

# Proactive System for Cyber Attack Detection Based on Big Data Analytics Using Machine Learning <sup>†</sup>

Plamen Spahiev \*  and Desislava Ivanova 

Faculty of Applied Mathematics and Informatics, Technical University of Sofia, 1797 Sofia, Bulgaria;  
d\_ivanova@tu-sofia.bg

\* Correspondence: plamen@tu-sofia.bg

<sup>†</sup> Presented at the 15th International Scientific Conference TechSys 2026—Engineering, Technologies and Systems, Plovdiv, Bulgaria, 14–16 May 2026.

## Abstract

This paper presents the conceptual foundation and software architecture of a proactive system for real-time cyber-attack detection. The system is based on a nested approach of building two-level chains of machine learning models. It is composed of two components—one for learning and one for evaluating real-time data. Both components are written in the Python programming language. The data has been processed and manipulated using PySpark. The initial experiments showed a combined increase of up to 25% in accuracy, compared to using non-combined ML models and unbalanced data.

**Keywords:** cyber attack detection; big data analytics; machine learning; proactive cybersecurity; real-time detection; intrusion detection system; data balancing; feature selection; model nesting; tabular data

## 1. Introduction

The rapid development of Artificial Intelligence algorithms, models, tools and systems in recent years has led to their vast spread and entry into practically every level of our day-to-day activities and ordinary life. According to some reports [1], the production and usage of AI models in recent years has grown by up to 5 times. Also, their capabilities have improved and nowadays it is normal for some models to outperform humans in several directions [1].

Unfortunately, alongside the positive benefits that these technologies bring, they have also brought more sophisticated and adaptive means of committing cybercrimes [2]. Conventional cyber-crime countermeasures are in most cases reactive—they react to an already known attack vector. However, AI-initiated attacks can change and adapt in real time [3].

All this imposes the need for a new approach to cybersecurity, with more automated and autonomous cyber-crime prevention tools and more adaptive architectural approaches.

The purpose of this study is to determine the ability of such a system to exist, self-educate and proactively search for malicious patterns.

## 2. Conceptions

The purpose of this research is to propose a fundamental concept of a real-time cyber attack detection system based on an innovative method of nesting several machine learning algorithms together and thus increasing their combined accuracy. For the experiment,



Academic Editors: Nikola G. Shakev,  
Valyo N. Nikolov, Nikolay  
R. Kakanakov and Sevil  
A. Ahmed-Shieva

Published: 17 July 2026

**Copyright:** © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

the system uses a publicly available data set, but the concept is generic and can therefore be applied (with specific data preprocessing) to any variety of input data. Since various different elements have to be taken into consideration, a decision has been made to order them in a tabular view and use this format as a source data format for the entire system.

### 3. Data Set

To start the experiment, a suitable data set and adequate analytics engine were needed. When speaking of data sets for cyber-security analysis, the first one to be considered is the famous KDD CUP 99. For data analysis and machine learning, Apache Spark and more precisely the Python implementation—PySpark—were chosen.

#### 3.1. KDD CUP 99

The KDD CUP 99 data set [4] was created as the result of a knowledge discovery and data mining competition in which information about LAN attacks was gathered for several weeks in an isolated environment [5]. The raw data was then additionally processed, labeled and compiled in what is currently known as the KDD CUP 99 data set. The data set, being one of the few freely distributed ones, has been used in the vast majority of studies and analyses aimed at building or improving intrusion detection systems.

#### 3.2. ISCX NSL-KDD Dataset 2009

The original KDD CUP 99 data set has served for more than a decade as a basis for analysis and research. However, there were several problems at a design level, which caused biasing of the machine learning algorithms and thus produced unreliable results. In the revised version of the original data set—the ISCX NSL-KDD [6]—problems such as redundant records and level of difficulty have been solved [7] and a more reliable data set has been produced.

