ is a candidate machine learning model
▷ $m$ is a given machine-learning metric, which is illustrated below:
▷ $m \leftarrow$ Multi-Label Accuracy metric (i.e. mlACC)
▷ Considering a K-label ground truth dataset $\mathcal{G}$ with total instances $D$, where:
  $D = |\{(Y_i, Y_j) \in \mathcal{G}, Y_j \in \{0,1\}^K\}|$
▷ $Z$ is the resulting $ML$ labelled data set:
  $Z = \{(Z_i, Z_j) \in ML, Z_j \in \{0,1\}^K\}$

  **For** $j = 1, \ldots, D$ **do**
    **For** $k = 1, \ldots, K$ **do**
      **if** $Z_{(j,k)} == Y_{(j,k)}$ ς $Y_{(j,k)} == 1$ **then**
        $S_j^{ACC} += 1, S_j^{Total} += 1$
      **else**
        **if** $Z_{(j,k)} != Y_{(j,k)}$ **then**
          $S_j^{Total} += 1$
        **End if**
      **End if**
    **End For**
    $S_j^{mlACC} = \frac{S_j^{ACC}}{S_j^{Total}}$
  **End For**
  $P^{mlACC} = \frac{1}{D}\sum_{i=1}^{D} P_j^{mlACC}$
**End procedure**

Next we discuss different evaluation metrics. The details of the metric computing are listed in Table 2.

*Binary-class evaluation metrics*

We employ a confusion matrix to validate the correctness of a classification. This is performed by computing: (i) the number of correctly classified class instances (true positives, or TP), (ii) the number of correctly classified instances that do not belong to the class (true negatives, or TN), and (iii) instances that are incorrectly assigned to the class (false positives, or FP), and (iv) instances that are not recognised as class instances (false negatives, or FN). Based on the above definitions,

we further compute measurements like accuracy, precision, recall (or sensitivity) and F-score (or F1).

Accuracy represents the overall effectiveness of a classifier. Precision, on the other hand, describes the percentage of TP in all the predicted positives. Recall shows the effectiveness of a classifier to identify positive labels. Finally, F-score maintains a balance between precision and recall, respectively. The higher the value of accuracy, precision, recall and F-score, the better the performance of the learning algorithm.

*Multi-class single-label evaluation metrics*

Here we apply both macro-average and micro-average validation. Macro-average evaluation of an C-class ($C > 2$) classification problem is given by the average of per class evaluation, to compute macro-accuracy (maACC), macro-precision (maPRE), macro-recall (maREC), and macro F-score (maFS). Alternatively, micro-average classification is evaluated by summing up the amounts of TP, FN, TN, and FP to aggregate the contributions of all classes to compute the average metric. When using micro-average metrics, the value of micro-precision (miPRE), micro-recall (miREC) and micro F-score (miFS) are the same. We also employ Cohen's Kappa to indicate how the adopted classifier is performing over the performance of a classifier that makes random classification assumptions (Sokolova and Lapalme, 2009; Aly, 2005)

*Multi-label evaluation metrics*

For multi-label classification, we use different metric formulations for accuracy, precision, recall, and F-score (Sorower, 2010). For each instance, multi-label accuracy (mlACC) is defined as the proportion of the predicted correct labels to the total number (both predicted and the ground-truth) of labels for that instance. Multi-label precision (mlPRE) is the proportion of predicted correct labels to the number of predicted labels. Multi-label recall (mlREC) is the proportion of predicted correct labels to the number of ground-truth labels. Multi-label F-score (mlFS) is the harmonic mean of mlPRE and mlREC. The exact-match-ratio (EMR) calculates the percentage of samples that have all the labels correctly predicted.

Another metric used in our evaluation of security data classification is the hamming-loss (mlHLS) metric. This metric is the fraction of incorrectly predicted labels, including both the prediction error (an incorrect label is predicted) and the missing error (a relevant label is not predicted). We define $I$ as the indicator function, and assuming $Y_i$ is the ground truth and $Z_i$ is the prediction. The lower the value of the hamming loss, the better the performance of the classifier.

*CVSS score evaluation metrics*

The standard mechanism CVSS is widely used to support quantitative vulnerability-severity assessment in both academic research (Khazaei et al., 2016; Johnson et al., 2016; Spanos et al., 2017) and security-critical industrial domains (Stine et al., 2017). As part of our validation, we compare predicted severity labels to their original severity label counterparts, and apply the evaluation metrics of accuracy to evaluate the ML model performance. To contrast further our comparative study, we used an alternative scores' distance metric to evaluate the performance of CVSS-characteristic classification across ML models. First, we compute CVSS scores by applying CVSS mechanism onto predicted counterparts (Scarfone and Mell, 2009). Then we represent an existing score (true score, or TS) and a predicted score (or PS) as a two-dimensional vector space into one Cartesian coordinate system. The distance of the two vectors is used to measure the performance of our vulnerability score prediction system using Eq. (2), where $\delta$ is a threshold value. In our experiment that we present later, we use two values for $\delta$ (0.5 or 0.05) to evaluate the accuracy of our severity-score computing.

$$ScoreAccuracy_{macro} = \frac{\sum_{i=1}^{n} \{i | \frac{TS_i - PS_i}{TS_i} < \delta\}}{n} \quad (2)$$

### 3.5. Model selection and ensemble

The selected baseline component models construct an ensemble model to aggregate the predictions of each component model, and derive a final prediction of the risk level corresponding to a single reported vulnerability. Next, we reveal further details of the proposed Ensemble construction technique.

Given dataset $D$, $N$ ML models $ML_i$ $(0 < i \leq N)$, and a set of $M$ related evaluation metrics $m_j$ $(0 < j \leq M)$, the following algorithmic steps construct the base ensemble of classifiers. For $N$ individual models, we conduct $N$ rounds of training tasks, and construct $[\binom{N}{1}, \binom{N}{2}, \binom{N}{3}, \ldots, \binom{N}{N-1}, 1]$ amount of ensemble models in each round. For example, in the first round, we construct $N$ ML ensemble models, and each ensemble has only one base classifier. In the second round, we construct $\binom{N}{2}$ ensemble models, and each ensemble has two base classifiers.

- Step 1: In the first round, we train every given individual ML model $ML_i^{(1)}$ $(0 < i \leq N)$ with the dataset $D$. Then, we measure the performance metric values for each resulting classifier using every input metric $m_j$ $(0 < j \leq M)$. At this stage, we have a vector of $N$ prediction performances $[P_1^{(1)}, \ldots, P_i^{(1)}, \ldots, P_N^{(1)}]$ corresponding to the first round of the framework algorithm, where the prediction performance $P_i^{(1)}=[P_{i,1}^{(1)}, \ldots, P_{i,j}^{(1)}, \ldots, P_{i,M}^{(1)}]$. Therefore, the overall first round produces the following prediction performance $matrix^{(1)}$:

$$\begin{bmatrix} P_{1,1}^{(1)} & P_{1,2}^{(1)} & \ldots & P_{1,M}^{(1)} \\ P_{2,1}^{(1)} & P_{2,2}^{(1)} & \ldots & P_{2,M}^{(1)} \\ \ldots & \ldots & \ldots & \ldots \\ P_{N,1}^{(1)} & P_{N,2}^{(1)} & \ldots & P_{N,M}^{(1)} \end{bmatrix}$$

- Step 2: Compute rating scores $S_i^{(1)}$ $(0 < i \leq N)$ for each model $ML_i^{(1)}$. This is done by rewarding best performing ML models $ML_k^{(1)} = \arg \max_{ML_i^{(1)}(0<i\leq N)} P_{i,j}^{(1)}$ under varying metrics $m_j$ where $j = 1 \ldots M$, with score increments.
- Step 3: Determine the best rated ML model with the highest score $ML^{(1)} = \arg \max_{ML_i^{(1)}(0<i\leq N)} S_i^{(1)}$ in the first algorithm round, and assert the corresponding classifier's performance vector $P_k^{(1)}=[P_{k,1}^{(1)}, \ldots, P_{k,j}^{(1)}, \ldots, P_{k,M}^{(1)}]$.
- Step 4: Repeat Step 1 to Step 3 for the remaining $(N-1)$ rounds which results in $N$ best performing ensemble models from each round $ML^{(1)}$, $ML^{(2)}, \ldots, ML^{(N)}$, with respectively performance vectors $P^{(1)}$, $P^{(2)}, \ldots, P^{(N)}$. By the end of this iterative selection process, the following performance matrix $matrix^{(final)}$ is obtained:

$$\begin{bmatrix} P_1^{(1)} & P_2^{(1)} & \ldots & P_M^{(1)} \\ P_1^{(2)} & P_2^{(2)} & \ldots & P_M^{(2)} \\ \ldots & \ldots & \ldots & \ldots \\ P_1^{(N)} & P_2^{(N)} & \ldots & P_M^{(N)} \end{bmatrix}$$

