Identification of Malicious Slovak Domains using Machine Learning

Sophia Petra Krišáková[1,†], Rastislav Krivoš-Belluš[1,*,†], Eva Marková[1,†] and Vanda Matušíková[1,†]

[1]*Institute of Computer Science, Faculty of Science, Pavol Jozef Šafárik University in Košice, Jesenná 5, 040 01 Košice, Slovakia*

Abstract

The shift of services to cyberspace has created new opportunities for attackers who generate malicious domains that mimic popular websites in order to deceive users and obtain their sensitive information, such as login credentials, personal data, or payment information. Such domains pose a significant risk to cybersecurity and user privacy. This paper aims to design and implement a tool for the fast and efficient identification of new Slovak domains that may represent security threats. For this purpose, we created a custom dataset comprising approximately 900 benign and 900 malicious domains. The benign domains were selected manually based on their established reputation, while the malicious domains were collected from publicly available lists. After extracting 29 relevant domain attributes, we applied three machine learning models: logistic regression, Random Forest, and XGBoost. The highest detection accuracy was achieved by Random Forest (97.5%) and XGBoost (97.4%), confirming the effectiveness of the proposed approach.

Keywords

blacklist, detection, domains, machine learning, reputation analysis

1. Introduction

Nowadays, there are a large number of suspicious websites on the Internet, which are perceived as harmless at first glance by the average user. They look like harmless existing sites, imitating them in name but also in appearance. Often the aim of these sites is to deceive the user, to infect him, or to obtain personal data.

There are now various domain verification tools and approaches that use a combination of blacklists and services to see if a domain is suspicious. However, there is a problem here if the domain is new and these services and blacklists have not yet picked up information about it. In that case, it would be advisable to use a tool that would act proactively and try to analyze the characteristics of the site and classify it as benign or malignant accordingly.

To summarize the problems outlined above, we emphasize the following questions that we aim to answer:

- Which methods are shown to be suitable for identifying malicious domains?
- Which attributes influence the identification of malicious domains?

To answer these questions, we have focused on the analysis of '.sk' top-level domains that uses machine learning models trained on our own dataset, which consists only of Slovak domains.

The aim of the article is to create a dataset with Slovak domains and use existing machine learning methods. We focus on all domains with the aim that if a new domain is added, our model will be able to immediately classify it as malignant or benign.

In section 2, we explored existing approaches to proactive analysis and classification of domains, focusing on machine learning models and their effectiveness. In section 3, we described methods for

ITAT'25: Information Technologies – Applications and Theory, September 26–30, 2025, Telgárt, Slovakia

*Corresponding author.

†These authors contributed equally.

sophia.petra.krisakova@upjs.sk (S. P. Krišáková); rastislav.krivos-bellus@upjs.sk (R. Krivoš-Belluš); eva.markova@upjs.sk (E. Marková); vanda.matusikova@student.upjs.sk (V. Matušíková)

0009-0007-0650-5180 (R. Krivoš-Belluš); 0000-0002-3120-1695 (E. Marková)

collecting domain data from open sources, which we utilized in our tool and we detailed the practical part, including attribute selection, data preparation for model training, data cleaning and preprocessing, and the training of machine learning models. In the final, section 4, we presented our solution for automating the analysis of network entities using machine learning and explained the operational principles of our tool. Additionally, we evaluated the effectiveness of the applied machine learning models and discussed their performance.

2. Related works

In the field of classifying domains as malicious or benign, various methods have been developed to tackle this issue. Malicious domains may include those hosting phishing sites, fraudulent shops, or other platforms used for harmful activities. Most research in this area follows a similar approach, primarily focusing on expanding the range of features monitored to identify a domain's maliciousness. The key distinction between these methods typically lies in the choice of machine learning techniques, which are selected based on the specific requirements of each study.

2.1. Blacklist

Domain blacklists were among the earliest methods to counter malicious domains, particularly phishing ones, by warning users against visiting dangerous sites. These lists store known malicious domains to inform users to avoid them [1, 2]. However, this approach has limitations: it may fail to evaluate new or previously unassessed domains, or it may misjudge a domain's reputation [2, 3]. Additionally, blacklists require frequent updates to remain effective, adding newly identified malicious domains while removing those no longer posing a threat [2].

Fukushima et al. [4] introduced a proactive blacklist enhanced by monitoring attributes commonly exploited by attackers, such as the autonomous system number, IP address block, IP address, domain name, and domain registrar. By analyzing these patterns, they determined whether a domain was malicious.

2.2. Machine learning

Machine learning, a subset of artificial intelligence, involves algorithms that enable computers to predict outcomes by identifying patterns in input data [5] . The process begins with learning from a training set, followed by generalization to address real-world problems. Machine learning can be applied in several ways: classification, where a model categorizes inputs into classes (e.g., identifying spam emails); regression, which predicts continuous values (e.g., estimating damage from a security incident); clustering; and reinforcement learning, which involves learning through feedback [6].

Machine learning is divided into supervised and unsupervised learning. Supervised learning, similar to learning with a teacher, uses labeled training data, assigning each example to a class in classification or an output value in regression. Unsupervised learning, or learning without a teacher, relies on unlabeled data, requiring the model to independently identify similarities among examples [6].

2.2.1. Logistic regression

Logistic regression, a generalized linear model derived from linear regression, is designed to predict categorical data, particularly binary outcomes, rather than continuous values [7]. Palaniappan et al. [8] applied logistic regression to a dataset of 20,000 records to classify domains as benign or malicious. They analyzed DNS-based attributes, such as ASN, IP address, PTR record, domain owner, and registrar, alongside web-related attributes like user time spent on a website, number of websites, and geolocation. Lexical attributes included counts of periods, underscores, commas, numerals, and a list of suspicious words. Their model achieved a classification accuracy of 60%.

2.2.2. Decision tree

Decision tree, a type of supervised learning algorithm, operates using decision nodes that evaluate specific data attributes. Based on these evaluations, the algorithm navigates through subsequent nodes until reaching leaf nodes, which represent the predicted value or class for the input data [9].

2.2.3. J48

J48, Java implementation of the C4.5 algorithm [10] in data mining tool WEKA, generates a classification decision tree by recursively splitting a dataset into smaller subsets using a depth-first approach. It selects attributes to serve as decision nodes based on their ability to effectively divide the data [11].