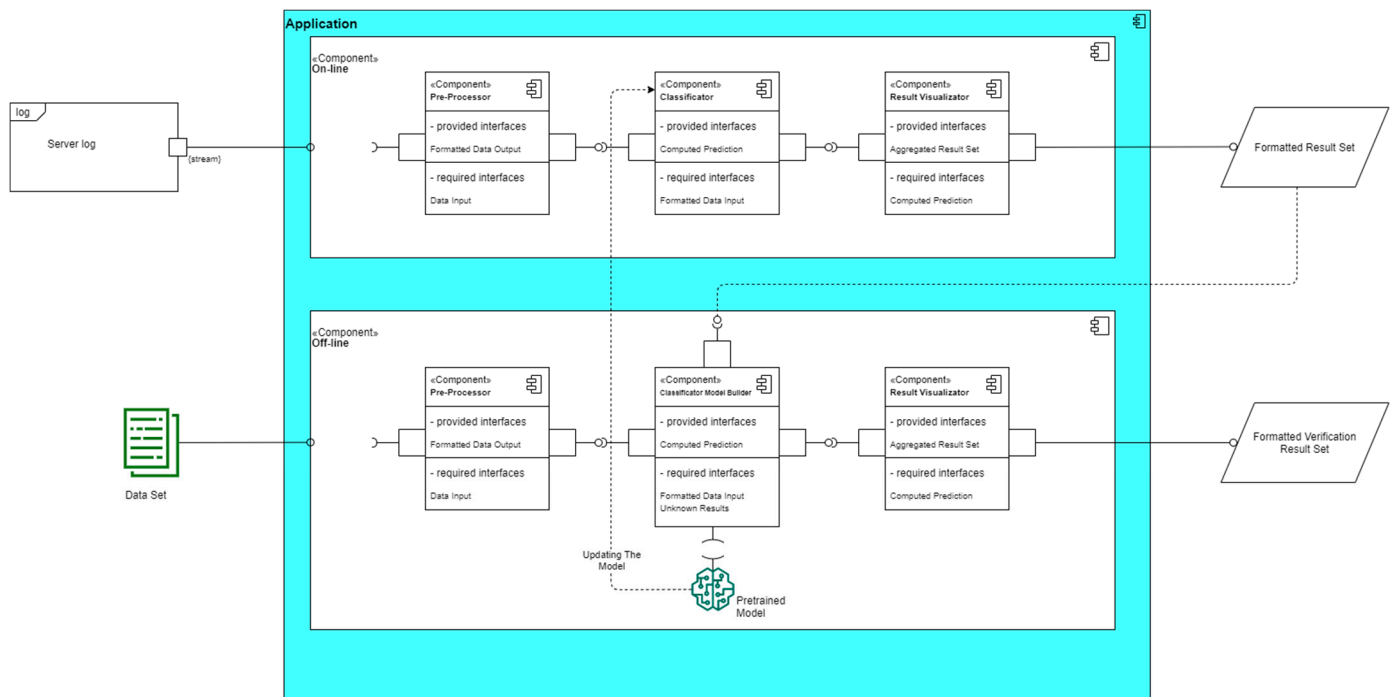
### 4. Technologies

#### *Apache Spark and PySpark*

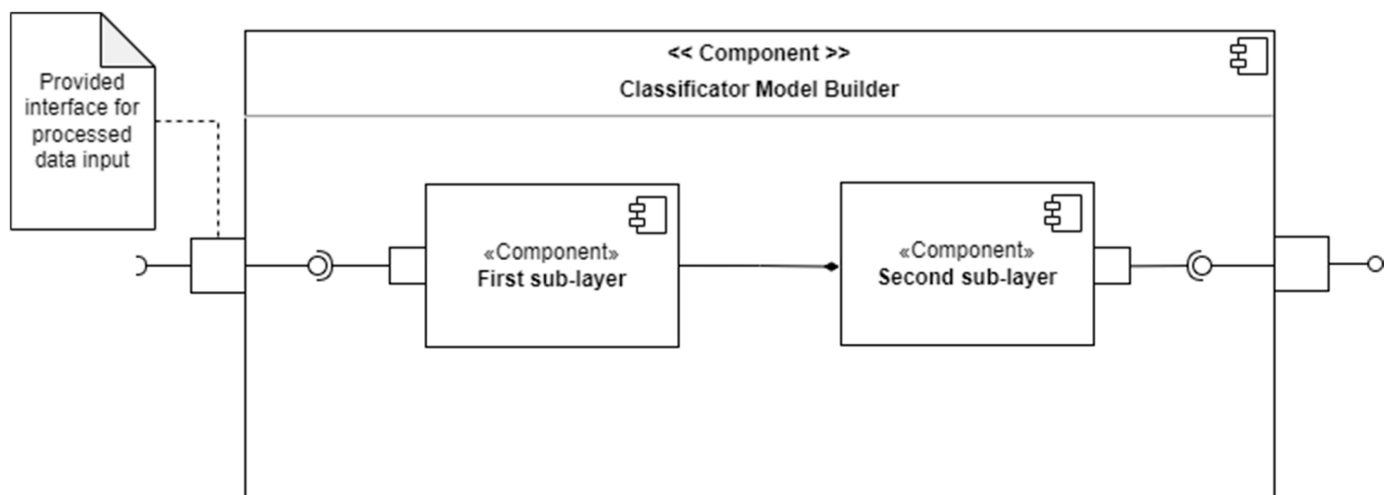
Apache Spark is an open-source multi-language analytics software for large-scale data processing [8]. It is a particularly good tool for machine learning on single node machines [8]. PySpark version 3.4.0, on the other hand, is an interface for Apache Spark version 3.3.3 in Python version 3.11 [9]. `pyspark.ml` is a data-frame machine learning API providing access to different precompiled machine learning algorithms and different methods for the evaluation of their performance.

### 5. Architecture

The proposed concept consists of a two-component/three-layered system in which the first component serves to train and update the models, used later in the second component for analysis of the real data. The first component also has three layers—one aiming to balance and format the data accordingly, the second training the specific machine learning models, and the third used for visualization of the results. We are also using an innovative model nesting: the second layer also can be split into two sub-layers, with the first one training the basic models and the second one training the second-layer nested model (Figures 1 and 2).