- Step 5: Repeat Step 2 to Step 3 to determine scores $S^{(i)}$ $(0 < i \leq N)$ for each model $ML^{(i)}$, and determine the best performing model with the highest score $ML = \arg \max_{ML^{(i)}(0<i\leq N)} S^{(i)}$, and assert the corresponding classifier's performance vector $P_k = [P_{k,1}, \ldots, P_{k,j}, \ldots, P_{k,M}]$

The formal algorithm outlined through the previous steps to optimise Ensemble classifiers selection for cybersecurity analysis is depicted in Algorithm 2, where the performance evaluation for each model is depicted in Algorithm 1 explained earlier.

**Algorithm 2** Selective Ensemble

**procedure** SELECTENSEMBLE($\mathcal{ML}, m$)
▷ $\mathcal{ML}$ is a set of individual machine learning models $ML_i$ $(0 < i \leq N)$, and $m$ is a set of related evaluation metrics $m_j$ $(0 < j \leq M)$
  $N = |\mathcal{ML}|, M = |m|$
  **For** Round $r = 1, \ldots, N$ **do**
    **For** $\binom{i=1,\ldots,N}{r}$ **do**
      $ML_i^{(r)} = Ensemble(\mathcal{ML}, r)$
      $S_i^{(r)} = 0$ ▷ Initialize rating scores for each model at each round
    **End For**
    **For** $j = 1, \ldots, M$ **do**
      **For** $\binom{i=1,\ldots,N}{r}$ **do**
        $P_{i,j}^{(r)} = EvaluatePerformance(ML_{i,j}^{(r)}, m_j)$
      **End For**
      **Set** $ML_k^{(r)} = \arg\max_{ML_i^{(r)}} P_{i,j}^{(r)}, (0 < k \leq \binom{N}{r})$
      $S_k^{(r)} += 1$ for $ML_k^{(r)}$ $(0 < k \leq \binom{N}{r})$ ▷ Reward best performing models
    **End For**
    **Set** $ML^{(r)} = \arg\max_{ML_i^{(r)}} S_i^{(r)}$ ▷ Assert the best ensemble model of Round $r$
    $S^{(r)} = 0$
  **End For**
  **For** $j = 1, \ldots, M$ **do**
    **Set** $ML^{(k)} = \arg\max_{ML^{(r)}} P_j^{(r)}, (0 < k \leq N)$
    $S^{(k)} += 1$ for $ML^{(k)}$ $(0 < k \leq N)$
  **End For**
  **Set** $ML = \arg\max_{ML^{(r)}} S^{(r)}$ as the best ensemble model
**End procedure**

It is also critical to identify an appropriate combination scheme for a selected set of individual classifiers and a given dataset (Onan et al., 2016). A hard-voting scheme (or majority voting) is used only when the predicted labels are available. If continuous outputs like posterior probabilities are accessible, then a soft-voting scheme (or average voting) or other linear combinations may be adopted (Kittler et al., 1998). Stacking methods (Wolpert, 1992) are also used to train the output classifier, while taking the feedback from the input classifiers as new features. In bagging (Breiman, 1996) and boosting methods, homogeneous classifiers are usually trained using different samples of the dataset, to produce an ensemble model that is more robust than the individual classifiers. Here we present soft voting ensemble and hard voting ensemble as examples to cluster ML models for a multi-label classification task, as depicted further in Algorithms 3 and 4. These two ensemble techniques only require ML model training for the first round, based on which performance predictions for the other rounds are calculated accordingly. Nonetheless, our ensemble paradigm is extended to involve other techniques like stacking methods that demand model training for multiple rounds. For such applications, we train these models offline and in parallel to better harness computational resources.

**Algorithm 3** Soft Voting Ensemble Scheme

**procedure** SOFTVOTING($\mathcal{ML}, D, \mathcal{K}, \mathcal{L}$)
▷ $\mathcal{ML}$ is a set of individual machine learning models $ML_i$ $(0 < i \leq N)$
▷ K-label dataset has $D$ instances $(D_j, Y_j)$ $(0 < j \leq D)$ with ground truth $Y_j \in Y = \{0, 1\}^k$
  $N = |\mathcal{ML}|, D = |(D_j, Y_j)|$
  **For** $i = 1, \ldots, N$ **do**
    Train($ML_i$)
    $Prob^{(i)} = \emptyset$
  **End For**
  **For** $j = 1, \ldots, D$ **do**
    **For** $i = 1, \ldots, N$ **do**
      $Prob_j^{(i)} = ML_i(D_j)$
    **End For**
    $Prob_j = \frac{1}{N}\sum_{i=1}^{N} Prob_j^{(i)}$
  **End For**
**End procedure**

**Algorithm 4** Hard (Majority) Voting Ensemble Scheme

**procedure** HARDVOTING($\mathcal{ML}, D, \mathcal{K}, \mathcal{L}$)
▷ $\mathcal{ML}$ is a set of individual machine learning models $ML_i$ $(0 < i \leq N)$
▷ K-label dataset has $D$ instances $(D_j, Y_j)$ $(0 < j \leq D)$ with ground truth $Y_j \in Y = \{0, 1\}^k$
  $N = |\mathcal{ML}|, D = |d|$
  **For** $i = 1, \ldots, N$ **do**
    Train($ML_i$)
  **End For**
  **For** $d = 1, \ldots, D$ **do**
    **For** $i = 1, \ldots, N$ **do**
      $Z_j^{(i)} = ML_i(D_j)$
    **End For**
    $Z_j = \arg\max_k [card(L \| Z_j^{(i)})]$
  **End For**
**End procedure**

## 4. Experimental analysis

In this section, we present the implementation details of our experimental methodology and reveal related analysis results. Data sources are used to train the candidate ML models and then validate the performance of the constructed ensemble models. The goal of our experiments is two-fold: (i) illustrate the pivotal role of a localised database that periodically syncs with online and heterogeneous cybersecurity repositories to support security operations; (ii) automatically infer a severity-score of a new vulnerability instance, as well as a threat type that the vulnerability is mostly exposed to.

### 4.1. Experiment design

The experiments are divided into two main steps. The first step is to set up a database that correlates multiple online vulnerability repositories. The resulting dataset contains groups of vulnerability reports that are classified into different exploiting threat types, as well as CVSS categories such as access-vector types, access-complexity levels, etc. The goal is to focus on attention and investment on specific acute risks arising from threat-exploitability with varying degrees of impact-severity. Each class type refers to one label, i.e. one cluster of the whole data set. The second step is to build a pipeline of specific ML techniques applied to different groupings of data classes. Conventional approaches

to quantitative similarity measures involve feature extraction and analysis. These are low-level text features that may consist of words or their compound morphology using the CHARM algorithm (Zaki and Hsiao, 2002). In our experiments, we use both word and CHARM compound features for threat classification. Only word features are used for CVSS-metric categorisation. CVSS version 2 (V2) is considered in the experiments. These two sub-steps are conducted in tandem to train ensembles for threat classification respectively, and to train ensembles for CVSS-categories.