Bilge et al. [12] utilized passive DNS analysis to identify domains involved in malicious activities. They monitored characteristics of malicious domains over several months, extracting temporal properties, DNS response-oriented properties, TTL properties, and lexical domain name properties. Their dataset comprised 100 billion DNS queries, and they tested their approach at ISPs using 15 features with a J48 decision tree as the classifier.

Messabi et al. [13] enhanced domain classification by introducing new features after reviewing existing approaches. They identified the average number of characters in legitimate domains as approximately 12–13, noting that malicious domains often had more dots, hyphens, and digits. They also analyzed the frequency of individual characters and highlighted frequently abused top-level domains (TLDs) such as .zip, .us, .top, .men, and .ru. Additionally, they examined specific words, termed ”tokens,” including ”direct,” ”redirect,” ”transfer,” and obscene words prevalent in malicious domains. After evaluating multiple classifiers, they selected the J48 decision tree for classification.

2.2.4. XGBoost

XGBoost, developed by Chen and Guestrin in 2016, is a gradient boosting algorithm applied to tree-based models, known as a Gradient Boosting Machine. It excels in classification and regression tasks, efficiently handling large datasets and missing data [14].

Horak [15] integrated external data sources, including WHOIS, RDAP, TLS, geolocation, and IP address reputation, using XGBoost and SHAP [16] to classify domains and assess feature importance. To evaluate the classifier on an unbalanced dataset with many benign domains, he employed k-fold cross-validation, aiming to maximize the F1 score. K-fold cross-validation and the F1 score are discussed in detail later in 3.7 and 4.3.

2.2.5. Others

Hamadouche et al. [17] employed four supervised learning algorithms for domain classification: Support Vector Machine (SVM), Random Forest, Decision Tree, and XGBoost. SVM, suitable for classification and regression, identifies a hyperplane to separate data into classes using support vectors as decision functions [18, 19]. Random Forest, a classifier, aggregates multiple decision trees, determining the class through majority voting [19]. They analyzed 39 attributes, including lexical, content, and network features. Notably, content attributes included indicators like keyloggers, location tracking, microphone and camera access, blank pages, and pop-ups. Their comparison revealed that XGBoost achieved the highest performance and accuracy.

Ma et al. [20] developed an application to classify domains by analyzing lexical and network attributes. Lexical attributes included the hostname, main domain, tokens in URL components (e.g., ”.com”), the segment after the last slash, and the Top-Level Domain (TLD). Network attributes encompassed WHOIS data (registration date, registrar, domain holder), location details (IP address prefix, AS number, registrar location), connection speed, blacklist presence, TTL, and similarity to spam-related domains. Article’s authors utilized online learning algorithms, which update with each new example, unlike batch algorithms that learn periodically from data batches. The Confidence-Weighted (CW) algorithm, a recently developed method, yielded the best results.

3. Methodology

In addition to exploring existing approaches, our research includes designing our own tool for classifying malicious or suspicious domains and evaluating it. In our approach, we have used machine learning because it seems to be a suitable, powerful and efficient approach to solve the problem of this paper. In the following subsections, we discuss the process of attribute selection, data collection, processing and editing, and describe the selected machine learning models, their hyperparameter settings and training.

3.1. Dataset

For the selection of attributes, we were inspired by existing approaches, which we described in Chapter 2. We mainly used existing attributes collected by Marques et al [21]. But after a closer analysis, some seemed redundant or we found that it would be useful to add others. For example, TLD tracking was found to be redundant for our work, as we only deal with Slovak domains in our work. Another was DomainInAlexaDB, as this list of the most visited domains is no longer updated, and moreover it is a list that observes TLD domains of all types, so domains in our dataset would rarely appear there.
Some WHOIS-related attributes were also problematic. The attribute in the aforementioned work RegisteredCountryCode was also difficult to obtain at times, most of the time this data was obscured due to GDPR, so we decided to omit it. Similarly, the attribute we wanted to add in our own interest, the domain owner data, was also hidden in most cases, and since it would be problematic to insert an attribute into the dataset whose values would be mostly empty, we decided to omit it. We also discarded the number of subdomains because, after manually analyzing the domains, it seemed that this attribute would not have a specific impact on the result, as similar counts occurred for both benign and malicious domains.

3.1.1. WHOIS

WHOIS data includes contact information for the domain owner, administrator, and technical contacts, such as names, addresses, or organization names. Initially, we aimed to analyze the following attributes: registrar name, domain name or owner name, country code, creation date, and last update date, labeled in the dataset as Registrar, Registrant, CountryCode, CreationDate, and LastUpdate, respectively. Over time, we excluded the Registrant attribute due to its frequent concealment in malicious domains. Similarly, the CountryCode attribute was largely empty and thus also dropped.

The retained WHOIS data attributes were Registrar, CreationDate, and LastUpdate, with Registrar indicating the domain lessor. For benign domains, common lessors included Websupport s.r.o., Webglobe, a.s., WebHouse, s.r.o., WEBY GROUP, s.r.o., and INTERNET CZ, a.s. Notably, WebHouse, s.r.o. and Webglobe, a.s. are accredited registrars under SK-NIC, a.s., the registry administrator for the '.sk' top-level domain and 'org.sk' second-level domain, with accreditation ensuring service quality verification [22]. Among malicious domains, Gransy s.r.o. was the most prevalent registrar, favored by attackers for its low-cost hosting for certain TLDs and WHOIS data obfuscation, which complies with GDPR but is often indicative of malicious domains. Using ViewDNS.info's Reverse IP Lookup tool [70], we determined that IP addresses linked to Gransy Ltd. were almost always associated with phishing or other malicious sites.

For the CreationDate and LastUpdate attributes, we focused on the domain's age and its most recent update. We modified these attributes based on the original dataset as follows: domains younger than one month or last updated less than one month ago were assigned a value of 1; those less than six months old or updated less than six months ago were given a value of 2; those less than one year old or updated less than one year ago were tagged with a value of 3; and those older than one year or updated more than one year ago were assigned a value of 4. If no data was available for either attribute, we assigned a value of 0.