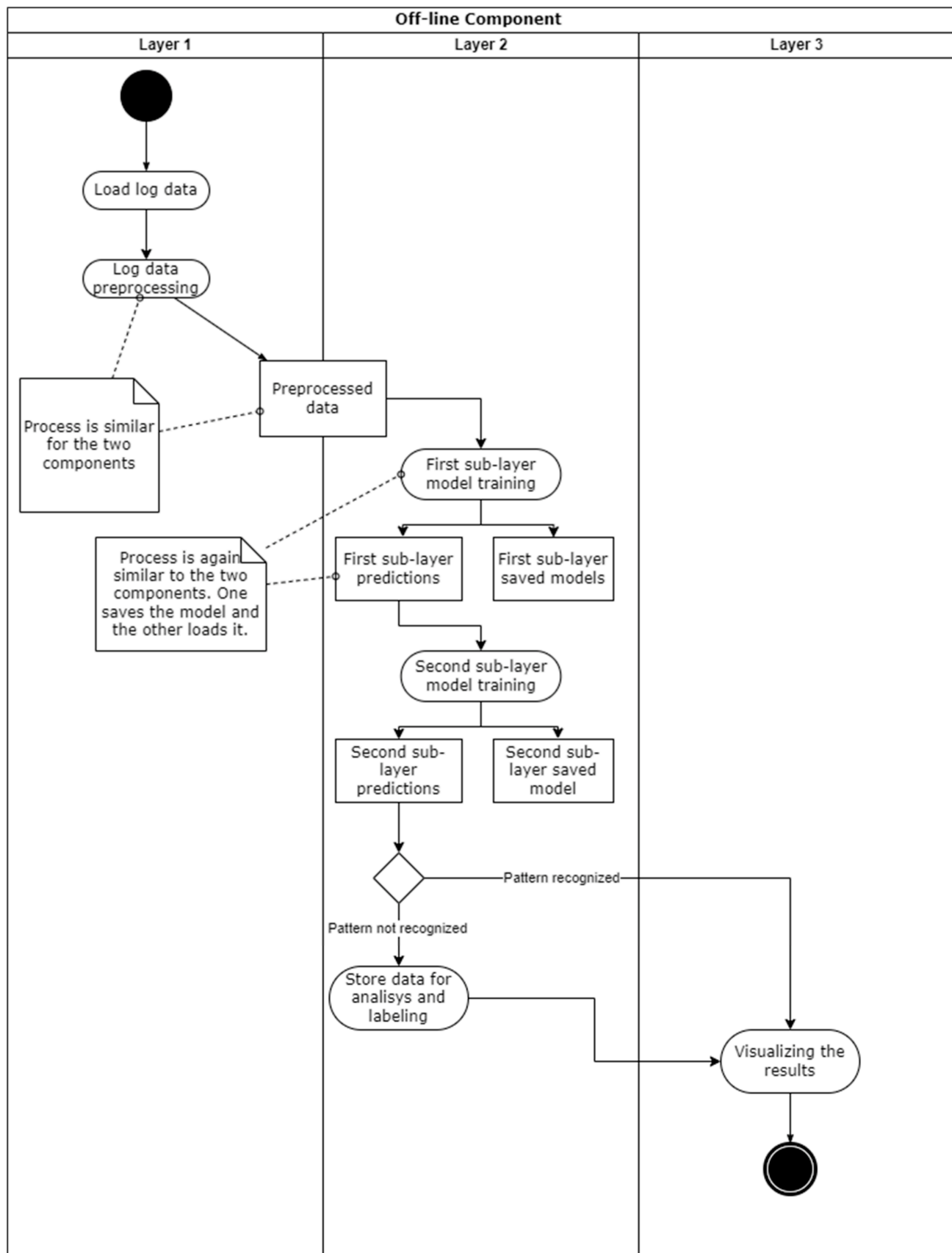
**Figure 1.** High-level component diagram for the entire system.



**Figure 2.** Component view for the model builder with clearly visible layered structure.

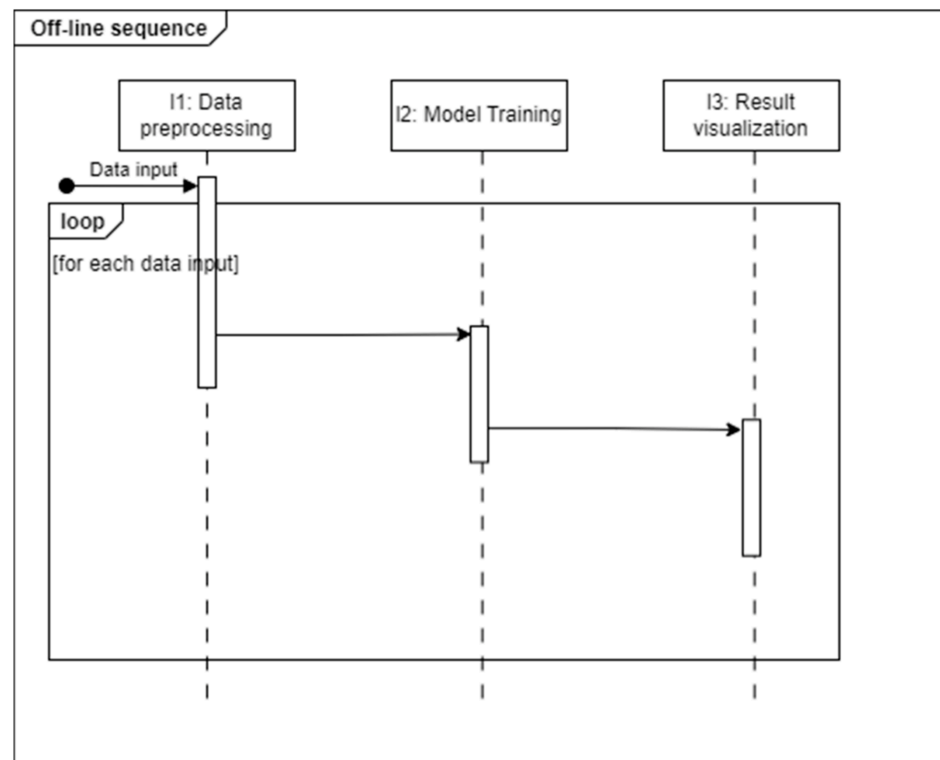
The second component is also three-layered: the first layer preprocesses the input data, the second one uses the models pre-trained by the first layer to analyze the input data, and the third one visualizes the results. The second layer also has the task of feeding back to the first component all data that is not recognized, so that it can be later labeled and used in additional model training.