*Cross-linked DB implementation*

We set up a vulnerability database that is kept inherently synchronised with multiple online vulnerability-reports repositories, to feed our proposed ML pipeline. Our database is built on top of cve-search Python API,[9] which brings several online cybersecurity repositories into a local MongoDB,[10] system that can handle extensive unstructured data, as illustrated in Fig. 4. We adopt CVE data to include vulnerability instances that are disclosed but not yet published in NVD. We also adopt NVD to get the CVSS scores added to the reports. We correlated manufacturer websites for standardised component names, vulnerability scoring and patching updates. To do so, we located references mapped to the associated CVE Entries,[11] crawled vulnerability-related data in the cross-linked websites, and stored crawled data into files indexed with CVE-IDs. When the local MongoDB engine starts running, it is kept synchronised on an hourly basis with feeds from public data repositories. Hourly intervals are considered for vulnerability-disclosure schedule when scanning online vulnerability repositories and consolidating collected data into a common localised database.

Subsequently, we built a filter with query keywords to extract vulnerability instances, which are further mapped to the CPE dictionary as well as product names listed in www.cvedetails.com, to generate component configuration patterns. These vulnerability instances are also mapped to CAPEC dictionary and CWE dictionary, to generate component attack patterns and vulnerability patterns separately. We cross-checked vulnerability instances against threat categorisations provided in www.cvedetails.com to retrieve the threat types that a vulnerability instance may be exposed to. This localised database approach supports vulnerability analysis using queries, to retrieve structured information such as the values of component-level vulnerability attributes.

*Data sets*

We retrieved 140 818 vulnerability records with index year values ranging from 2000 to 2019 using our localised database (updated till Nov 30th, 2020), and removed the reports that are marked as *REJECT*, that would otherwise distort the experiment results. With the remaining 132 371 vulnerability reports, we set up a corpus for vulnerability analysis.

During data pre-processing, we generated a stop-word list in the context of cybersecurity reports to remove them from further consideration. We used PorterStemmer[12] from the Natural Language Toolkit (NLTK) for word stemming implementation. We also adopt the bigram functionality in *Gensim.models* to group together common bigram phrases, as well as some Python lemmatisation tools to convert a word to its root form. Following the dataset process, we created a TF-IDF sparse matrix using n-gram or sequence of words features. More specifically, we employ CountVectorizer and TdidfTransforer utilities from Scikit-learn library for vectorisation and TF-IDF value computation. Simultaneously, we sort root words in descending order of TF-IDF values to extract the top-k features.

[9] https://github.com/cve-search/cve-search
[10] https://www.mongodb.com/
[11] https://cve.mitre.org/data/refs/index.html
[12] https://www.nltk.org/howto/stem.html

We checked 132 371 data records for threat labels and found out that we have 39 060 or 29.5% un-labelled data, meaning that those vulnerability reports are not categorised to any existing threat category. Meanwhile, a vulnerability might be exposed to more than one type of threats, meaning that a vulnerability instance can be labelled with multiple threat classes. We used the remaining 93 311 records as ground truth for threat classification training and validation. We also checked the CVSS v2 score labels and found out that 20 reports are not scored under CVSS v2 mechanism. Similarly, we utilised the 132 351 scored reports as ground truth for CVSS v2 metrics classification training and validation.

### 4.2. Experiment methodology

We built a ML pipeline by using the existing package *pipeline* from Scikit-learn library, to automate the ML workflow. We applied different techniques in the ML stack (data processing, feature extraction, etc.), according to the classification tasks and data subsets. In our comparative analysis, we consider five supervised ML models, namely LR, NBSVM, LSTM-ANN, MLP, as well as KNN. Next, we briefly introduce these candidate ML models. While these five models are used individually in cybersecurity and text-mining applications, our ensemble paradigm extends their potential capabilities together with other ML techniques in a range of cybersecurity analysis.

- (a) LR: works as a discriminative supervised-learning classifier that learns to assign a high weight to document features, and then assigns a class $c$ to a document $d$ by directly computing the likelihood $P(c|d)$ (Almukaynizi et al., 2017; Zhang et al., 2011). In more details, we utilise the LR text-mining method training using stochastic gradient descent and the cross-entropy loss, which returns a predicted class for a given document in the test set. We adopt the package *LogisticRegression* from the *Scikit-learn* library to implement this algorithm.
- (b) NBSVM: combines both NB and SVM models. SVM creates optimal hyperplanes, or decision boundaries, to distinctly separate observations into different classes, meaning data points falling on either side of a hyperplane can be attributed to different classes (Joachims, 2001). Methods using SVM calculate the maximum margin between the data points of different classes. Maximising the margin distance provides some reinforcement to improve the model performance, which is usually done by acquiring support vectors where data points are closer to the hyperplanes. NB is based on Bayes theorem and classifies text categorisations of an observation by computing the conditional probability values $P(d|c)$ for each class $c$, given an observation $d$. Our model is based on a NBSVM algorithm proposed by Wang and Manning that uses the NB log-count ratios as feature values (Wang and Manning, 2012).
- (c) LSTM-ANN): LSTM or ANN works by recursively feeding the output of a previous network into the input of the current network, and take the final output after X number of recursions (Zhou et al., 2016). We built a simple two-layer bidirectional LSTM with return-sequences set to True, a dropout-layer with probability = 0.5, and its first layer has a density of 50 classes. Its second layer has a dense of 13 classes for threat categorisation, while a density of 3 classes for CVSS categorisation.
- (d) MLP: is one type of ANN where all the units of the previous layer are connected with the units of the next layer (Zanaty, 2012). Between the input layer and the output layer, the hidden layer would adjust network weights through supervised learning. We adopt Python package *MLPClassifier* from the *Scikit-learn* library to implement this algorithm.
- (e) KNN: works by comparing distances between unknown samples with distances between the closest known k-samples (Trstenjak et al., 2014). We adopt the package *KNeighborsClassifier* from the *Scikit-learn* library to implement this algorithm.

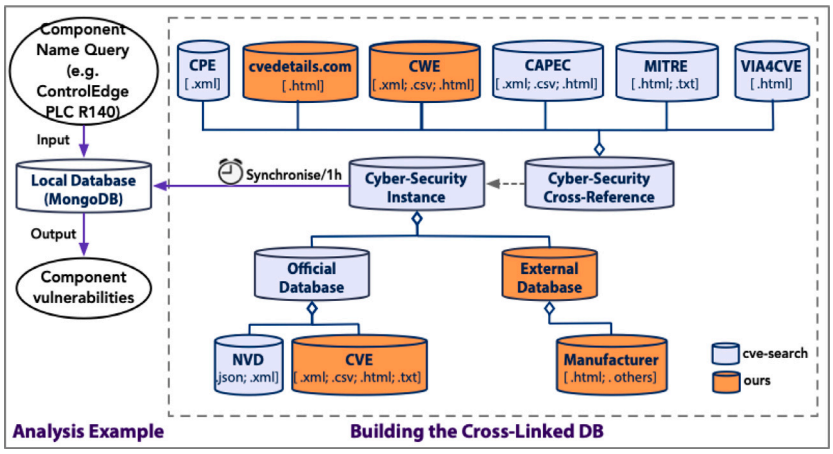


**Fig. 4.** Implementation of cross-linked database.

*Threat categorisation*

Threat categorisation modelling is a multi-label classification problem. We applied a one-to-rest strategy to solve the multi-label problem by decomposing it into multiple independent binary classification problems (one per category). We randomly divided the retrieved 93 311 data records into a 75% (or 69 983 records) training dataset and a 25% (or 23 328 records) validation dataset. Then, we trained the five previously mentioned ML models individually to learn threat categorisation. We applied 3-fold stratified cross-validation using cross-val-score from the *sklearn* package on the training dataset for training and testing threat classification.

We evaluated the learning algorithms' performances on an unseen validation dataset and chose the model with the best performance. Using 5 individual trained models, we generate 5 files with predicted probabilities of different labels for each instance in the first training round. Instead of training multi-round ML models, we utilised the soft-voting technique to combine the predicted probabilities of base individual ML models, in order to compute predictions for ensemble models.

*CVSS categorisation*

Similarly, we modelled the optimised vulnerability scoring system based on the CVSS V2 mechanism, to automatically evaluate the severity and exploit likelihood of a vulnerability instance. We correlated existing CVSS scores of vulnerability instances in NVD to other data sources, such as vendor websites and technical reports from third party reviewers, to adjust scores and better describe the actual severity of vulnerability instances. We employ these resolved scores and corresponding counterparts as a training ground for our ML models. Classification tasks of CVSS V2 characteristics are multi-class tasks. For example, a vulnerability instance is classified into one and only one of three non-overlapping Integrity-impact measurements, i.e. high-, low- or none- integrity impact.

