DOI: 10.63550/ICEIP.2025.10.95.033

Classifying PowerShell Commands Using Machine Learning Algorithms

Grigorev Maksim [1] [a], Nikolskaia Kseniia [2,3], Shtyrkina Anna [4] [b], Varkentin Vitalii [1] [c] and Minbaleev Aleksey [5] [d] [e]

[1] *School of Electronic Engineering and Computer Science, South Ural State University (national research university), Lenin prospekt, Chelyabinsk, Russia*

[2] *Scientific and educational center "Legal regulation in the field of high technologies", Kutafin Moscow State Law University (MSAL), Moscow, Russia*

[3] *OOO Datagile (limited liability company), Saint Petersburg, Russia*

[4] *OOO R-Vision (limited liability company), Enthusiasts Boulevard, Moscow, Russia*

[5] *Department of Information Law and Digital Technologies, Kutafin Moscow State Law University (MSAL), Moscow, Russia*

ya.iridium2018@yandex.ru, nikolskaya174@gmail.com, ashtyrkina@rvision.ru, varkentinvv@susu.ru, alexmin@bk.ru

Keywords: Trusted AI, machine learning, PowerShell, malicious command analysis, trusted AI platform, MITRE, MITRE ATT&CK.

Abstract: This study presents a web application for classifying PowerShell commands using machine learning algorithms using Scikit-learn and XGBoost libraries. Classification was performed using the MITRE ATT&CK (MITRE) matrix into two classes (malicious and legitimate). The dataset includes 1,743 commands and was labeled by cybersecurity experts. The methods "Bag of Words" and TF-IDF using the Scikit-learn library. Were trained and tested such algorithms such as Naïve Bayes, Decision Tree, Random Forest, XGBoost, Logistic Regression, SVM and K-Nearest Neighbours. The average accuracy of all machine learning algorithms was (0.98 by the F1-score metric) even on small data sets (300 instances). The web application is implemented on the framework Streamlit. For ease of deployment, the application is packaged in a Docker container, which guarantees stable operation in any environment. The study has practical significance for cybersecurity specialists in terms of: a unique dataset with PowerShell commands marked up according to the MITRE matrix was created; a web application for automatic analysis of PowerShell commands was created; a web application for training machine learning algorithms was created.

1 INTRODUCTION

In today's world, there are many threats, challenges, and opportunities that come from the synergy of machine learning and cybersecurity technologies (Salem, 2024) (Shafin, 2023) (Prasad, 2023) (Wazid, 2022). The study (Owa, 2025) uses Random Forest, Support Vector Machines and Decisions Tree for analyzing phishing detection and classifying safe URLs. The dataset contained over 640,000 URLs. According to the study, the Random algorithm Forest showed the highest accuracy, and it was 87.85% on the validation set. Independent testing was carried out on the Aalto dataset (Kaggle), the accuracy was 86.86%. The study [Wu, 2025] describes cybersecurity entity recognition model (CERM). This model is developed to accurately identify cybersecurity entities in IoT. CERM uses a hierarchical attention mechanism that finds dependencies between cybersecurity entities which can identify cybersecurity entities in IoT. The model architecture is presented as a CNN-BiLSTM-CRF.

[a] https://orcid.org/0009-0008-1874-8558
[b] https://orcid.org/0000-0002-2141-6780
[c] https://orcid.org/0000-0003-3069-2258
[d] https://orcid.org/0000-0001-5995-1802
[e] https://orcid.org/0000-0001-8335-2084