The process can be visualized in Figure 3.



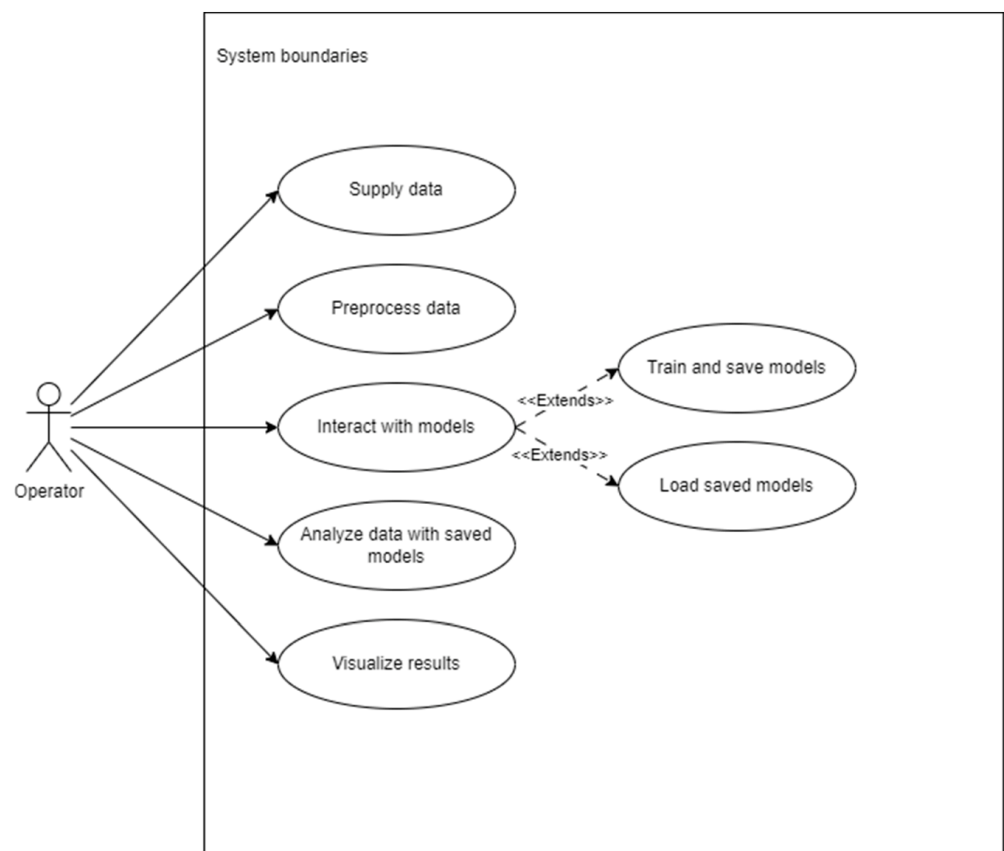
**Figure 3.** Off-line component activity diagram.

It can also be seen in Figure 4.



**Figure 4.** Sequence diagram for the offline component.

The general use case can be visualized in Figure 5.



**Figure 5.** General use case diagram.

## 6. Essence of the Experiment

### 6.1. Data

As stated earlier, the ISCX NSL-KDD [6] data set was used for the experiment. The data set contains predefined training and test data, with the test data showing considerable inconsistencies compared to the train data. Although this makes the entire data set very unbalanced, it represents a real-world scenario in which a person can encounter an unrecognized attack. Figure 6 briefly represents the differences between the data in the two data sets (aggregated).

| +-----+-----+       |  | +-----+-----+       |  |
|---------------------|--|---------------------|--|
| attack count        |  | attack count        |  |
| +-----+-----+       |  | +-----+-----+       |  |
| normal  67343       |  | normal  9711        |  |
| neptune  41214      |  | neptune  4657       |  |
| satan  3633         |  | guess_passwd  1231  |  |
| ipsweep  3599       |  | mscan  996          |  |
| portsweep  2931     |  | warezmaster  944    |  |
| smurf  2646         |  | apache2  737        |  |
| nmap  1493          |  | satan  735          |  |
| back  956           |  | processtable  685   |  |
| teardrop  892       |  | smurf  665          |  |
| warezclient  890    |  | back  359           |  |
| pod  201            |  | snmpguess  331      |  |
| guess_passwd  53    |  | saint  319          |  |
| buffer_overflow  30 |  | mailbomb  293       |  |
| warezmaster  20     |  | snmpgetattack  178  |  |
| land  18            |  | portsweep  157      |  |
| imap  11            |  | ipsweep  141        |  |
| rootkit  10         |  | httptunnel  133     |  |
| loadmodule  9       |  | nmap  73            |  |
| ftp_write  8        |  | pod  41             |  |
| multihop  7         |  | buffer_overflow  20 |  |
| phf  4              |  | multihop  18        |  |
| perl  3             |  | named  17           |  |
| spy  2              |  | ps  15              |  |
| +-----+-----+       |  | sendmail  14        |  |
|                     |  | rootkit  13         |  |
|                     |  | xterm  13           |  |
|                     |  | teardrop  12        |  |
|                     |  | xlock  9            |  |
|                     |  | land  7             |  |
|                     |  | xsnoop  4           |  |
|                     |  | ftp_write  3        |  |
|                     |  | sqlattack  2        |  |
|                     |  | perl  2             |  |
|                     |  | udpstorm  2         |  |
|                     |  | worm  2             |  |
|                     |  | loadmodule  2       |  |
|                     |  | phf  2              |  |
|                     |  | imap  1             |  |
|                     |  | +-----+-----+       |  |