We randomly divided the retrieved 132 351 data records into a 75% (or 99 263 records) training dataset and a 25% (or 33 088 records) validation dataset. Then, we also trained 5 afore-mentioned ML models to learn CVSS v2 categorisation. Here we applied 5-fold stratified cross-validation on the training dataset for training and testing CVSS categorisation. Using trained models, we generated five files with predicted labels of different CVSS v2 characteristics for each model instance. When we applied the ensemble models, we took the majority voting of the predicted labels.

### 4.3. Experiment results

Evaluation of the classification algorithms is a measurement of how far the classification systems' predictions are from the actual class labels, tested on some unseen data.

*Data fusion analysis*

In total, we have 93 311 records labelled under threat categorisation, among which 70.75% vulnerability instances have a single threat-class label. 22.14% vulnerability instances have two threat-class labels. 4.35% vulnerability instances have three threat-class labels. 2.75% vulnerability instances have four threat-class labels, and even fewer vulnerability instances have five threat-class labels. We get the threat types that a reported vulnerability instance is exposed to for the labelled vulnerability reports. *Code Execution*, *Denial of Service*, and *Overflow* are the most typical threat types, with occurrence rates of 30.06%, 25.32%, and 19.46% separately. The least presented threat type is *Http Response Splitting* with an occurrence rate of 0.18%.

Each vulnerability instance has a specific CVSS v2 vector which shows the exploitability and impact of this vulnerability. For each CVSS v2 characteristic, we further compared the contribution of each label to explore the trend of vulnerability exploitability and impact over time. From our statistical analysis of historical data, we observed that most of the reported vulnerability instances are highly exploitable and do not require additional conditions to exploit. The majority of vulnerability instances are accessed through the network and are therefore remotely exploitable, while a minority of vulnerability instances require local access or local account. Vulnerability instances that require adjacent network access, i.e. access to the collision domain of the vulnerable software, has been reported only since the year 2012, and has appeared in a comparably high frequency in the index year of 2014. Generally, the knowledge level and skills to trigger a successful attack have increased, as evidenced by the trend of the even distribution between low complexity and medium complexity over the past ten years. Meanwhile, attackers need to authenticate none or only one time to be able to exploit most of the existing vulnerability instances. The distribution of impact severities indicates a higher diversity of impact compared to the diversity of exploitability in general. The distributions of confidentiality impact, integrity impact and availability impact also show similar diversity patterns, in which partial-impact and none-impact have more appearances.

### 4.4. Threat categorisation

We evaluated the learning algorithms' performance on an unseen validation dataset, and chose six multi-label classification metrics introduced earlier in Section 3.2. We computed measurements for mlACC, mlPRE, mlREC, mlFS, mlEMR by applying corresponding equations, and then we used *hamming_loss* metric from the *sklearn.metrics* package to calculate mlHLS.

To evaluate the performance of our pipeline algorithm and to choose base Ensemble models, we generated a total of 31 performance files in 5 rounds and marked the best models. The individual models

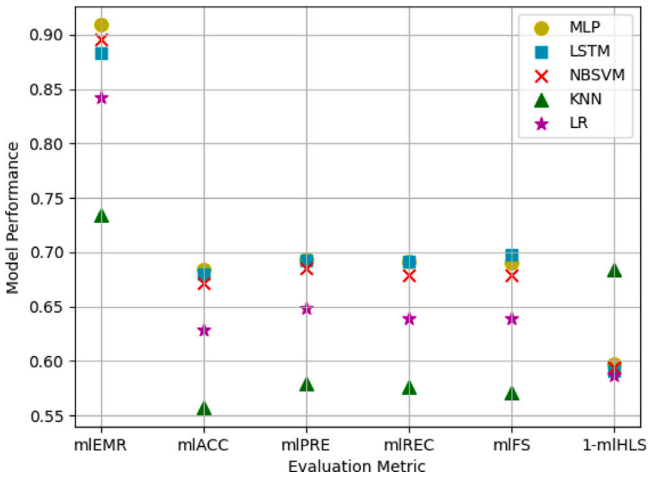


**Fig. 5.** Individual machine-learning model performance for threat classification.

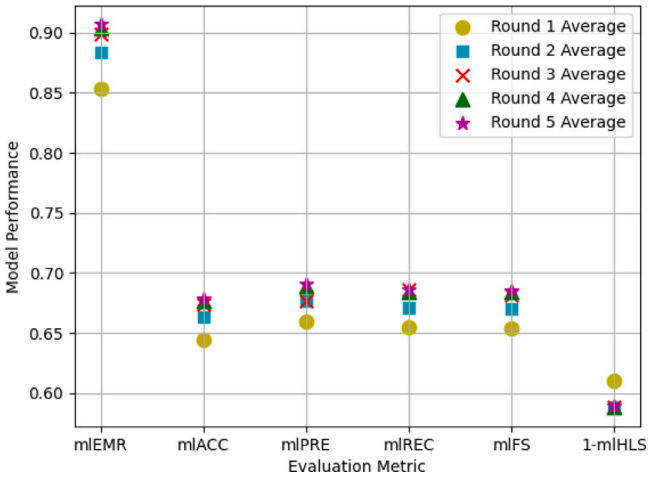


**Fig. 6.** Average performances of each training-round for threat classification.

KNN and LR have comparatively weaker performance than LSTM, NBSVM and MLP, as illustrated further in Fig. 5. Still, the overall ensemble's performance stayed relatively stable when these weaker base learners are counted in for the ensemble. This is the benefit of soft-voting that considers the confidence of each base learner and not just binary choices. We calculate the average performance of models in each round. Except for mlHLS, all the other five metrics have better performance when the amount of participating base learners increase, as illustrated further in Fig. 6.

We further compared best-performing models of each round, as shown in Fig. 7. In all configurations, our pipelined ensemble model gives the best performance. In the first round, individual model MLP reveals the best performance. In the second round, the ensemble of LSTM and MLP exhibits the best performance. In the third round, the ensemble of LSTM, SVM and MLP has the best performance. In the fourth round, the ensemble of LSTM, SVM, KNN and MLP has the best performance. And finally, the fifth round has only one ensemble model, therefore no comparative counterparts. The ensemble of LSTM and MLP provides the best performance overall. Nevertheless, the ensemble of LSTM, NBSVM and MLP provides very close performance. Most of the predictive performances like mlACC, mlPRE, mlFS reach their peak value for the ensemble model with base learners LSTM and MLP. Interestingly, mlHLS has the best performance among individual models, meaning that ensemble models may have more loss generated in the bit string of class labels during prediction. There is no similar

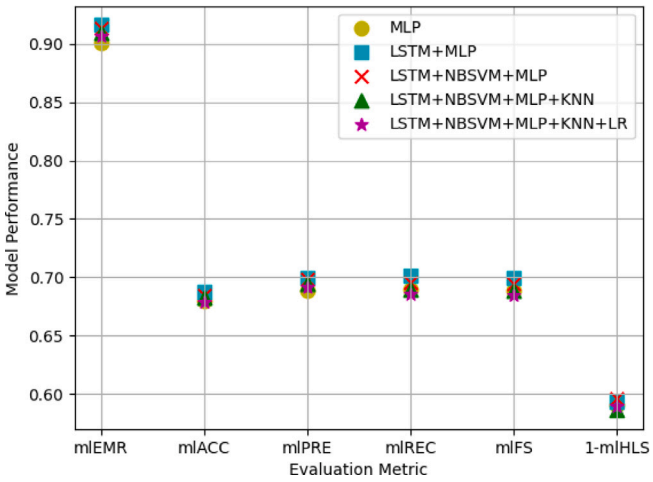


**Fig. 7.** Best performing models of each training-round for threat classification.

threat classification study in the literature results up to our knowledge. Nevertheless, a mlEMR score of 91.63% by the ensemble of LSTM and MLP proves the prediction power of our model.