The study used two datasets: DNRTI - a publicly available dataset; OpenData - a dataset collected by researchers from security documents. According to the F1-score metric, the accuracy was the highest, equal to 92.86. The study [Bharath, 2025] considers the problems of adversarial attacks on machine learning algorithms, raises issues of confidentiality of training datasets, as well as issues of interpreting the results of machine learning algorithms. PowerShell is both a command line and a scripting language that is installed by default on computers running the Windows operating system. Despite its legitimate source, PowerShell is very often used by attackers for various malicious activities. Command-line attacks make up the majority of malicious commands described in the MITRE matrix. MITRE is a matrix of cyberattack tactics and techniques. Cybersecurity researchers use MITRE to assess threats and develop strategies to protect an information system [Georgiadou, 2021]. A study [Hendler, 2018] developed classifiers of PowerShell commands based on natural language processing (NLP), based on convolutional neural networks (CNNs) and a hybrid detector of several models. According to the results of the study, the best result was shown by a hybrid detector, which consists of several NLP models and CNNs. Analysis of PowerShell commands is a very promising area of research, since such analyzers are used in modern cybersecurity systems of the User class and Entity Behavior Analytics (UEBA) and Security Information and Event Management (SIEM) [Avdoshin, 2019]. The study [Barrué, 2025] describes two implemented quantum machine learning algorithms that were used for malicious software detection. There are not many studies in the area of analyzing PowerShell commands using machine learning algorithms. This is due to the fact that obtaining a data set that will contain legitimate and malicious PowerShell commands is a complex and labor-intensive job. As the researchers themselves write. This fact confirms the relevance of this study in terms of contributing to the study of the subject area. There are also a small number of studies in which the marking of PowerShell commands was made according to the MITRE matrix.

This study describes the first prototype of a platform that will allow researchers to:

- upload your datasets to have them automatically checked.
- cybersecurity datasets.
- cybersecurity models with hyperparameter tuning and using various machine learning libraries.
- download trusted trained machine learning and quantum machine learning algorithms.

The study has the following structure. Section 2 describes the dataset and its preprocessing. Section 3 describes the application design stages, which included a description of the functional and non-functional requirements, and also presents a diagram of the system components. Section 4 describes the application implementation, which includes a description of the technical implementation of the system, as well as the machine learning algorithms that were implemented in this study. Section 5 describes the experiments and results. Section 7 describes the conclusions of the study and future work.

2 DATASET

For the first dataset in the field of cybersecurity, a dataset was selected that contains PowerShell interpreter commands. The dataset includes legitimate and malicious commands that a user can enter into the command line. Examples of commands are presented in Table 1.

Table 1: Example commands from the dataset.

Team	What does?
Get -ADDomain	Used to get information about the current Active domain. Directory. Not malicious.
Get-Process	Used to obtain a list of running processes on a local or remote computer. Not malicious.
Invoke-Expression	Used to execute a string expression as PowerShell code. Not malicious.
Wait Job	Used to wait for a background task (Job) to complete before executing further commands. Not malicious.
Import-Module BitsTransfer	This command loads the BitsTransfer module, which provides commands to control file transfers using Background. Intelligent Transfer Service (BITS) is malicious.
Start- BitsTransfer - Source $ url - Destination $output	used to download files from the network or copy data from a remote server

	to a local computer using Background Intelligent Transfer Service (BITS), where - Source $ url – specifies the link to the file to download (HTTP, HTTPS, SMB, FTP), - Destination $ output – the path to save the downloaded file. It is malicious.
Powershell -version 2	Used to run an older version of PowerShell, but it is no longer supported or recommended on modern Windows due to security issues. It is malicious.
$ a.rulecollections	This command can be used in scenarios related to Windows Firewall because the rulecollections property is found in objects returned by cmdlets. Get-NetFirewallPolicy or Get-WindowsFirewallSetting. Is malicious.

The data set was then labeled into two classes: malicious commands (class "1" in the data set) and legitimate commands (class "0" in the data set). The labeling was performed by cybersecurity experts according to the MITRE matrix. Malicious commands comprised 920 instances. Legitimate commands comprised 823 instances. A portion of the labeled data set is presented in Table 2.

Table 2: Part of the labeled dataset.