In Figure 1 we see the 5 most used registrars associated with benign domains. As we mentioned earlier, the most used among them is Websupport s.r.o. In Figure 2, on the other hand, the 5 most used

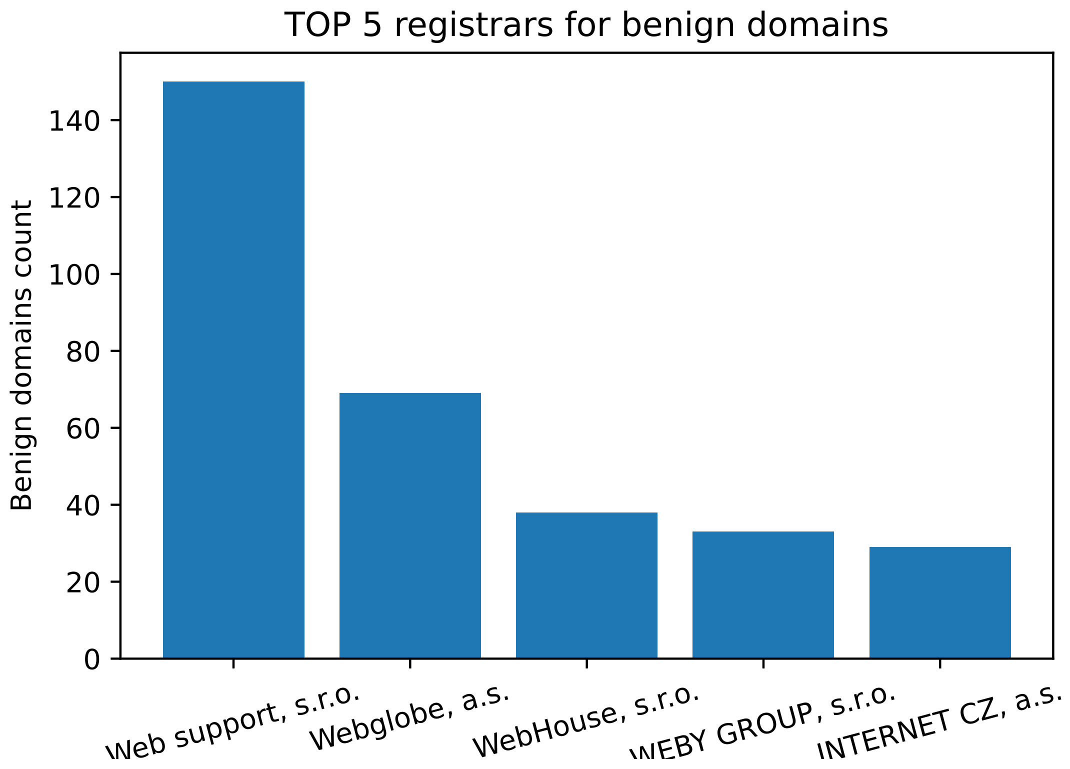

Figure 1: Benign domains

registrars associated with malicious domains are shown. Gransy s.r.o. used more than 400 malicious domains contained in our dataset.

3.1.2. DNS records

Domain Name System (DNS) and its records provide insight into domain classification. We retained attributes related to MX, SPF, DKIM, and DMARC records from the dataset, tracking their presence as binary attributes. A domain with an MX record was assigned a value of 1, while those without received a 0. Similarly, we tracked SPF, DKIM, and DMARC records, which indicate domain legitimacy and security. Although these records are often automatically added by service providers, most malicious domains in our dataset lacked them.

3.1.3. Header information

From query headers for a given domain, we can extract details such as the server type processing the request and generating the response, the page's Content Type, and the charset (character set) used. Additionally, the query response code—such as 200 OK or codes starting with 3 (redirects), 4, or 5 (errors)—provides further insight. Our analysis focused primarily on the page response to identify redirects, which could suggest fraudulent activity, and the server type, as outdated servers may indicate vulnerabilities. Initially, we examined Content-Type and charset, but these were consistent across domains and thus deemed negligible.

We aimed to detect malicious domains via 3xx redirect responses, but our current dataset lacks such cases. However, we note that the 4xx and 5xx response codes, often associated with malicious domains, indicate server issues, either temporary or permanent, suggesting poor server configuration [23] or a domain that no longer exists.