### 4.5. CVSS characteristic classification and score prediction

To compute the missing CVSS v2 score of a reported vulnerability instance, we need to conduct six separate classification tasks for all the CVSS v2 characteristics. For each classification task, we applied 5-fold stratified cross-validation for selected models. Then, we evaluated the learning algorithms' performance on unseen validation dataset, and chose 6 metrics from *sklearn.metrics* packages, i.e. *balanced_accuracy_score* package for baACC; *confusion_matrix* and *sklearn.utils.multiclass* for maPRE, maREC, and maFS; *cohen_kappa_score* for maCKS; *hamming_loss* for maHLS. These metrics are introduced earlier in Section 3.2, and are chosen considering the imbalanced classes of CVSS metrics.

Unlike the soft-voting scheme adopted in threat classification, we apply hard voting or majority voting for CVSS-characteristic classification and score prediction. Majority voting works when the amount of base learners is equal to or bigger than 3. And hence, we only generated 21 performance files in 4 rounds for each CVSS characteristic classification or score prediction. The main results, i.e. the best performing models of each training round, are illustrated in Fig. 8. More details of the average performances of the trained models of each round are presented in Fig. 9. Next, we provide a narrative assessment of the obtained results in those tables.

- In AccessVector Classification (Figs. 8(a) and 9(a)), the individual model NBSVM has the best prediction performance overall. We can therefore regard NVSVM classifier as a strong classifier, and consider the other four classifiers as weak classifiers. The ensemble of these four weak classifiers performs much better than their individual performances. The average performance increases when the amount of base learners increases from 1 (or Round 1) to 3 (or Round 2), and from 3 (or Round 2) to 5 (or Round 4), but drops when the amount is 4 (or Round 3).
- In AccessComplexity Classification (Figs. 8(b) and 9(b)), the individual model NBSVM and the ensemble of LSTM, NBSVM and MLP deliver very close performance, both of which outperform the other models. For metrics baACC, maFS, maCKS, maHLS, the average performance of the models increases when the amount of base learners increases from 1 to 3, and from 3 to 5, but drops when the amount is 4. Metric maPRE has better average performance when the amount of base learners increases. Metric

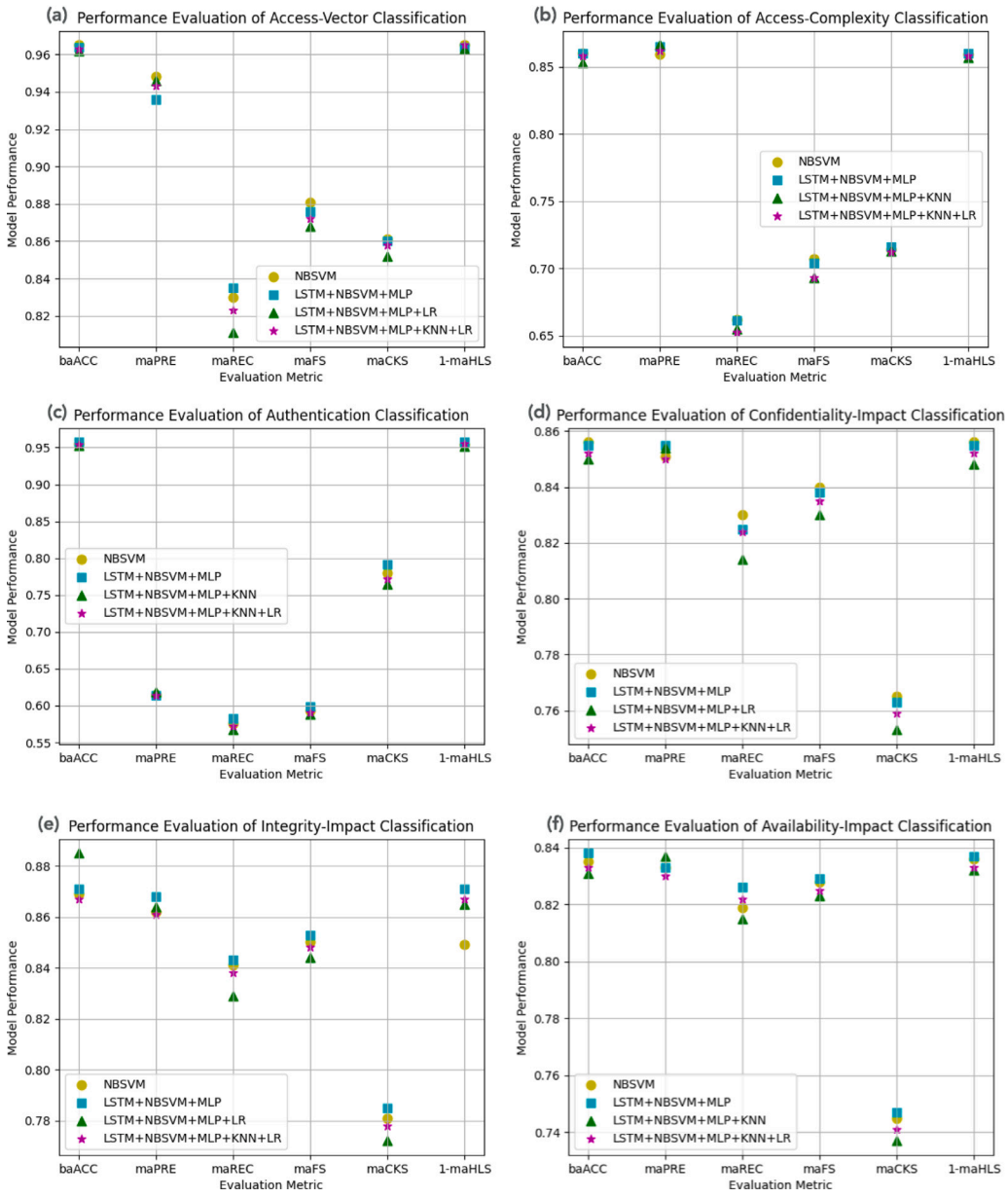


**Fig. 8.** The best performing models of each training-round for cvss-characteristic classifications.

maREC has the best average performance when the amount of base learns is 3.

- In Authentication Classification (Figs. 8(c) and 9(c)), it is clearly seen that the ensemble of base learners LSTM, NBSVM and MLP has the best prediction performance. Most of the metrics have better performances when the amount of participated base learners increases.
- In ConfidentialityImpact Classification (Figs. 8(d) and 9(d)), the individual model NBSVM has the best prediction performance. However, the ensemble of base learners LSTM, NBSVM and MLP also shows a strong performance.
- In IntegrityImpact Classification (Figs. 8(e) and 9(e)), the ensemble of base learners LSTM, NBSVM and MLP has the best prediction performance. Individual models in Round 1 has the weakest average performance, while the ensemble model in Round 4 has the strongest average performance.
- In AvailabilityImpact Classification (Figs. 8(f) and 9(f)), the ensemble of base learners, LSTM, NBSVM and MLP, has the best prediction performance. In general, the ensemble models in Round 2 have a sharp improvement in average performance compared to the individual models in Round 1.

Experimental analysis results for threat and CVSS characteristic categorisations emphasises the need to utilise the multi-round ensemble paradigm, since it is unknown beforehand which round can deliver the best ensemble model, as illustrated in Figs. 7 and 8. For example, in the authentication-categorisation task, Round 2 delivers the best ensemble model. In the confidentiality-impact categorisation task, Round 1 delivers the best ensemble model. In the threat-categorisation task, ensemble models in Round 2 and Round 3 outperformed the ones in the other rounds. The experimental analysis shows that our ensemble model picks appropriate algorithms from 5 standard classifiers, to extend their individual performances in the context of cybersecurity analysis.