Team	Class
Where-object	0
get-service	0
C:\Windows\system32\lpremove.exe	0
ctfmon.exe	0
C:\Windows\system32\ipconfig /renew	1
C:\Windows\system32\net session	1
Import-Module <modulepath>	1
Get-Help process	1

The dataset consists of 1,743 command records. The training dataset consists of 1,220 commands. The test dataset consists of 435 commands. The validation dataset consists of 88 commands. The commands were manually entered into the datasets to ensure diversity across all three datasets.

The dataset was saved in. csv format for further convenience of training machine learning models.

For the model to work, the original data set had to be preprocessed. Since the original data is string, it was decided to use the "Bag" model as a preprocessing. of words" (bag of words). A bag of words is a model for text processing, which is an unordered set of words included in the processed text. The Scikit-learn library was chosen as a preprocessing tool. To implement the bag of words from the Scikit-learn library, two classes were used: CountVectorizer [CountVectorizer, 2025], which allows you to transform a collection of text documents into a matrix of the number of tokens; TfidfVectorizer [TfidfVectorizer, 2025], which allows you to transform a collection of raw documents into a TF-IDF feature matrix (Term Frequency-Inverse Document Frequency). TF-IDF calculates the importance of each word in a document relative to the number of its occurrences in the document and in the entire text collection, this method allows you to highlight keywords and understand which words have more weight for a particular document in the context of the entire collection. For the full operation of the CountVectorizer and TfidfVectorizer classes, you need to use another one, it is called GridSearchCV. This class allows you to conduct a study to find the optimal parameters for the CountVectorizer and TfidfVectorizer classes for each machine learning algorithm. The GridSearchCV class itself determines which of the classes to use based on internal calculations. It does not transform data with two vectorizers at once, only one that will show the best result for each machine learning algorithm.

3 APPLICATION DESIGN

3.1 Functional and non-functional requirements

Before you start implementing an application, you need to determine the functional and non-functional requirements that it must meet.

3.1.1 Functional requirements

Functional requirements are requirements for an application that describe how the application should behave in various cases [Sindhgatta, 2005].

The following functional requirements were developed for this application:

- The application must implement functionality for configuring hyperparameters of machine learning models.
- The application must implement functionality for saving classification results in a .csv file.
- The application must implement functionality for selecting different trained machine learning models.
- The application must implement functionality for classifying data into two classes: malicious command and legitimate command.
- The application must implement functionality for downloading the trained model.
- The application must implement functionality for displaying information about classification results.
- The application should implement functionality for selecting different options for classifying commands: using trained machine learning algorithms that are loaded into the application in advance; training your own machine learning model with parameter settings.

3.1.2 Non-functional requirements

Non-functional requirements are those requirements for an application that directly relate to what the final result should be and what means it should be achieved [Sindhgatta, 2005].

The following non-functional requirements were developed for this application:

- Final presentation of the application: web application. It was decided to choose the implementation option of the web application, because this type of presentation is convenient for the end user.
- Framework: Streamlit [https://streamlit.io/]. The framework chosen for the backend was Streamlit, as it allows you to quickly make a web application, since all forms are pre-written by the developers of the framework. Streamlit provides high-quality work with data visualization for training and various machine learning libraries.
- Programming language and version: Python 3.12. Python was chosen as the programming language because it is the main language for developing systems with machine learning algorithms.
- End-Use Application: The application should be deployed from a Docker container. This is necessary to ensure ease of testing by various end users, as well as further use for mass consumption.
- Library for training machine learning algorithms: Scikit-learn [Scikit-learn, 2025] and Xgboost [Xgboost, 2025]. At the first stage of implementation, the Scikit-learn library was chosen, since it has greater adaptability for tuning hyperparameters of machine learning models, and also has high performance. The Xgboost library was used to implement the XGBClassifier algorithm, since this library is the best for creating boosting models.
- The format of the loaded and processed data sets is. csv. At the first stage, it was decided to support only one format. csv, since it is universal for the tasks of classifying statistical information.