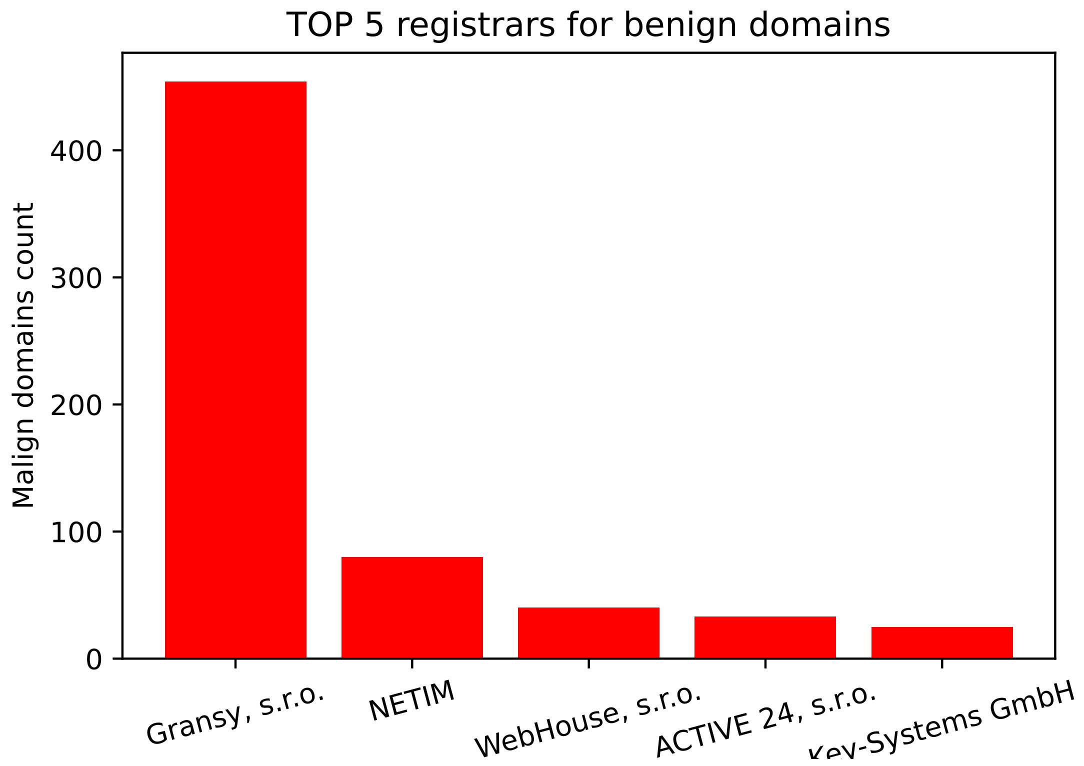

Figure 2: Malign domains

3.1.4. Lexical properties

Numerous studies have tracked attributes like domain length and specific keywords to assess domain maliciousness. We observed that malicious domains typically have longer names. Furthermore, certain words such as 'obchod', 'vypredaj', 'slovakia', 'slovensko', 'topanky' and similar terms were prevalent in malicious domains. These were often used to mimic the Slovak versions of popular fashion brand websites or to entice users with discounted merchandise offers.

We analyzed the ratios of vowels, consonants, digits, and hyphens to the total number of characters in domain names. In our dataset, domains with a high number of digits were predominantly malicious, with the exception of the benign domain '101drogeria.sk'. Hyphens were also more common in malicious domains, particularly those that mimicked well-known brands, such as 'slovak-converse.sk'. Although hyphens alone are not a definitive indicator of maliciousness, their presence, combined with the length of the above average domain name, often suggests a malicious domain. Furthermore, we observed the'sk' substring at the end of domain names as a notable attribute in fraudulent domains. For example, the malicious domain 'intimissimislovensko-sk.sk' leveraged these characteristics: long length, hyphens and specific substrings to masquerade as the benign domain 'intimissimi.com'.

We also examined lexical features such as the longest sequences of vowels, consonants, numerals, and hyphens in domain names. In Slovak domains, long sequences are uncommon, as consonants typically alternate with vowels. However, fraudulent domains aiming to appear as foreign websites often exhibited longer consonant sequences.

Additionally, we considered Levenshtein distance, which measures the similarity between two strings by calculating the minimum number of character additions, deletions, or substitutions needed to transform one into the other [24]. This metric is particularly useful for detecting typosquatting, as discussed in Section 2.2. A smaller Levenshtein distance indicates greater similarity to a legitimate domain. For example, the malicious domain 'decathlo.sk' has a Levenshtein distance of 1 from the benign domain 'decathlon.sk', as only one character addition is needed to convert the malicious domain into the benign one.

Levenshtein distance between two strings s and t of lengths m and n is defined as the minimum number of operations (insertion, deletion, or substitution of a character) required to transform string s into string t. The algorithm uses dynamic programming and constructs a matrix D of dimensions $(m+1) \times (n+1)$.

1. **Initialization**: Create a matrix D with dimensions $(m+1) \times (n+1)$. Initialize the first row and column:
$$D(i,0) = i \quad \text{for} \quad i = 0, \dots, m, \qquad D(0,j) = j \quad \text{for} \quad j = 0, \dots, n.$$
2. **Computation**: For each cell (i,j), where $i = 1, \dots, m$ and $j = 1, \dots, n$, compute the costs:
$$D(i,j) = \min \begin{cases} D(i-1,j) + 1 & \text{(deletion)}, \\ D(i,j-1) + 1 & \text{(insertion)}, \\ D(i-1,j-1) + \delta(s_{i-1}, t_{j-1}) & \text{(substitution)}, \end{cases}$$
where $\delta(s_{i-1}, t_{j-1}) = 0$ if $s_{i-1} = t_{j-1}$, otherwise $\delta(s_{i-1}, t_{j-1}) = 1$.
3. **Result**: The Levenshtein distance is the value $D(m,n)$.

We used this metric and inserted it into the dataset as a lexical attribute, observing whether it had an impact on the results of the methods used.

Detecting Unicode characters in domain names appeared promising for identifying malicious domains, particularly for typosquatting detection, but our current dataset lacks domains with Unicode characters among the malicious ones identified. Despite this, we consider Unicode detection important, as it can reveal attempts to create visually similar domain names, akin to typosquatting. For instance, attackers may use Cyrillic characters that resemble Latin ones—e.g., the Latin 'O' (U+004F) versus the Cyrillic 'O' (U+043E)—to craft deceptive domains [25].

Additionally, we examined Shannon entropy as a lexical attribute to measure the randomness of characters in a domain name [26]. This metric is particularly useful for detecting domains generated by Domain Generation Algorithms (DGAs) which often produce highly random strings indicative of malicious activity.

3.1.5. Attributes related to certificates

We tracked the provider or Certificate Authority (CA) issuing SSL certificates for domains, along with the Time To Live (TTL) of these certificates, indicating their remaining validity period. This is particularly relevant for newly created domains, as malicious ones often have short TTLs.

The most prevalent CA was Let's Encrypt [27], offering free TLS/SSL certificates. While HTTPS encryption secures data transmission, it can create a false sense of trust for users, as malicious domains exploit the browser's lock symbol, which does not guarantee a site's trustworthiness. Another notable provider was Google Trust Services [28], also issuing free certificates.

3.1.6. Geolocation attributes

The attribute concerning the server location, specifically the country hosting the domain, revealed distinct patterns in our dataset. Benign domains were frequently hosted in Slovakia or the Czech Republic, though some, like 'kaufland.sk', associated with foreign trade, originated elsewhere. In contrast, malicious domains were commonly hosted in Canada or other locations, such as the USA.

3.2. Dataset

We selected 29 relevant attributes, description in table 5 (appendix A).

3.3. Creating a list of monitored domains

To create the dataset, we first needed to collect samples to extract attributes for populating it. SK-NIC, a.s. [22], the administrator of Slovak domains, provides a comprehensive list of all '.sk' top-level domains, giving us access to all Slovak domains for inclusion. The challenge was distinguishing between malicious and benign domains.

We began by compiling a list of benign domains, manually selecting those we confirmed were non-malicious. These included domains associated with reputable brands, offices, stores, driving schools, hotels, and similar entities.