We also evaluated the accuracy of CVSS v2 score prediction, which is computed considering the results of the above six classification tasks, using Eq. (2) and using two values for $\delta$ (0.5 and 0.05). And hence, the CVSS score accuracy is a harsh metric similar to Exact-Match-Ratio, which reflects the proportion of complete and correct predictions of all

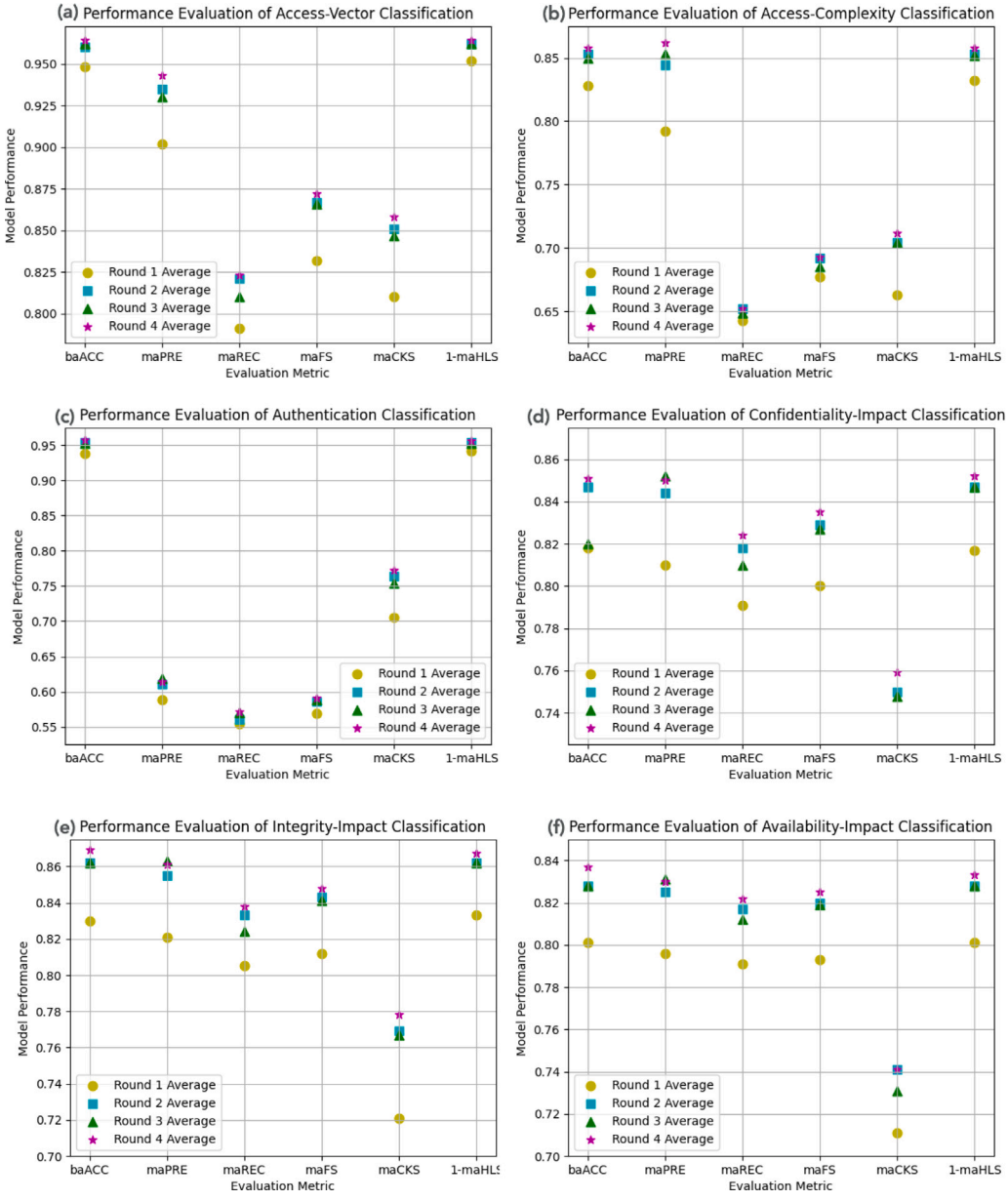


**Fig. 9.** Average performances of the trained models of each training-round for cvss-characteristic classifications.

**Table 3**
Evaluation of CVSS V2 Score Prediction on Untrained Dataset.

| Model | CVSS-ACC ($\delta$ = 0.05) | CVSS-ACC ($\delta$ = 0.5) |
|---|---|---|
| NBSVM | 64.58% | 92.93% |
| **LSTM+NBSVM+MLP** | **64.75%** | **93.05%** |
| LSTM+NBSVM+MLP+LR | 63.27% | 92.81% |
| LSTM+NBSVM+MLP+KNN+LR | 64.06% | 92.90% |

the previous six classifications. Using the CVSS score accuracy metric, the ensemble of LSTM, NBSVM, and MLP has the best performance, as shown in Table 3.

## 5. Conclusion and future works

In this paper, we propose a cognitive cybersecurity analysis framework that streamlines data integration, information processing and knowledge generation to enable cybersecurity intelligence. We discuss the detailed process and logic of (a) heterogeneous data streams categorisation and fusing into a collaborative information system, (b) ingested information transformation into knowledge combined with knowledge formalism, and (c) ML pipeline that automatically finds out the best ensemble model for different cybersecurity classification tasks. More precisely, our ensemble-based approach enables context-aware data analysis that aids situation awareness. In doing so, we empower security operators involved at various SOC levels with a localised and synchronised database that fetches data from several online sources of cybersecurity information. We resolve conflicting vulnerability-severity scores and diverse terminologies used by different parties before adopting the discovered vulnerability instances as training ground truth. We evaluate the proposed ensemble paradigm

through an experimental analysis that involves five commonly used text-mining models. This comparative study shows promising results of our ensemble approach in the cybersecurity scenario of threat categorisation and severity scoring. Furthermore, this exercise provides means to adjust security investments at various organisational levels.

We plan three future directions in our machine-leaning based cybersecurity research (i) adding more cybersecurity data sources such as technical blogs to support analysis, and (ii) further testing ensemble techniques like stacking methods in order to better differentiate vulnerability instances, and (iii) addressing challenging issues like class imbalance problems and computing resource management.

## CRediT authorship contribution statement

**Yuning Jiang:** Conceptualization, Methodology, Simulation, Data collection, Writing - original draft. **Yacine Atif:** Supervision, Writing - reviewing and editing.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgement

This research has been supported in part by EU ISF (Internal Security Fund), Sweden in the context of Project Grant # A431.678/2016.

## References

Allodi, Luca, Massacci, Fabio, 2014. Comparing vulnerability severity and exploits using case-control studies. ACM Trans. Inf. Syst. Secur. 17 (1), 1–20.

Almukaynizi, Mohammed, Nunes, Eric, Dharaiya, Krishna, Senguttuvan, Manoj, Shakarian, Jana, Shakarian, Paulo, 2017. Proactive identification of exploits in the wild through vulnerability mentions online. In: 2017 International Conference on Cyber Conflict. CyCon US. IEEE, pp. 82–88.

Aly, Mohamed, 2005. Survey on multiclass classification methods. Neural Netw. 19, 1–9.

Andrade, Roberto Omar, Fuertes, Walter, Cadena, Susana, Cadena, Alyssa, Tello - Oquendo, Luis, Córdova, Daniela, Garcés, Iván Ortiz, Cazares, María Fernanda, 2019. Information security management in university campus using cognitive security. Int. J. Comput. Sci. Inf. Secur. 13 (4), 124.

Andrade, Roberto O., Yoo, Sang Guun, 2019. Cognitive security: A comprehensive study of cognitive science in cybersecurity. J. Inform. Secur. Appl. 48, 102352.

Angelini, Marco, Blasilli, Graziano, Catarci, Tiziana, Lenti, Simone, Santucci, Giuseppe, 2018. Vulnus: Visual vulnerability analysis for network security. IEEE Trans. Vis. Comput. Graphics 25 (1), 183–192.

Anwar, Afsah, Abusnaina, Ahmed, Chen, Songqing, Li, Frank, Mohaisen, David, 2020. Cleaning the NVD: Comprehensive quality assessment, improvements, and analyses. arXiv preprint arXiv:2006.15074.

Bhatt, Sandeep, Manadhata, Pratyusa K., Zomlot, Loai, 2014. The operational role of security information and event management systems. IEEE Secur. Priv. 12 (5), 35–41.

