UDC 004.855.5
https://doi.org/10.47533/2023.1606-146X.31

***G. ALPYSBAY*, A. BEDELBAYEV, A. USMANOVA, ZH. DUISENBEKKYZY***

*Al-Farabi Kazakh National University, Almaty, Kazakhstan*
*e-mail: gulbanu.alpysbay@gmail.com, agyn08@yandex.ru, aseka_usmanova@mail.ru, jansaya_sugirbaeva@mail.ru*

# COMPARATIVE ANALYSIS OF MACHINE LEARNING ALGORITHMS USED IN MALWARE ANALYSIS

*Despite the achievement of the field of cyber security in the modern world of science and the continuous development of its methods, malware is still one of the biggest threats to information security. Malware is evolving every day, and its types and behaviors are increasing day by day. And the importance of using modern, sophisticated technologies in identifying and combating such complex and diverse malicious programs is increasing. In this regard, it is possible to mention the advantages of using intelligent systems in the field of information security. In this article, we will analyze PE (Portable Executable) files on the Windows operating system, that is, the processes running on the computer and analyzing the malicious programs using machine learning algorithms. At the same time, we will focus on the operation of different machine learning algorithms and show which method is most effective to use for our example.*

*In this article, we will have the following tasks:*

*1. Providing information about malicious software. Definition of PE files, its structure and nature.*

*2. Preparation of data for practical work (collection of files with clean and malicious code). An overview of methods for separating files into clean and malicious files.*

*3. Sorting the signs necessary for training according to pre-prepared files, that is, getting only those signs that allow to achieve the most accurate result during training.*

*4. Implementation of several machine learning algorithms and selection of the most effective among them.*

***Keywords**: information security, malicious software, PE files, malware analysis, machine learning, feature.*

**Introduction.** Although the rapid development of information technologies is an achievement of science, it seems that it is becoming more and more difficult to ensure the safety of various information in it. The number of attacks aimed at gaining access to hidden, confidential information is definitely increasing, and each of them is directed against a different target. Many attacks target individuals or organizations to obtain valuable information. But sometimes they are associated with cybercrime or criminal gangs. Malware analysis requires knowledge, skills, and tools to detect, investigate, and defend against such attacks. Malware analysis is an area of research into the functionality, purpose, origin, and potential impact of malware. This task is usually done mostly by hand and involves a lot of labor. To solve it, analysts with an expert level of knowledge about the internal organization of software and reverse engineering are required. Data science and machine learning hold promise for automating some of the steps in malware analysis, but their techniques remain primarily concerned with extracting the most important features from data. This is a very difficult task, which also requires experts-practitioners with a special set of knowledge and skills.

* E-mail корреспондирующего автора: gulbanu.alpysbay@gmail.com

**Literature review and problem statement.** Malware is code that performs malicious actions; it can take the form of an executable file, script, code, or any other piece of software. Attackers use malware to steal sensitive information in order to spy on an infected system or take control of a system. Malicious software can be embedded in a variety of binary file formats that work in completely different ways. For example, PE files in Windows OS (Portable Executables, with extensions .exe, .dll, .efi, etc.), ELF files in Unix systems (Executable and Linkable Format) and APK files in OS Android (Android Package Kit format with .apk extension, etc.) have completely different internal file structures and require different execution contexts. It is quite natural that for the analysis of each class of executable files, special additional requirements are also completely different. Also, be aware that malware can also exist in forms other than individual binary executables. There are widespread malicious components that infiltrate document files, such as those with extensions .doc, .pdf, and .rtf, and use macros and dynamically executed elements in the document structure to perform malicious actions. Malware can also take the form of extensions and plug-ins for common software platforms, such as web browsers and complex web environments. Figure 1 shows the types of possible threats in the field of information security, including the types of malware and their relationship with other threats [1].