To identify malicious domains, we initially compiled a list using domains published by the Slovak Trade Inspection on its website [29]. Additionally, we sourced approximately 1,200 malicious domains identified by characteristics such as terms in their names, including 'slovakia', 'slovensko', 'bratislava', 'outlet', 'bunda', 'obuv', 'topánky', 'kabelka', 'tenisky', 'akcia', 'eshop', 'ruksak', or 'výpredaj'. Other observed attributes included grammatical errors and typosquatting [30].

Beyond categorizing domains as legitimate or malicious, we created a separate list for the Levenshtein distance attribute, incorporating global domains like 'paypal.com'. This list helps detect typosquatting attempts where attackers create illegitimate Slovak domains with names mimicking legitimate ones.

3.4. Data collection

To develop the tool, we created data collection scripts to gather attributes for training machine learning models. These scripts were implemented in Python, with separate scripts for each attribute category: WHOIS, DNS, lexical properties, headers, and location. For example, we utilized the 'requests' library [31] to send HTTP/1.1 requests and combined it with the IP-API service to retrieve the server location hosting the domain.

We consolidated the individual scripts into a single master script that sequentially executed each attribute collection function on the domain list we compiled. By importing these functions into the master script and processing the list, we used the 'pandas' library [32] to generate a CSV file containing the collected data.

The dataset contains 1799 rows and 30 attributes described in Table 5. In our dataset, each entry represents a domain. Each of them is also assigned a class, the attribute Class, which characterizes the domain as either benign (0) or malignant (1). There are 1156 benign domains and 643 malignant ones.

3.5. Selection of machine learning methods

Building on prior work, we selected three machine learning models known for high success in domain classification: Logistic Regression, Random Forest, and XGBoost, as detailed in 2.2. We chose multiple algorithms to evaluate their performance on our dataset and to address the limitations of our smaller dataset, which could affect a single model's ability to accurately classify domains. Our approach involved training each model on the training set and testing it on the test set. For new input domains, each model provides a classification, and the final label is determined by majority voting—e.g., if two of the three models classify a domain as malicious, it is labeled as such. This ensemble method enhances confidence in the classification compared to relying on a single model.

3.6. Data preprocessing

The preprocessing phase was crucial for preparing the data for machine learning models. We cleaned the data by removing or handling empty values, transformed categorical attributes using appropriate encoding, and scaled the data for the Logistic Regression model. Specifically, we removed rows with empty values, as Logistic Regression and Random Forest models cannot process them.

3.6.1. Target Encoding

Encoding transforms categorical attributes into numerical ones, as most machine learning models perform better with numerical data or cannot process categorical attributes at all. Given the high cardinality of our categorical attributes (many unique values), we applied Target Encoding. This method replaces each category with the average of the target variable—Class in our case—across all samples in that category [33].

The formula for Target Encoding can be expressed as:

$$\text{TE}(c) = \frac{\sum_{i=1}^{n_c} y_i}{n_c}$$

where:

- $\text{TE}(c)$ represents the Target Encoding for category c,
- y_i is the value of the target class for the i-th sample in category c,
- n_c is the total number of samples in category c.

During Target Encoding, we encountered an issue due to our small dataset, where certain categories had limited samples, making their relationship with the target class less reliable. This could lead the model to incorrectly assume that all domains associated with a specific category, such as location, are malicious, introducing bias. To address this, we applied a smoothing parameter, 'smooth', which adjusts the Target Encoding values by incorporating the global target average.

The formula for smoothed Target Encoding is:

$$\text{TE}_{\text{smoothed}}(c) = \frac{n_c \cdot \text{TE}(c) + \alpha \cdot \bar{y}_{\text{global}}}{n_c + \alpha}$$

where:

- $\text{TE}_{\text{smoothed}}(c)$ is the smoothed Target Encoding for category c,
- n_c is the number of samples in category c,
- $\text{TE}(c)$ is the unsmoothed Target Encoding value for category c,
- α is the smoothing factor,
- $\bar{y}_{\text{global}}$ is the global average of the target attribute (Class) across the entire dataset.

3.6.2. Standardisation

For many models, such as Logistic Regression or Support Vector Machines (SVM), dataset standardization is essential, as these models perform better when sample values are centered around zero. Without standardization, models may face convergence issues, meaning they cannot properly learn from the data.

The standard score, or z-score, is calculated for each sample using the formula:

$$z = \frac{x - \mu}{s}$$

where:

- x is the sample value,
- μ is the mean of the training samples,
- s is the standard deviation of the training samples.

We compute the mean and standard deviation separately for each attribute (column) and apply these values to the samples in that attribute. To standardize our dataset, we used the StandardScaler [34] from the scikit-learn library.

3.7. Hyperparameter tuning

Hyperparameters are model settings defined by the programmer, and optimizing them can enhance model accuracy. Tools like GridSearchCV [35] and RandomizedSearchCV [36] from scikit-learn automate the process of finding the optimal hyperparameter combination. We provide a set of hyperparameters in the input grid, and the algorithm tests various combinations to identify the best-performing one. GridSearchCV evaluates all possible combinations, while RandomizedSearchCV tests a user-specified number of random combinations, selecting the one with the highest accuracy. The latter is faster and more efficient for large sets of combinations, as it reduces computational demands.

To optimize hyperparameter selection, we employed 10-fold cross-validation, where the dataset is divided into ten equal subsets. The model is trained on nine subsets and tested on the remaining one, generating a validation score for each subset. The overall validation score is computed as the average of these scores. The hyperparameters yielding the highest average validation score are selected as the optimal configuration.

To evaluate model performance, we used cross-validation with ten subsets for each hyperparameter combination. The score for a given combination is the average of the validation scores across these subsets. Multiple hyperparameter combinations were tested, and the one yielding the highest score was selected as the final configuration.

For **Logistic Regression**, the optimal hyperparameters identified by GridSearchCV were:

- **Penalty**: `l1` – regularization method to mitigate overfitting.
- **Solver**: `liblinear` – algorithm used for optimization in logistic regression.
- **C**: 1.0 – inverse of regularization strength.