Bozorgi, Mehran, Saul, Lawrence K, Savage, Stefan, Voelker, Geoffrey M, 2010. Beyond heuristics: Learning to classify vulnerabilities and predict exploits. In: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. pp. 105–114.

Breiman, Leo, 1996. Bagging predictors. Mach. Learn. 24 (2), 123–140.

Bullough, Benjamin L, Yanchenko, Anna K, Smith, Christopher L, Zipkin, Joseph R, 2017. Predicting exploitation of disclosed software vulnerabilities using open-source data. In: Proceedings of the 3rd ACM on International Workshop on Security and Privacy Analytics. pp. 45–53.

Christey, Steve, Martin, Brian, 2013. Buying into the Bias: Why Vulnerability Statistics Suck. Tech. Rep, 1, BlackHat, Las Vegas, USA.

Debole, Franca, Sebastiani, Fabrizio, 2004. Supervised term weighting for automated text categorization. In: Text Mining and Its Applications. Springer, pp. 81–97.

Dietterich, Thomas G., 2000. Ensemble methods in machine learning. In: International Workshop on Multiple Classifier Systems. Springer, pp. 1–15.

Dong, Ying, Guo, Wenbo, Chen, Yueqi, Xing, Xinyu, Zhang, Yuqing, Wang, Gang, 2019. Towards the detection of inconsistencies in public security vulnerability reports. In: 28th {$USENIX$} Security Symposium. {$USENIX$} Security 19. pp. 869–885.

Edkrantz, Michel, Said, Alan, 2015. Predicting cyber vulnerability exploits with machine learning. In: SCAI, pp. 48–57.

Fang, Yong, Liu, Yongcheng, Huang, Cheng, Liu, Liang, 2020. FastEmbed: Predicting vulnerability exploitation possibility based on ensemble machine learning algorithm. PLoS One 15 (2), e0228439.

Feng, Charles, Wu, Shuning, Liu, Ningwei, 2017. A user-centric machine learning framework for cyber security operations center. In: 2017 IEEE International Conference on Intelligence and Security Informatics. ISI. IEEE, pp. 173–175.

Freund, Yoav, Schapire, Robert E., 1995. A desicion-theoretic generalization of on-line learning and an application to boosting. In: European Conference on Computational Learning Theory. Springer, pp. 23–37.

Geer, Dan, Roytman, Michael, 2013. Measuring vs. modeling. login 38 (6), 64–67.

Heelan, Sean, 2011. Vulnerability detection systems: Think cyborg, not robot. IEEE Secur. Priv. 9 (3), 74–77.

Holm, Hannes, Korman, Matus, Ekstedt, Mathias, 2015. A bayesian network model for likelihood estimations of acquirement of critical software vulnerabilities and exploits. Inf. Softw. Technol. 58, 304–318.

Holzinger, Andreas, Plass, Markus, Kickmeier-Rust, Michael, Holzinger, Katharina, Crişan, Gloria Cerasela, Pintea, Camelia-M, Palade, Vasile, 2019. Interactive machine learning: Experimental evidence for the human in the algorithmic loop. Appl. Intell. 49 (7), 2401–2414.

Householder, Allen D., Wassermann, Garret, Manion, Art, King, Chris, 2017. The Cert Guide to Coordinated Vulnerability Disclosure. Carnegie-Mellon Univ Pittsburgh, Pa, Pittsburgh, United States.

Husari, Ghaith, Al -Shaer, Ehab, Ahmed, Mohiuddin, Chu, Bill, Niu, Xi, 2017. Ttpdrill: Automatic and accurate extraction of threat actions from unstructured text of cti sources. In: Proceedings of the 33rd Annual Computer Security Applications Conference. ACM, pp. 103–115.

Jiang, Yuning, Atif, Yacine, Ding, Jianguo, 2019. Cyber-physical systems security based on a cross-linked and correlated vulnerability database. In: International Conference on Critical Information Infrastructures Security. Springer, pp. 71–82.

Jiang, Jie, Yu, Lele, Jiang, Jiawei, Liu, Yuhong, Cui, Bin, 2018. Angel: A new large-scale machine learning system. Natl. Sci. Rev. 5 (2), 216–236.

Jo, Hyeonseong, Kim, Jinwoo, Porras, Phillip, Yegneswaran, Vinod, Shin, Seungwon, 2020. GapFinder: Finding inconsistency of security information from unstructured text. IEEE Trans. Inf. Forensics Secur. 16, 86–99.

Joachims, Thorsten, 2001. A statistical learning learning model of text classification for support vector machines. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 128–136.

Joh, HyunChul, Malaiya, Yashwant K., 2011. Defining and assessing quantitative security risk measures using vulnerability lifecycle and cvss metrics. In: The 2011 International Conference on Security and Management. Sam. pp. 10–16.

Johnson, Pontus, Lagerström, Robert, Ekstedt, Mathias, Franke, Ulrik, 2016. Can the common vulnerability scoring system be trusted? A bayesian analysis. IEEE Trans. Dependable Secure Comput. 15 (6), 1002–1015.

Khazaei, Atefeh, Ghasemzadeh, Mohammad, Derhami, Vali, 2016. An automatic method for CVSS score prediction using vulnerabilities description. J. Intell. Fuzzy Systems 30 (1), 89–96.

Kittler, Josef, Hatef, Mohamad, Duin, Robert P.W., Matas, Jiri, 1998. On combining classifiers. IEEE Trans. Pattern Anal. Mach. Intell. 20 (3), 226–239.

Kowsari, Kamran, Jafari Meimandi, Kiana, Heidarysafa, Mojtaba, Mendu, Sanjana, Barnes, Laura, Brown, Donald, 2019. Text classification algorithms: A survey. Information 10 (4), 150.

Van der Laan, Mark J., Polley, Eric C., Hubbard, Alan E., 2007. Super learner. Stat. Appl. Genet. Mol. Biol. 6 (1).

Ladd, B., 2017. The race between security professionals and adversaries. Recorded Future Blog. Available online in November.

Larcher, Jr., Celio, H.N., Barbosa, Helio J.C., 2019. Auto-CVE: A coevolutionary approach to evolve ensembles in automated machine learning. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 392–400.

Li, Longjie, Yu, Yang, Bai, Shenshen, Hou, Ying, Chen, Xiaoyun, 2017. An effective two-step intrusion detection approach based on binary classification and $k$-NN. IEEE Access 6, 12060–12073.

Liao, Q. Vera, Gruen, Daniel, Miller, Sarah, 2020. Questioning the AI: Informing design practices for explainable AI user experiences. In: Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems. CHI '20, Association for Computing Machinery, New York, NY, USA, pp. 1–15. http://dx.doi.org/10.1145/3313831.3376590.

Liao, Xiaojing, Yuan, Kan, Wang, XiaoFeng, Li, Zhou, Xing, Luyi, Beyah, Raheem, 2016. Acing the ioc game: Toward automatic discovery and analysis of open-source cyber threat intelligence. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. CCS '16, ACM, New York, NY, USA, pp. 755–766. http://dx.doi.org/10.1145/2976749.2978315.

Lower, Nicholas, Zhan, Felix, 2020. A study of ensemble methods for cyber security. In: 2020 10th Annual Computing and Communication Workshop and Conference. CCWC. IEEE, pp. 1001–1009.

Mavroeidis, Vasileios, Bromander, Siri, 2017. Cyber threat intelligence model: An evaluation of taxonomies, sharing standards, and ontologies within cyber threat intelligence. In: Intelligence and Security Informatics Conference. EISIC, 2017 European. IEEE, pp. 91–98.