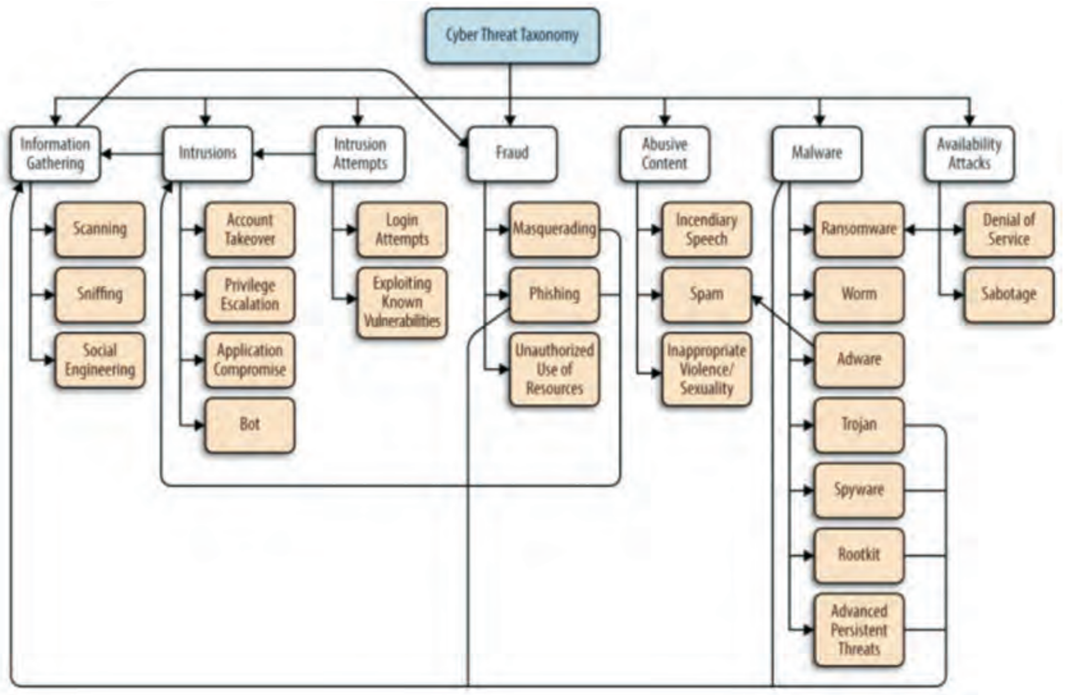

***Figure 1 –*** Types of cyber threats

**Method & Materials.** Virus-creating programmers have learned to successfully bypass signature searches by hiding the virus's body. Polymorphic and metamorphic malware were able to change their appearance. All this prompted antivirus companies to develop alternative methods of information security protection. According to 2 main directions of research:

1. Static analysis (analysis of the structure of the binary file, its attributes, logical structures, execution flow and data).
2. Dynamic analysis (analysis of the actions of the program during execution).

Each of the methods has its own advantages and disadvantages. It is best to use both of these methods to better detect malware. Each of these methods may fail to detect the presence of a virus in a file. In such cases, planned processing may corrupt the file and lead to loss of information.

Dynamic parsing allows you to bypass binary obfuscation. For example, virus authors make extensive use of packaging systems, code and data encryption, and manipulation of function and control flow. But the same techniques are used by developers to protect intellectual property, making applications harder to reverse engineer. This method isolates several basic operations, such as deleting a file, writing to a file, communicating with the network, opening a port for listening, sending mails, etc. This profile of the file, its activity is studied by an expert or machine learning methods to draw conclusions about the maliciousness of the sample.

However, dynamic analysis is possible only during the execution of the studied code, which makes the operating system vulnerable, and some viruses can determine the execution environment, and behave differently in test and production environments. Thus, the requirement to create environments that are as similar as possible seems to be another problem that needs to be solved.