For the **Random Forest** model, the optimal hyperparameters determined by RandomizedSearchCV were:

- **n_estimators**: 300 – number of trees in the forest.
- **max_depth**: None – no limit on the depth of trees.
- **max_features**: `log2` – maximum number of features considered for node splitting.
- **min_samples_leaf**: 1 – minimum number of samples in a leaf node.
- **min_samples_split**: 2 – minimum number of samples required to split a node.
- **bootstrap**: False – data points are sampled without replacement.

We utilized RandomizedSearchCV to optimize the hyperparameters for the **XGBoost** model, resulting in the following configuration:

- **n_estimators**: 350 – number of trees in the model.
- **max_depth**: 100 – maximum depth of each tree.
- **subsample**: 0.9 – proportion of data used for training each tree under Boosting.
- **colsample_bytree**: 0.3 – proportion of features selected for each tree.
- **gamma**: 0.3 – minimum loss reduction required to split a node.
- **min_child_weight**: 1 – minimum number of data points in a child node.
- **reg_alpha**: 0 – L1 regularization to reduce overfitting.
- **reg_lambda**: 1 – L2 regularization to reduce overfitting.

3.8. Training and testing models

After data cleaning, preprocessing, and hyperparameter tuning, we proceeded with training and testing the models. The dataset was split into training and test sets using the ‘train_test_split‘ function [37], allocating 60% for training and 40% for testing. Initially, an 80%/20% split was used, but we adjusted to a 60%/40% split to assess model performance with a larger test set. The scikit-learn library was used to implement and train the models, except for XGBoost, which utilized the xgboost library [38]. Three models—Logistic Regression, Random Forest, and XGBoost—were defined, trained, and tested. Their performance evaluation is discussed in the following chapter.

3.9. Creating a Domain Analysis Tool

Before building the tool, functional models were critical, as they underpinned much of its functionality. We exported the trained models to separate files using the joblib library [39] for integration into the tool. Additionally, we processed data for scaling and Target Encoding to ensure input data matched the format required by the models.

The tool was developed using the argparse library [40], enabling user interaction via the command line. Upon launching, users input the domain name to be classified. Relevant features, consistent with those in the training dataset, are extracted from the domain using the same feature extraction process. Categorical attributes undergo Target Encoding, and data for logistic regression is scaled using saved scaling files. The processed data is then fed into the models, which classify the input domain.

We utilized the SHAP tool to assess the importance of attributes in our dataset and to identify which attributes influenced the classification of specific samples. This approach helps pinpoint the sources of errors, such as false positives or false negatives, by highlighting the contributing attributes.

When a user inputs a domain name, relevant features are extracted to serve as input for the machine learning models. The domain is then classified by three models—Logistic Regression, Random Forest, and XGBoost—each providing its prediction on the domain's class. A majority voting mechanism determines the final class, requiring at least two models to agree. The classification result is displayed to the user in the console, along with the key attributes influencing the decision, as determined by the SHAP method.

4. Results and discussion

4.1. Oversampling and undersampling

These techniques address the issue of imbalanced datasets, where one class has significantly more samples than another. Oversampling increases the number of samples in the minority class, while undersampling reduces the number of samples in the majority class.

Lemaître et al. [41] developed the imbalanced-learn [42] library to tackle the common problem of imbalanced datasets. This solution enhances model learning by balancing class sample counts, enabling models to better identify patterns for training. As the library is compatible with our chosen `scikit-learn` library, we opted to use it to increase the number of samples in the malicious domain class. After applying oversampling, we observed a slight improvement in model accuracy.

We documented the accuracy comparison in Table 1:

Table 1
Model Accuracy Comparison Before and After Oversampling

Model	Before Oversampling	After Oversampling
LR	0.945	0.955
RF	0.958	0.966
XGBoost	0.955	0.962

4.2. SHAP

SHAP (SHapley Additive exPlanations) [16] identifies the attributes most influential in classification outcomes by calculating Shapley values, which quantify each attribute's average contribution to the result across all possible combinations of other attributes. It evaluates how the inclusion of a specific attribute affects the outcome by analyzing its impact within various attribute combinations. The SHAP value for an attribute is derived by averaging these contributions. We used this method to assess the importance of selected attributes in our dataset and integrated it into our attribute evaluation tool to provide insights into the classification of specific domains.

Table 2 lists the ten most influential attributes in our dataset along with their Shapley values:

Table 2
Top Ten Most Influential Attributes

#	Attribute Name	Shapley Value
1	Registrar	1.721
2	MX	1.497
3	Contains_Blacklisted	0.792
4	IPv4_Num	0.655
5	Levenshtein_Distance	0.55
6	Server	0.486
7	Location	0.364
8	SSL_TTL	0.359
9	Update_Date	0.356
10	AS	0.325

When evaluating attribute contributions for a single domain, such as during tool execution, the SHAP evaluation results for the input domain “alza.sk” are shown below. For simplicity, we list only the top five attributes per model:

Logistic Regression Model:

- Registrar: -1.4109
- Levenshtein_Distance: -1.3126
- Location: 0.6238
- Server: 0.5950
- DMARC: -0.5747

Random Forest Model:

- Creation_Date: -0.1374
- Registrar: 0.1374

XGBoost Model:

- Registrar: -1.1744
- MX: -1.0401
- Server: 0.8987
- Entropy: 0.7367
- SSL_TTL: -0.7143

The results can be interpreted as follows: positive Shapley values indicate that the attribute influenced the model to classify the domain as Class 0 (benign), while negative values suggest a tendency toward Class 1 (malicious).

For the Random Forest model, only two attributes are reported. This is because SHAP is less compatible with Random Forest when analyzing a single sample (local interpretation) as opposed to the entire dataset (global interpretation), which can lead to inaccuracies. An alternative approach would be to use the feature_importance_ method [43], which, unlike SHAP, does not indicate the direction of influence toward a specific class but only quantifies the attribute’s overall role in classification.

4.3. Evaluation of machine learning models

After training the models on our preprocessed dataset, we evaluated their performance using 10-fold cross-validation, which provides the average accuracy of each model. The results are shown in Table 3:

Table 3
Comparison of Machine Learning Model Accuracies

Model	LR	RF	XGB
Accuracy	0.945	0.975	0.974

From these results, we can conclude that the Random Forest (RF) and XGBoost (XGB) models performed best. If we were to redesign the tool to use only one model, we would likely choose either Random Forest or XGBoost.