Merrouni, Zakariae Alami, Frikh, Bouchra, Ouhbi, Brahim, 2019. Automatic keyphrase extraction: A survey and trends. J. Intell. Inf. Syst. 1–34.
Na, Sarang, Kim, Taeeun, Kim, Hwankuk, 2016. A study on the classification of common vulnerabilities and exposures using naive Bayes. In: International Conference on Broadband and Wireless Computing, Communication and Applications. Springer, pp. 657–662.
Neuhaus, Stephan, Zimmermann, Thomas, 2010. Security trend analysis with CVE topic models. In: Software Reliability Engineering. ISSRE, 2010 IEEE 21st International Symposium on. IEEE, pp. 111–120.
Onan, Aytuğ, Korukoğlu, Serdar, Bulut, Hasan, 2016. A multiobjective weighted voting ensemble classifier based on differential evolution algorithm for text sentiment classification. Expert Syst. Appl. 62, 1–16.
Oprea, Alina, Li, Zhou, Norris, Robin, Bowers, Kevin, 2018. Made: Security analytics for enterprise threat detection. In: Proceedings of the 34th Annual Computer Security Applications Conference. pp. 124–136.
Osifeko, Martins O., Hancke, Gerhard P., Abu-Mahfouz, Adnan M., 2020. Artificial intelligence techniques for cognitive sensing in future IoT: State-of-the-art, potentials, and challenges. J. Sens. Actuator Netw. 9 (2), 21.
Patil, Sangameshwar, 2017. Concept-based classification of software defect reports. In: Proceedings of the 14th International Conference on Mining Software Repositories. IEEE Press, pp. 182–186.
Quintal, Kyle, Kantarci, Burak, Erol -Kantarci, Melike, Malton, Andrew, Walenstein, Andrew, 2020. Enterprise security with adaptive ensemble learning on cooperation and interaction patterns. In: 2020 IEEE 17th Annual Consumer Communications & Networking Conference. CCNC. IEEE, pp. 1–7.
Rajagopal, Smitha, Kundapur, Poornima Panduranga, Hareesha, Katiganere Siddaramappa, 2020. A stacking ensemble for network intrusion detection using heterogeneous datasets. Secur. Commun. Netw. 2020.
Resende, Paulo Angelo Alves, Drummond, André Costa, 2018. A survey of random forest based methods for intrusion detection systems. ACM Comput. Surv. 51 (3), 1–36.
Ruohonen, Jukka, 2019. A look at the time delays in CVSS vulnerability scoring. Appl. Comput. Inform. 15 (2), 129–135.
Russo, Ernesto Rosario, Di Sorbo, Andrea, Visaggio, Corrado A, Canfora, Gerardo, 2019. Summarizing vulnerabilities' descriptions to support experts during vulnerability assessment activities. J. Syst. Softw. 156, 84–99.
Sauerwein, Clemens, Pekaric, Irdin, Felderer, Michael, Breu, Ruth, 2019. An analysis and classification of public information security data sources used in research and practice. Comput. Secur. 82, 140–155.
Scandariato, Riccardo, Walden, James, Hovsepyan, Aram, Joosen, Wouter, 2014. Predicting vulnerable software components via text mining. IEEE Trans. Softw. Eng. 40 (10), 993–1006.
Scarfone, Karen, Mell, Peter, 2009. An analysis of cvss version 2 vulnerability scoring. In: 2009 3rd International Symposium on Empirical Software Engineering and Measurement. IEEE, pp. 516–525.
Shahzad, Muhammad, Shafiq, Muhammad Zubair, Liu, Alex X., 2012. A large scale exploratory analysis of software vulnerability life cycles. In: 2012 34th International Conference on Software Engineering, ICSE. IEEE, pp. 771–781.
Siboni, Shachar, Sachidananda, Vinay, Meidan, Yair, Bohadana, Michael, Mathov, Yael, Bhairav, Suhas, Shabtai, Asaf, Elovici, Yuval, 2019. Security testbed for internet-of-things devices. IEEE Trans. Reliab. 68 (1), 23–44.
Sokolova, Marina, Lapalme, Guy, 2009. A systematic analysis of performance measures for classification tasks. Inf. Process. Manage. 45 (4), 427–437.
Sommer, Robin, Paxson, Vern, 2010. Outside the closed world: On using machine learning for network intrusion detection. In: 2010 IEEE Symposium on Security and Privacy. IEEE, pp. 305–316.
Sorower, Mohammad S., 2010. A Literature Survey on Algorithms for Multi-Label Learning. 18, Oregon State University, Corvallis, pp. 1–25.
Spanos, Georgios, Angelis, Lefteris, Toloudis, Dimitrios, 2017. Assessment of vulnerability severity using text mining. In: Proceedings of the 21st Pan-Hellenic Conference on Informatics. pp. 1–6.
Stine, Ian, Rice, Mason, Dunlap, Stephen, Pecarina, John, 2017. A cyber risk scoring system for medical devices. Int. J. Crit. Infrastruct. Prot. 19, 32–46.
Tong, Haonan, Liu, Bin, Wang, Shihai, 2018. Software defect prediction using stacked denoising autoencoders and two-stage ensemble learning. Inf. Softw. Technol. 96, 94–111.
Torres, Javier Martínez, Comesaña, Carla Iglesias, García-Nieto, Paulino J, 2019. Machine learning techniques applied to cybersecurity. Int. J. Mach. Learn. Cybern. 10 (10), 2823–2836.
Trstenjak, Bruno, Mikac, Sasa, Donko, Dzenana, 2014. KNN with TF-IDF based framework for text categorization. Procedia Eng. 69, 1356–1364.
Tsoumakas, Grigorios, Katakis, Ioannis, 2007. Multi-label classification: An overview. Int. J. Data Wareh. Min. 3 (3), 1–13.
Vanerio, Juan, Casas, Pedro, 2017. Ensemble-learning approaches for network security and anomaly detection. In: Proceedings of the Workshop on Big Data Analytics and Machine Learning for Data Communication Networks. pp. 1–6.
Veksler, Vladislav D, Buchler, Norbou, LaFleur, Claire G, Yu, Michael S, Lebiere, Christian, Gonzalez, Cleotilde, 2020. Cognitive models in cybersecurity: Learning from expert analysts and predicting attacker behavior. Front. Psychol. 11.
Wang, Sida, Manning, Christopher D., 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In: Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers, vol. 2, Association for Computational Linguistics, pp. 90–94.
Wolpert, David H., 1992. Stacked generalization. Neural Netw. 5 (2), 241–259.
Xin, Yang, Kong, Lingshuang, Liu, Zhi, Chen, Yuling, Li, Yanmiao, Zhu, Hongliang, Gao, Mingcheng, Hou, Haixia, Wang, Chunhua, 2018. Machine learning and deep learning methods for cybersecurity. IEEE Access 6, 35365–35381.
Zaki, Mohammed J., Hsiao, Ching-Jui, 2002. Charm: An efficient algorithm for closed itemset mining. In: Proceedings of the 2002 SIAM International Conference on Data Mining. SIAM, pp. 457–473.
Zanaty, E.A., 2012. Support vector machines (SVMs) versus multilayer perception (MLP) in data classification. Egypt. Inform. J. 13 (3), 177–183.
Zhang, Su, Caragea, Doina, Ou, Xinming, 2011. An empirical study on using the national vulnerability database to predict software vulnerabilities. In: International Conference on Database and Expert Systems Applications. Springer, pp. 217–231.
Zhou, Peng, Qi, Zhenyu, Zheng, Suncong, Xu, Jiaming, Bao, Hongyun, Xu, Bo, 2016. Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. arXiv preprint arXiv:1611.06639.
Zhu, Ziyun, Dumitraş, Tudor, 2016. Featuresmith: Automatically engineering features for malware detection by mining the security literature. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. ACM, pp. 767–778.

**Yuning Jiang** is a Ph.D. student in Informatics at University of Skovde in Sweden. Her doctoral research investigates cybersecurity issues through the overabundance of big data that can be used in several algorithms to improve the current state of cybersecurity. She particularly specialises in cyber physical system vulnerability analysis, where she was acknowledged as the Best Young Research Award at the 14th International Conference on Critical Information Infrastructures Security.

**Yacine Atif** is a Professor at University of Skovde in Sweden since January 2016. He received a Ph.D. degree in Computer Science from Hong Kong University of Science and Technology (HKUST) in 1996. Over the last decade, he led several initiatives that are designed to create rich learning environments to promote the transformation of education through the innovative use of learning technologies. Lately, his scope expanded to include the convergence of cyber physical systems and Internet of Things in order to address some of the corresponding innovation opportunities, and underlying cybersecurity challenges.