3.1.3 System Component Diagram

Unified is often used for application design. Modeling Language (UML) is a unified modeling language. This modeling language can describe various components of a system, considering them from different angles. One of such descriptions is a system component diagram. This diagram describes the system components that are used in the application, as well as how they interact with each other. The diagram is shown in Figure 1.

Figure 1: System component diagram.

The developed application consists of three components:

- Streamlit App is an application written in Python that uses the Streamlit API.
- Application Streamlit In A Docker Container is an application that contains the entire application with all its components.
- User 's Browser is the user's web browser through which he must interact with the system.

The algorithm of the application:

- The user's browser is used to display the application interface.
- The application image is created using Docker.
- The container with the Streamlit application is launched.

- max_features = “sqrt”

The selection of optimal values was carried out using the GridSearchCV of Scikit-learn library.

4 APPLICATION IMPLEMENTATION

4.1 Implementation and training of machine learning algorithms

For the first stage of the study, the following machine learning algorithms were selected: Decision Tree, Naive Bayes, Random Forestry, Logistics Regression, XGBClassifier, Support Vector Machines and K- Nearest Neighbours.

Most of the machine learning models were implemented using the Scikit-learn library, except for the algorithm, it was implemented using the Xgboost library.

hyperparameters were selected for each machine learning model.

4.1.1 Naïve Bayes

Naive Bayes is a machine learning algorithm that is based on the Thomas Bayes theorem. This machine learning algorithm is used when it is necessary to calculate the probability of an event occurring.

Naïve Algorithm Bayes was implemented using the GaussianNB class of the Scikit-learn library. The hyperparameters used are:

- priors
- var_smothing

Two hyperparameters were used for the model, so the selection of optimal values was carried out using GridSearchCV of the Scikit-learn library and was applied only to the vectorizer.

4.1.2 Decision Tree

Decision Tree is a machine learning algorithm that is a hierarchical structure consisting of rules of the form “If …, then …”.

Decision Algorithm Tree was implemented using the Decision class Tree Classifier of the Scikit-learn library. The hyperparameters used are:

- criterion = “gini”
- splitter = “best”
- max_depth = “15”
- min_samples_split = “3”
- min_samples_leaf = “1”

4.1.3 Random Forest

Random Forest is a bagging algorithm that works by creating multiple copies of the training data, each copy being at least slightly different from the others. It then trains multiple machine learning models in parallel and then combines them.

Random Algorithm Forest was implemented using the Random class. Forest Classifier library Scikit-learn. In quality hyperparameters used:

- n_estimators = “200”
- bootstrap = “False”
- max_depth = “30”
- max_features = “sqrt”
- min_samples_leaf = “1”
- min_samples_split = “2”

The selection of optimal values was carried out using the GridSearchCV library of Scikit-learn.

4.1.4 XGBoost

XGBoost is a boosting algorithm that uses many simple models to create an ensemble model. The basic idea behind the algorithm is that new models are added to the ensemble sequentially, which allows for the correction of errors made by previous models.

XGBoost algorithm was implemented using the XGBClassifier class of the Xgboost library. The hyperparameters used are:

- colsample_bytree = “0.8”
- gamma = “0.75”
- learning_rate = “0.2”
- max_depth = “7”
- min_child_weight = “2”
- n_estimators = “200”
- subsample = “1.0”

4.1.5 Logistic Regression

Logistic Regression is a machine learning method that calculates the probability of a certain event occurring based on values that belong to a set of features.

Logistic Algorithm Regression was implemented using the XGBClassifier class of the Scikit-learn library. The hyperparameters used are:

- C = “100”

- class_weight = “balanced”
- fit_intercept = “True”
- intercept_scaling = “0.01”
- penalty = “12”
- solver = "lbfgs"

The selection of optimal values was carried out using the GridSearchCV of Scikit-learn library.