In Table 4, we evaluated the models using various metrics during a single measurement. The results show that Random Forest (RF) and XGBoost (XGB) again outperformed, though Logistic Regression (LR) also achieved respectable results. The main issue with Logistic Regression is a slightly higher number of false negatives (FN) compared to false positives (FP). For our purposes, it is preferable for a model to erroneously classify a domain as malicious (FP), as such cases can be manually verified. However, if a malicious domain is mistakenly classified as benign (FN), it may not undergo further scrutiny, potentially remaining publicly accessible and posing an ongoing threat.

To evaluate the performance of a classification model several metrics are used [9], e.g. Precision, recall (sensitivity) and the F1-score (particularly in scenarios where the classes are imbalanced).

Table 4
Evaluation of Machine Learning Models for a Single Measurement

Metric	LR	RF	XGB
Accuracy	0.95492	0.96311	0.96175
Precision	0.95493	0.96316	0.96190
Recall	0.95492	0.96311	0.96175
F1-Score	0.95491	0.96312	0.96176
TP	331/732	336/732	338/732
TN	368/732	369/732	366/732
FP	15/732	14/732	17/732
FN	18/732	13/732	11/732

5. Conclusions

We applied the trained models within our CLI tool, where the user writes the name of the domain they want to analyze. The result is a classification of the domain by all three models and their voting for the domain class. In addition to the class, the tool also returns the attributes according to which it classified the domain.

The dataset could be enlarged by analyzing new domains, which could also be used in other works. In the future, we could also supplement the analysis with other attributes and classification into multiple classes, e.g. phishing, malware, C&C and others. Another functionality that could be beneficial for the tool in the future could be the analysis of page content using Natural Language Processing.

6. Acknowledgments

This paper was supported by the Slovak Research and Development Agency under contract No. APVV-23-0137 .

Declaration on Generative AI

The authors have not employed any Generative AI tools.

References

[1] J. Ma, L. K. Saul, S. Savage, G. M. Voelker, Beyond blacklists: learning to detect malicious web sites from suspicious urls, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 1245–1254.

[2] S. Bell, P. Komisarczuk, An analysis of phishing blacklists: Google safe browsing, openphish, and phishtank, in: Proceedings of the Australasian Computer Science Week Multiconference, 2020, pp. 1–11.

[3] I. Skula, M. Kvet, Domain blacklist efficacy for phishing web-page detection over an extended time period, in: 2023 33rd Conference of Open Innovations Association (FRUCT), IEEE, 2023, pp. 257–263.

[4] Y. Fukushima, Y. Hori, K. Sakurai, Proactive blacklisting for malicious web sites by reputation evaluation based on domain and ip address registration, in: 2011 IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications, IEEE, 2011, pp. 352–361.

[5] T. Meng, X. Jing, Z. Yan, W. Pedrycz, A survey on machine learning for data fusion, Information Fusion 57 (2020) 115–129.

[6] T. Jo, Machine Learning Foundations. Supervised, Unsupervised, and Advanced Learning, Springer International Publishing, Cham, 2021.

[7] C. Starbuck, Logistic regression, in: The Fundamentals of People Analytics, Springer, Cham, 2023. doi:10.1007/978-3-031-28674-2_12.

[8] G. Palaniappan, S. Sangeetha, B. Rajendran, S. Goyal, B. S. Bindhumadhava, Malicious domain detection using machine learning on domain name features, host-based features and web-based features, Procedia Computer Science 171 (2020) 654–661.

[9] M. Bansal, A. Goyal, A. Choudhary, A comparative analysis of k-nearest neighbor, genetic, support vector machine, decision tree, and long short term memory algorithms in machine learning, Decision Analytics Journal 3 (2022) 100071.

[10] S. L. Salzberg, Programs for machine learning by j. ross quinlan, Mach Learn 16 (1994) 235–240.

[11] Y. Zhao, Y. Zhang, Comparison of decision tree methods for finding active objects, Advances in Space Research 41 (2008) 1955–1959.

[12] L. Bilge, E. Kirda, C. Kruegel, M. Balduzzi, Exposure: Finding malicious domains using passive dns analysis, in: NDSS, 2011, pp. 1–17.

[13] K. A. Messabi, M. Aldwairi, A. A. Yousif, A. Thoban, F. Belqasmi, Malware detection using dns records and domain name features, in: Proceedings of the 2nd International Conference on Future Networks and Distributed Systems, 2018, pp. 1–7.

[14] D. Nielsen, Tree boosting with xgboost - why does xgboost win every machine learning competition?, Master's thesis, NTNU, 2016.

[15] A. Horák, Malicious domain detection from external data sources, in: IEEE International Conference on Intelligence and Security Informatics (ISI), 2018, pp. 220–225.

[16] SHAP project, Projekt shap, 2025. URL: https://shap.readthedocs.io/en/latest/, accessed: 2025-04-19.

[17] S. Hamadouche, O. Boudraa, M. Gasmi, Combining lexical, host, and content-based features for phishing websites detection using machine learning models, EAI Endorsed Transactions on Scalable Information Systems 11 (2024) 1–15.

[18] M. Agarwal, B. Agarwal, Comparative analysis of classifier performance: A study of weka and python implementations across various machine learning models, 2024.

[19] G. A. Bansal M., C. A., A comparative analysis of k-nearest neighbor, genetic, support vector machine, decision tree, and long short term memory algorithms in machine learning, Decision Analytics Journal 3 (2022) 100071. URL: https://www.sciencedirect.com/science/article/pii/S2772662222000261. doi:https://doi.org/10.1016/j.dajour.2022.100071.

[20] J. Ma, L. K. Saul, S. Savage, G. M. Voelker, Identifying suspicious urls: an application of large-scale online learning, in: Proceedings of the 26th Annual International Conference on Machine Learning, 2009, pp. 681–688.

[21] C. Marques, S. Malta, J. P. Magalhães, Dns dataset for malicious domains detection, Data in

Brief 38 (2021) 107342. URL: https://www.sciencedirect.com/science/article/pii/S2352340921006260. doi:https://doi.org/10.1016/j.dib.2021.107342.

[22] SK-NIC, Spoločnosť sk-nic, 2025. URL: https://sk-nic.sk/, accessed: 2025-04-19.