Static analysis can complement dynamic analysis by providing information about binary file attributes. The static method analyzes the program before execution, extracts attributes from the binary file, calculates statistics, and based on this information makes a judgment about the risk of the file being examined. This approach is safe - the verdict is issued before the file is executed, but it does not work well on files with distorted, wrapped partitions. Also, as the file size increases, the time required for analysis increases. Additionally, to develop a high-quality static analyzer, you need to understand how the binary loader works. Viruses can use some fields of the binary file for their own purposes. For example, in the form of data storage or malicious code execution address [2, 3].

If we focus on the structure of the use of machine learning algorithms in the detection of malicious programs, it can be divided into three main stages and they are (figure 2):

1. Objective analysis, that is, describes the main purpose of the analysis, for example, to determine the malicious program, its type, to which group it belongs.
2. Describes the methods for analyzing features and the features needed for training, for example, the features we need can be obtained by dynamic analysis, static or hybrid analysis.
3. To determine what type of machine learning algorithm is used when creating practical work [4].

So, in the practical part of this article, we will analyze PE files. First of all, let's look at the structure of this type of file (Figure 3). Looking at the details, the PE file structure consists of these components:

– MS-DOS Stub. The PE file starts with this keyword, and this attribute is a dummy.

– PE Signature - 4-byte signature indicating that the file is a PE.

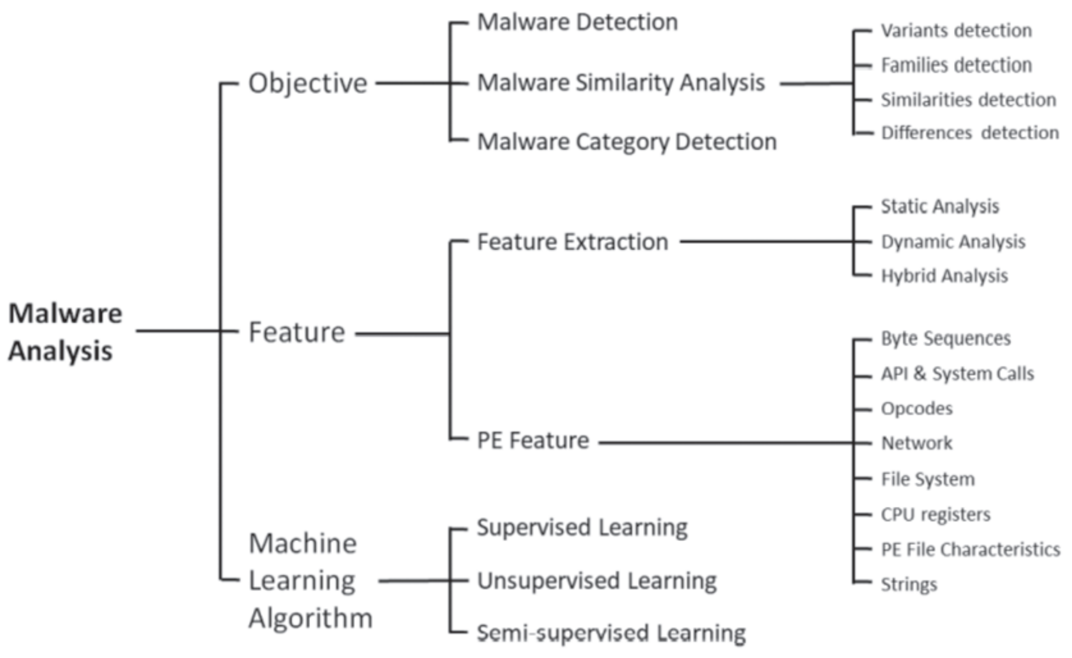

***Figure 2 –*** Sructure of machine learning techniques for malware analysis

– COFF File Header - general information about the file (number of sections, flags, attributes, target machine type).

– Optional Header - detailed information about the file (address space, total size of code sections)

– Section Table - sections, their names, size, address, virtual address.

– Overlay - extract file, end of file [5, 6].