(a) Training data

(b) Test data

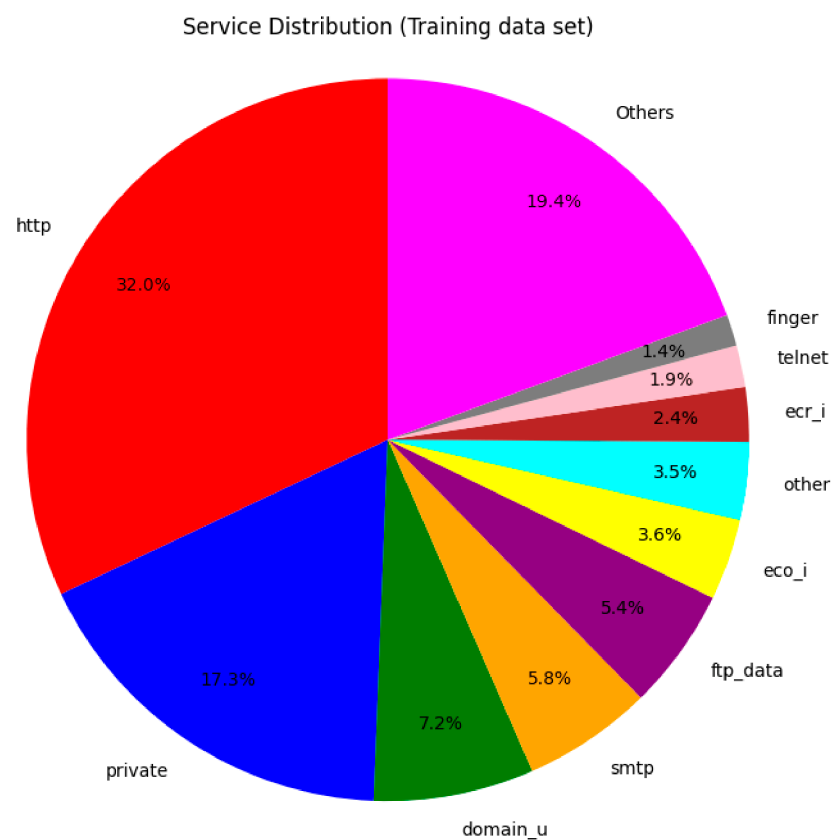
**Figure 6.** Differences between training and test data included in the ISCX NSL-KDD data set.

The Figure 7 represents the data types found in the test data, but not present in the training data.

| attack        | count |
|---------------|-------|
| mscan         | 996   |
| apache2       | 737   |
| processtable  | 685   |
| snmpguess     | 331   |
| saint         | 319   |
| mailbomb      | 293   |
| snmpgetattack | 178   |
| httptunnel    | 133   |
| named         | 17    |
| ps            | 15    |
| sendmail      | 14    |
| xterm         | 13    |
| xlock         | 9     |
| xsnoop        | 4     |
| sqlattack     | 2     |
| udpstorm      | 2     |
| worm          | 2     |

**Figure 7.** Attacks present in the test data but not in the training data.

The above statement can also be visualized in the Figures 8–11 (grouping based on service and attack distribution).