4.1.3 Support Vector Machines

Support Vector Machines (SVM) is a supervised learning algorithm used to solve classification and regression problems. The peculiarity of the SVM algorithm is that each object from the training data set must be represented as a feature vector. Then, all objects are placed on the coordinate axis. The task of the algorithm is to construct a hyperplane that will allow dividing all objects into two classes.

Support Algorithm Vector Machines was implemented using the SVC class of the Scikit-learn library. The hyperparameters used are:

- gamma = “scale”
- kernel = “rbf”
- C = “10.0”

The selection of optimal values was carried out using the GridSearchCV library of Scikit-learn.

4.1.7 K - Nearest Neighbors

Nearest Algorithm Neighbors (kNN) is a supervised learning algorithm used to solve classification and regression problems. This algorithm only works with numerical features, so converting commands to vector form allows it to be used in this study. The main idea of the algorithm is that it predicts the class label based on the distance to the nearest neighbors.
kNN algorithm was implemented using the KNeighborsClassifier class of the Scikit-learn library. The hyperparameters used are:

- algorithm = “auto”
- leaf_size = “30”
- n_neighbors = “3”
- p = “3”
- weights = “distance”

The selection of optimal values was carried out using the GridSearchCV library of Scikit-learn.

4.2 Implementation of the web client

According to the non-functional requirements, the user interface was implemented as a web client. A web client is software that runs on the user's end computer, and it must also ensure that the software works with the end user. The framework was chosen as the backend Streamlit. Streamlit is a tool for quickly creating web applications without in-depth knowledge of the frontend.

It was originally developed for data analysts and ML engineers who need to visualize models and algorithms in an interactive format without spending time on complex layout or backend development.

Key Features:

- Ease of use – an application on Streamlit can be written in literally a few lines of code.
- Compatibility with the Python ecosystem – works with popular libraries: Pandas, NumPy, XGBoost / LightGBM / CatBoost, Scikit-learn, Seaborn, Matplotlib, Plotly, TensorFlow, PyTorch and others.
- Interactive elements – supports sliders, buttons, charts and other UI components out of the box.

What is it used for?

- ML Model Demo - Quickly deploy a web interface for predictions.
- Data analysis – interactive dashboards without complex code.
- Prototyping is an instant visualization of ideas without full-fledged front-end development.

Streamlit allows you to turn Python scripts into web applications in minutes, making ML development more visual and accessible. The final version of the software should implement features that allow the user to work with machine learning algorithms, such as selecting an algorithm, tuning the hyperparameters of the selected algorithm, and other features.

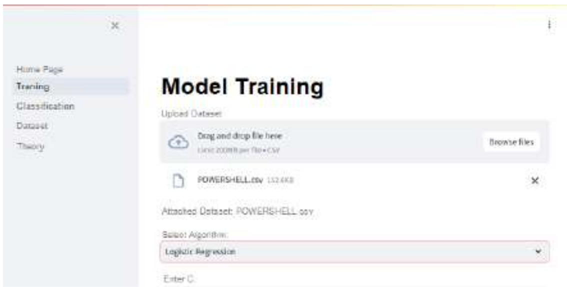

Figure 2: Model Training page .

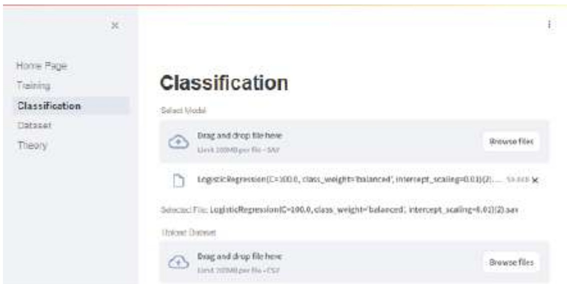

Figure 2: Classification page.

4.3 Implementing a Docker Container

Docker is a tool for creating and managing containers that packages an application along with all its dependencies into a single, portable unit.