[23] A. Lavrenovs, F. J. R. Melón, Http security headers analysis of top one million websites, in: 2018 10th International Conference on Cyber Conflict (CyCon), IEEE, 2018, pp. 345–370.

[24] R. Haldar, D. Mukhopadhyay, Levenshtein distance technique in dictionary lookup methods: An improved approach, 2011. URL: https://arxiv.org/abs/1101.1232, arXiv preprint arXiv:1101.1232.

[25] A. M. Almuhaideb, N. Aslam, A. Alabdullatif, S. Altamimi, S. Alothman, A. Alhussain, K. A. Alissa, Homoglyph attack detection model using machine learning and hash function, Journal of Sensor and Actuator Networks 11 (2022) 54.

[26] A. D. Wong, Detecting domain-generation algorithm (dga) based fully-qualified domain names (fqdns) with shannon entropy, 2023. URL: https://arxiv.org/abs/2304.07943, arXiv preprint arXiv:2304.07943.

[27] Let's Encrypt, Let's encrypt, 2025. URL: https://letsencrypt.org/, accessed: 2025-04-19.

[28] Google, Google trust services, 2025. URL: https://pki.goog/, accessed: 2025-06-22.

[29] Slovenská obchodná inšpekcia, Zoznam podvodných internetových stránok, 2025. URL: https://www.soi.sk/sk/informacie-pre-verejnost/internetove-obchody/podvodne-internetove-stranky-1.soi, accessed: 2025-04-19.

[30] Ministerstvo na kontrolu, Zoznam falošných e-shopov a phisingových stránok, 2024. URL: https://mnk.sk/fake/, accessed: 2024-03-07.

[31] Python Software Foundation, Projekt requests, 2025. URL: https://pypi.org/project/requests/, accessed: 2025-04-19.

[32] pandas development team, Projekt pandas, 2025. URL: https://pandas.pydata.org/, accessed: 2025-04-19.

[33] scikit-learn developers, class targetencoder, 2025. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.TargetEncoder.html, accessed: 2025-04-19.

[34] scikit-learn developers, class standardscaler, 2025. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html, accessed: 2025-04-19.

[35] scikit-learn developers, Modul gridsearchcv, 2025. URL: https://scikit-learn.org/stable/modules/grid_search.html, accessed: 2025-04-19.

[36] scikit-learn developers, Modul randomizedsearchcv, 2025. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html, accessed: 2025-04-19.

[37] scikit-learn developers, Modul train_test_split, 2025. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html, accessed: 2025-04-19.

[38] XGBoost project, Projekt xgboost, 2025. URL: https://xgboost.readthedocs.io/en/stable/install.html, accessed: 2025-04-19.

[39] Joblib project, Projekt joblib, 2025. URL: https://joblib.readthedocs.io/en/stable/, accessed: 2025-04-19.

[40] Python Software Foundation, Projekt argparse, 2025. URL: https://docs.python.org/3/library/argparse.html, accessed: 2025-04-19.

[41] G. Lemaître, F. Nogueira, C. K. Aridas, Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning, Journal of Machine Learning Research 18 (2017) 1–5.

[42] imbalanced-learn project, Projekt imbalanced-learn, 2025. URL: https://imbalanced-learn.org/, accessed: 2025-04-19.

[43] scikit-learn developers, Metóda feature_importance_, 2025. URL: https://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html, accessed: 2025-05-05.

A. Domain attributes

Table 5
Description of Domain Attributes

Attribute	Description
Domain	The domain name (e.g., example.com) that identifies a website or service.
Registrar	The organization managing the domain's registration (e.g., GoDaddy, Namecheap).
Creation_Date	The date when the domain was first registered.
Update_Date	The date of the last update to the domain's registration (e.g., ownership change or renewal).
SSL_Issuer	The issuer of the SSL certificate securing the connection (e.g., Let's Encrypt, DigiCert).
SSL_TTL	The validity period of the SSL certificate (in days), indicating how long it remains valid.
IPv4_Num	The number of IPv4 addresses associated with the domain.
NumOfIPs	The total number of IP addresses (IPv4 and IPv6) linked to the domain.
SPF	Sender Policy Framework, a record specifying which servers can send emails on behalf of the domain.
MX	Mail Exchange, a record indicating the email servers for the domain.
DKIM	DomainKeys Identified Mail, a mechanism for verifying email authenticity using cryptographic signatures.
DMARC	Domain-based Message Authentication, Reporting, and Conformance, a policy for email authentication and handling suspicious messages.
AS	Autonomous System, the number identifying the network hosting the domain.
HTTP_Status	The HTTP status code (e.g., 200 OK, 404 Not Found) indicating the result of a server request.
Server	The type of server hosting the domain (e.g., Apache, Nginx).
Entropy	The entropy of the domain name, a measure of character randomness, often used to detect suspicious domains.
Vowel_Ratio	The ratio of vowels in the domain name, expressed as the proportion of vowels to total characters.
Consonant_Ratio	The ratio of consonants in the domain name, expressed as the proportion of consonants to total characters.
Numerical_Ratio	The ratio of digits in the domain name, expressed as the proportion of digits to total characters.
Special_Char_Ratio	The ratio of special characters (e.g., hyphens, underscores) in the domain name.
Vowel_Sequence	The longest sequence of consecutive vowels in the domain name.
Consonant_Sequence	The longest sequence of consecutive consonants in the domain name.
Numerical_Sequence	The longest sequence of consecutive digits in the domain name.
Special_Char_Sequence	The longest sequence of consecutive special characters in the domain name.
Is_Unicode	An indicator of whether the domain name contains Unicode characters (e.g., diacritics or non-standard characters).
Levenshtein_Distance	The Levenshtein distance, a measure of the difference between the domain name and a reference domain, used to detect similar domains.
Contains_Blacklisted	An indicator of whether the domain contains words or patterns from a blacklist.
Last_is_invalid	An indicator of whether the last part of the domain (e.g., TLD) is invalid or non-existent.
Location	The geographic location of the server hosting the domain (e.g., country).
Class	The classification of the domain (e.g., legitimate, suspicious, malicious), often used in security analysis.

B. Online Resources

All sources are published on GitHub CLI Project .