**Figure 8.** Service distribution in the training data set.

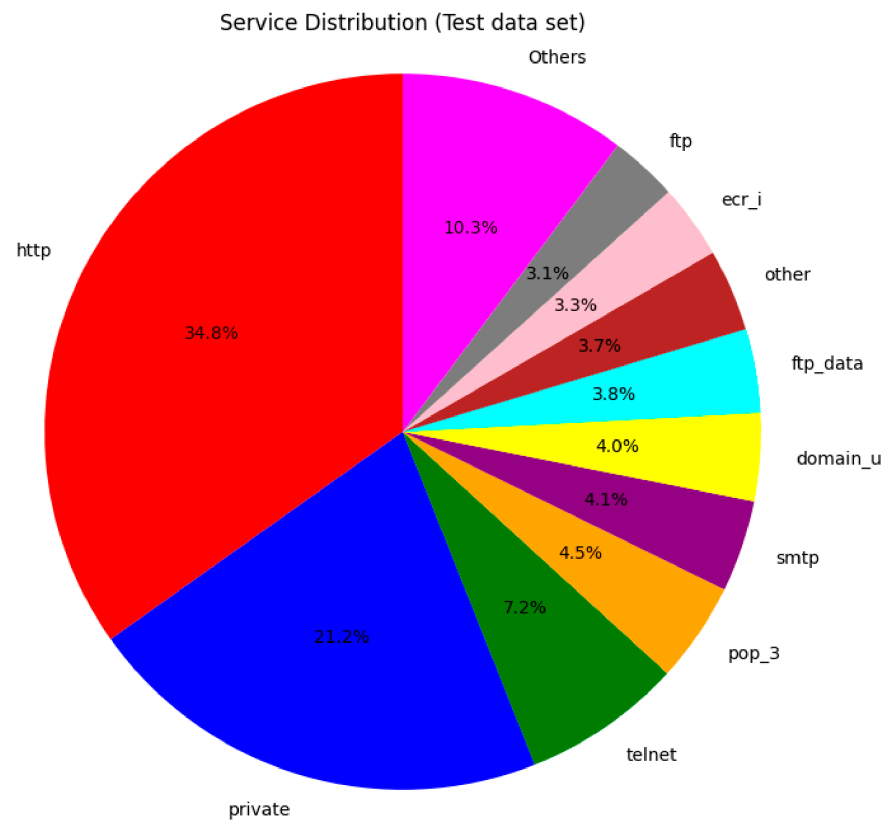


Figure 9. Service distribution in the test data set.

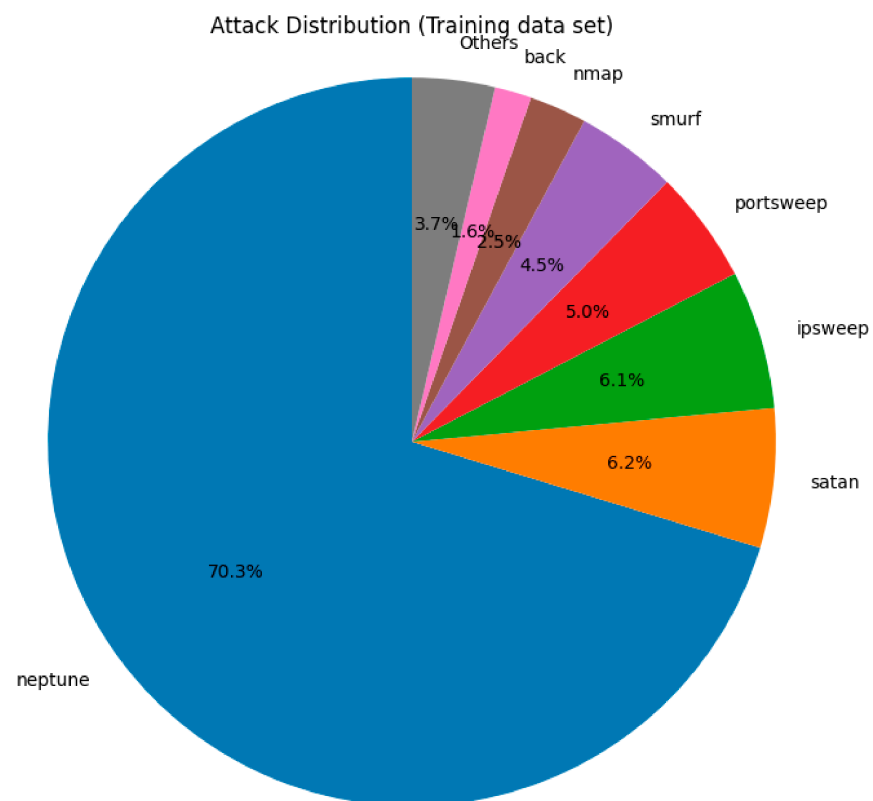
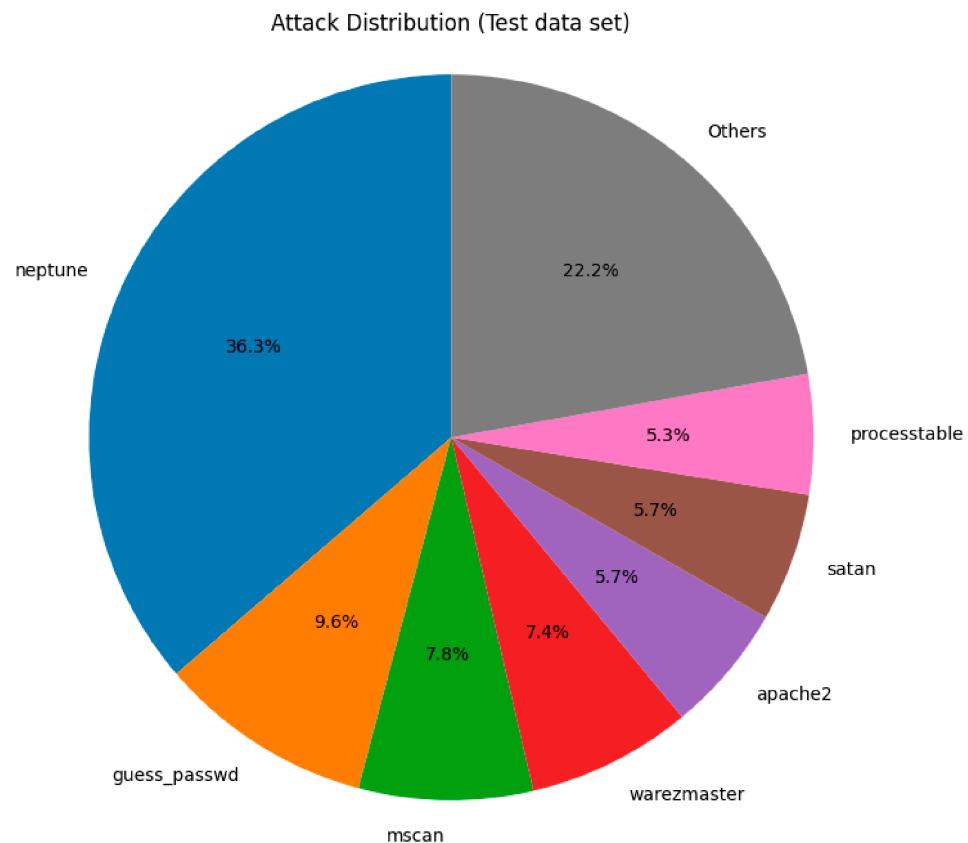


Figure 10. Attack distribution in the training data set.