A Docker container is an isolated environment that contains the application itself, the necessary libraries, settings, and system components. Thanks to this isolation, the application runs stably and identically on any computer or server, regardless of its configuration.

Docker uses virtualization at the operating system level, which allows you to run multiple containers simultaneously on a single server and personal computer without environmental conflicts. Each Docker container is created from an image that includes all dependencies and application settings.

Docker eliminates the "it works on my machine, but it doesn't work on the server" problem, allowing you to provide a single execution environment from development to the end user.

Advantages of Docker:

- Simplifies development and deployment - no more manual environment setup.
- Guarantees compatibility - the application will work wherever Docker is available.
- Increases security – containers are isolated from each other and from the main system.
- Makes scaling easy - you can quickly start and stop containers as needed.

Docker is a standard in modern DevOps practice and allows you to speed up the processes of testing, delivery and launching applications.

5 EXPERIMENTS AND RESULTS

To determine the optimal amount of data required for training, A/B testing of the trained machine learning algorithms was conducted. The main purpose of using A/B testing in this study was to identify the dependence of the model accuracy on the amount of training data. The essence of testing was that in the course of several experiments, machine learning algorithms were trained on different amounts of data, which made it possible to assess the effect of data volume on their effectiveness. The results of the A/B testing experiments are presented in Table 3.

Table 3: A/B testing results.

Quantity data	Algorithm	Score
300	Naive Bayes	0.98
	Logistic Regression	0.97
	Decision Tree	0.98
	Random Forest	0.96
	XGBoost	0.97
	Support Vector Machines	0.98
	K-Nearest Neighbours	0.97
600	Naive Bayes	0.96
	Logistic Regression	0.98
	Decision Tree	0.98
	Random Forest	0.99
	XGBoost	0.97
	Support Vector Machines	0.99
	K-Nearest Neighbours	0.98
900	Naive Bayes	0.98
	Logistic Regression	0.99
	Decision Tree	0.98
	Random Forest	0.99
	XGBoost	0.98
	Support Vector Machines	0.96
	K-Nearest Neighbours	0.97
1 200	Naive Bayes	0.99
	Logistic Regression	0.99
	Decision Tree	0.98
	Random Forest	0.99
	XGBoost	0.98
	Support Vector Machines	0.96
	K-Nearest Neighbours	0.99
1 500	Naive Bayes	0.98
	Logistic Regression	0.96
	Decision Tree	0.97
	Random Forest	0.99
	XGBoost	0.99
	Support Vector Machines	0.97
	K-Nearest Neighbours	0.97

The implemented algorithms were evaluated in the following way: to check the accuracy of the trained models, validation datasets were prepared that were not involved in training.

During the experiments it was found that the amount of data does not affect the prediction results of the models.

To check the functionality of the web application, functional testing was used, which consisted of checking the functionality of all functions of the web application page by page.

To check the functionality of the Docker container, the correctness of the assembly was checked, and also whether the necessary components were launched inside the Docker container image.
All experiments were conducted on Ubuntu 22.04 CPU: AMD Ryzen 5 3500U with Radeon GPU Vega Mobile Gfx @ 1.40GHz 2.10 GHz and 8 GB of RAM.

6 CONCLUSIONS

Based on the results of this study, the following conclusions can be drawn:

- To solve the problem of team classification PowerShell requires about 300 commands according to the MITRE matrix. Therefore, it can be concluded that for future experiments with CMD and Bash interpreter commands, experiments should start with 300 commands.
- Statistics on trained machine learning algorithms indicate that any of the implemented algorithms can be used for final implementation in product development. And also for future researchers to compare their developed algorithms for comparison.
- cybersecurity specialists, this study is of practical importance in terms of practical implementation of the algorithms under study. The application in the Docker container format and a unique labeled data set can be provided upon request to any author of the article. The data is not made publicly available due to ethical considerations. The authors of the study believe that malicious commands that will be labeled as malicious can cause harm when used by attackers.