**Figure 11.** Attack distribution in the test data set.

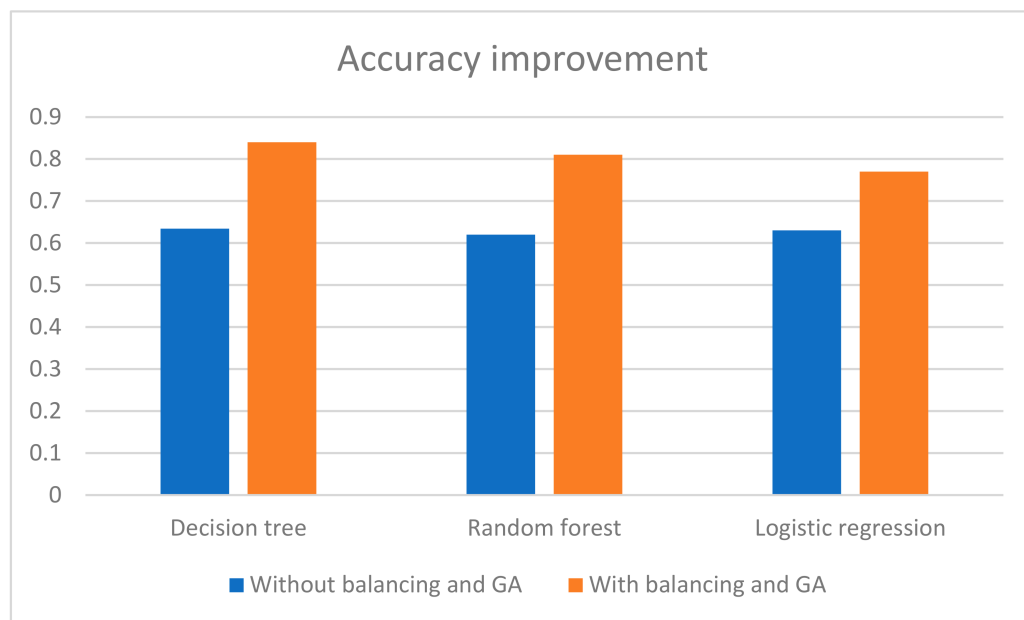
### 6.2. Steps of the Experiment

1. As a first step, the unbalanced training data must be balanced so that the predictions stay unbiased [10,11]. For this specific experiment, the data was oversampled then under sampled (for fewer computing errors and to prevent model overfitting) using the SMOTE library, v. 0.7.1 [12].
2. As a second step, the most important features were selected using the Chi-squared test [9].
3. As a third step, the first sub-layer models were trained using the balanced training data from the previous steps and the test data provided by the data set. The models performed well on tabular data [13–16] and were chosen after experimentation and result comparison. Hyperparameters for the actual training were carefully selected using the genetic algorithm (GA) provided by the DEAP framework [17]. The models chosen were:
  - a. Decision tree classifier;
  - b. Random forest classifier;
  - c. Logistic regression.
4. As a fourth step, the trained models were saved for future use in the second component.
5. As a fifth step, the prediction data from the three models was aggregated into a data set by itself.
6. As a sixth step, the aggregated data was used to train a second layer model, after hyperparameter tuning using GA from DEAP, v. 1.3.3. The algorithm chosen for that step was a decision tree classifier, which as stated performs well with tabular data.
7. As a seventh step, the data was analyzed and conclusions were made.

### 6.3. Results of the Experiment

By applying data balancing and GA, the accuracy changed as follows (Figure 12):

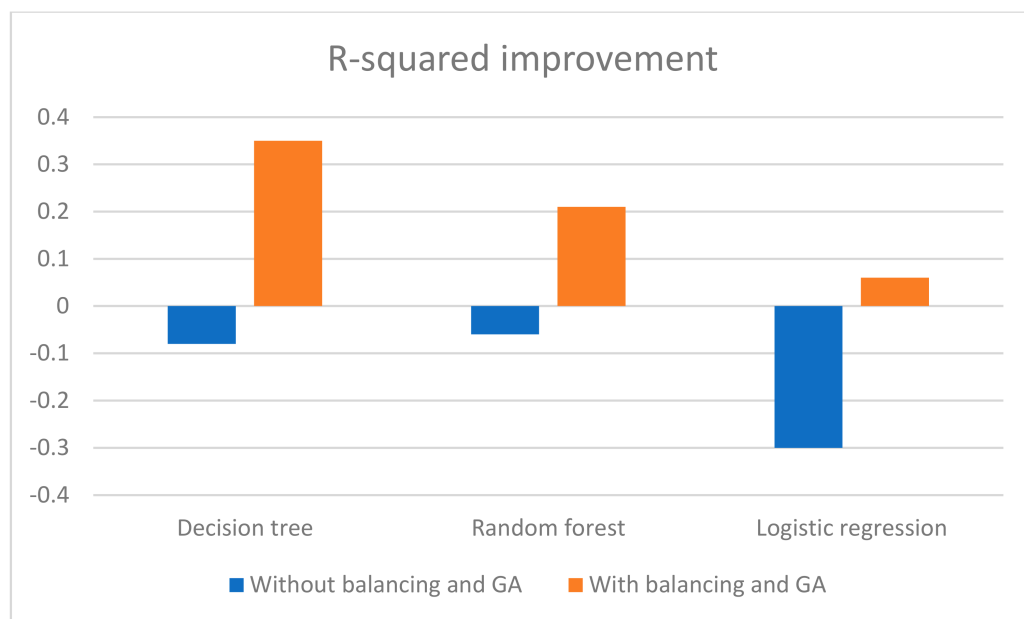
1. For the decision tree classifier, from 0.634 to 0.84;
2. For the random forest classifier, from 0.62 to 0.81;
3. For logistic regression, from 0.63 to 0.77.