This study was carried out within the framework of a grant for the study of problems and prospects for regulating quantum communications in the context of the data economy. It was decided to create a platform on which trusted data sets in the field of cybersecurity will be placed, which will then be used to train machine learning algorithms, including quantum ones. At this stage, the first stage has been completed, which included a prototype with hosted models of classical machine learning and one data set in the field of cybersecurity. At the next stage, it is planned to develop the application in the following directions:

- In terms of the data set, it is planned to expand the data set in the future to include CMD and Bash interpreter commands.
- In terms of machine learning: it is planned to add the ability to work with neural networks, as well as quantum machine learning algorithms.
- In terms of trust in data sets: it is planned to add a module for checking data sets for gaps; outputting statistics on the types and classes that are in the data set.
- In terms of trust in machine learning algorithms: it is planned to add a module for checking machine learning algorithms.

ACKNOWLEDGEMENTS

The research was supported by Russian Science Foundation No 24-18-00950 "Problems and Prospects of Regulating Quantum Communications in the Context of the Data Economy".

REFERENCES

Avdoshin, S. M., Lazarenko, A. V., Chichileva, N. I., Naumov, P. A., Klyucharev, P. G., 2019. Machine Learning Use Cases in Cybersecurity. Trudy ISP RAN/Proc. ISP RAS, vol. 31, issue 5, pp. 191-202

Barrué, G., Quertier, T., 2025. Quantum Machine Learning for Malware Classification. In: Meo, R., Silvestri, F. (eds) Machine Learning and Principles and Practice of Knowledge Discovery in Databases. ECML PKDD 2023. Communications in Computer and Information Science, vol 2137. Springer, Cham.

CountVectorizer https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

Hendler, D., Kels, Sh., Rubin, A., 2018. Detecting Malicious PowerShell Commands using Deep Neural Networks. In Proceedings of the 2018 Asia Conference on Computer and Communications Security (ASIACCS '18). Association for Computing Machinery, New York, NY, USA, 187–197.

Georgiadou, A., Mouzakitis, S., Askounis, D., 2021. Assessing MITER ATT&CK Risk Using a Cyber-Security Culture Framework. Sensors. 21.3267.

Kayode O., 2025. International Journal of Computer Science and Mobile Computing, Vol.14 Issue.1, pg. 20-37

Machine Learning For Cybersecurity: Enhancing Intrusion Detection Systems And Threat Mitigation. (2025). International Journal of Multidisciplinary Engineering In Current Research, 10(2), 69-85

Wazid, M., Das, AK, Chamola, V., Park, Y., 2022. Uniting cyber security and machine learning: Advantages, challenges and future research, ICT Express, Volume 8, Issue 3, Pages 313-321

Prasad, A, Chandra, S., 2023. Machine learning to combat cyberattack: a survey of datasets and challenges. J Model Simul. 20(4):577–88.

Salem, A.H., Azzam, S.M., Emam, O.E., 2024. Advancing cybersecurity: a comprehensive review of AI-driven detection techniques. J Big Data 11, 105.

Scikit -learn https://scikit-learn.org/stable/

Shafin, S.S., Karmakar, G., Mareels, I., 2023. Obfuscated memory malware detection in resource-constrained IoT devices for smart city applications. Sensors.

Sindhgatta, R., Thonse, S., 2005. Functional and Non-functional Requirements Specification for Enterprise Applications. In: Bomarius, F., Komi- Sirviö, S. (eds) Product Focused Software Process Improvement. PROFES 2005. Lecture Notes in Computer Science, vol 3547. Springer, Berlin, Heidelberg.

TfidfVectorizer https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Wu, Ch., Liu, J., Huang, Ch., Li, L., 2025. Cybersecurity entity recognition model for IoT via hierarchical attention mechanism. *International Journal of Modern Physics C.*

Xgboost https://xgboost.readthedocs.io/en/release_3.0.0/