**Figure 12.** Accuracy improvement during the experiment.

The R-squared changed as follows (Figure 13):

1. For the decision tree classifier, from  $-0.08$  to  $0.35$ ;
2. For the random forest classifier, from  $-0.06$  to  $0.21$ ;
3. For logistic regression, from  $-0.3$  to  $0.06$ .



**Figure 13.** R-squared improvement during the experiment.

The second layer by itself showed an accuracy of 0.88 and R-squared of 0.5, which is higher compared to the results shown on the first level.

## 7. Conclusions

- Based on the experiment, the accuracy of the algorithms for the first sub-layer improved by an average of 18%.
- Bearing in mind that we improved the accuracy in the second sub-layer, we can conclude that the architectural approach leads to better accuracy and that the system is built in such a way that will enable more accurate predictions.

**Author Contributions:** Conceptualization, P.S. and D.I.; methodology, P.S.; software, P.S.; validation, P.S. and D.I.; formal analysis, P.S.; investigation, P.S.; data curation, P.S.; writing—original draft preparation, P.S.; writing—review and editing, P.S. and D.I.; visualization, P.S.; supervision, D.I. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** All data created during the experiment is published in the current article.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. Artificial Intelligence Index Report 2024, Stanford University, Stanford California. Available online: <https://hai.stanford.edu/ai-index/2024-ai-index-report> (accessed on 1 March 2024).
2. DEAP Project. DEAP Documentation. Available online: <https://deap.readthedocs.io/en/master/> (accessed on 2 March 2024).
3. Nagar, G.; Manoharan, A. Understanding the Threat Landscape: A Comprehensive Analysis of Cyber-Security Risks in 2024. *Int. Res. J. Mod. Eng. Technol. Sci.* **2024**, *6*, 5706–5713.
4. KDD Cup 1999 Data. Available online: <https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html> (accessed on 5 March 2024).
5. Available online: <https://kdd.ics.uci.edu/databases/kddcup99/task.html> (accessed on 5 March 2024).
6. ISCX NSL-KDD Dataset 2009. Available online: <https://www.unb.ca/cic/datasets/nsl.html> (accessed on 5 March 2024).
7. Tavallae, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, Ottawa, ON, Canada, 8–10 July 2009. [CrossRef]
8. Spark. Available online: <https://spark.apache.org/> (accessed on 2 March 2024).
9. Spark Overview. Available online: <https://archive.apache.org/dist/spark/docs/3.3.1/> (accessed on 2 March 2024).
10. Abdelkhalek, A.; Mashaly, M. Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning. *J. Supercomput.* **2023**, *79*, 10611–10644. [CrossRef]
11. Mijalkovic, J.; Spognardi, A. Reducing the False Negative Rate in Deep Learning Based Network Intrusion Detection Systems. *Algorithms* **2022**, *15*, 258. [CrossRef]
12. SMOTE. Available online: [https://imbalanced-learn.org/stable/references/generated/imblearn.over\\_sampling.SMOTE.html](https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html) (accessed on 5 March 2024).
13. ZhangZhou, J.; He, C.; Sun, J.; Zhao, J.; Lyu, Y.; Wang, S.; Zhao, W.; Li, A.; Ji, X.; Agarwal, A. Geochemistry  $\pi$ : Automated machine learning Python framework for tabular data. *Geochem. Geophys. Geosyst.* **2024**, *25*, e2023GC011324. [CrossRef]
14. Shwartz-Ziv, R.; Amitai, A. Tabular data: Deep learning is not all you need. *Inf. Fusion* **2022**, *81*, 84–90. [CrossRef]
15. Gorishniy, Y.; Rubachev, I.; Khrulkov, V.; Babenko, A. Revisiting deep learning models for tabular data. In Proceedings of the 35th International Conference on Neural Information Processing Systems (NIPS '21), Online, 6–14 December 2021.

16. Grinsztajn, L.; Oyallon, E.; Varoquaux, G. Why do tree-based models still outperform deep learning on typical tabular data? *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 507–520. [[CrossRef](#)]
17. Katiyar, N.; Tripathi, S.; Kumar, P.; Verma, S.; Sahu, A.K.; Saxena, S. AI and Cyber-Security: Enhancing threat detection and response with machine learning. *Educ. Adm. Theory Pract.* **2024**, *30*, 6273–6282. [[CrossRef](#)]

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.