– Among the various attributes and parameters mentioned above, it will be necessary to sort out the signs necessary for machine learning to determine whether the file contains malicious code or not.

– In order to train a classifier, we need to have data that is labeled, i.e. it has been previously determined whether it is harmful or not. For our example, we need to get a set of clean and malicious PE files. There is no problem in finding clean files, you can continue to use the normal executable files in the Windows operating system. Malicious files can also be obtained from open sources these days, for example the following types of data:

– VirusTotal service, the database contains more than a million files with malicious code in PE format [7].

– MalwareTrafficAnalysis.net website contains comprehensive, fully researched 600 samples of malicious code [8].

– VirusShare.com site 30 million provides an integrated database of malicious code patterns [9];

– VX Heaven team collected 270,000 malicious code samples for scientific use [10];

– In 2015, Kaggle and Microsoft managed to collect more than 10,000 malicious code samples into a single database by organizing the Malware Classification Challenge project [11].

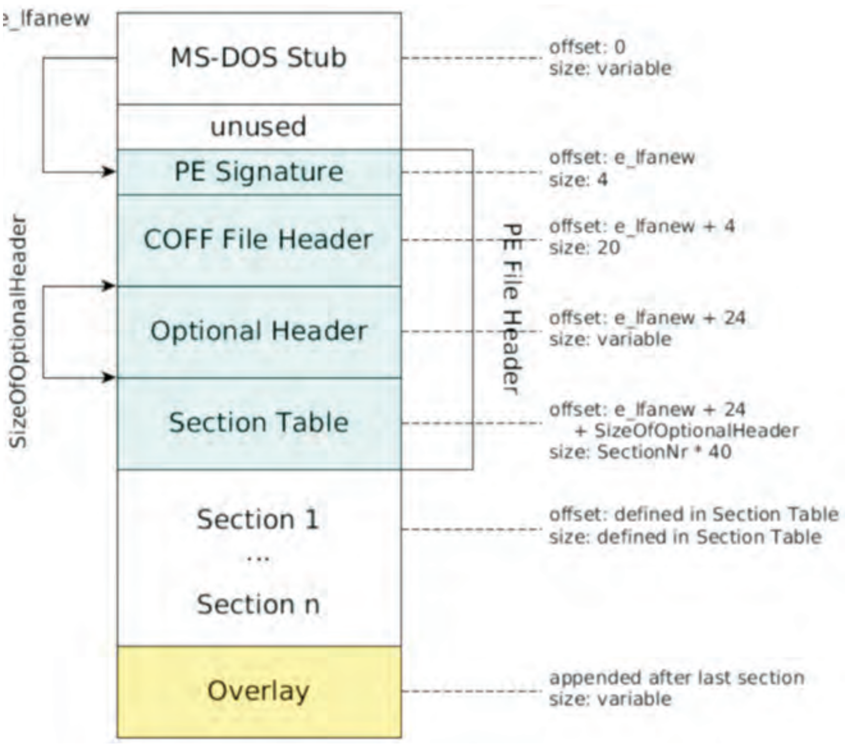

***Figure 3 –*** Structure of PE file

Next, we determined how and what type of attributes we would use for machine learning from the PE file. We extract the maximum number of features from the file, and then select from them using a greedy algorithm. That is, at each step we will highlight the attribute that can maximize the accuracy of the model/classifier. We repeat this process until we get the highest classifier quality result. We make selections on the training set using 3-fold cross-validation. We use a random forest algorithm to select features.

In addition to the attributes, we calculate the entropy of the partitions. Entropy is calculated as follows (Shannon entropy) [12]:

$$H = -\sum_{i=1}^{n} p_i \log p_i \tag{1}$$

We take the base of the logarithm as 256, which is equal to the number of possible byte values. Thus, the entropy H will take values from 0 to 1.

High entropy can indicate compression of sections in a file, packages to hide malicious code, and the use of encryptors.

Attributes collected from files:

1. Numeric attribute values taken directly from PE file fields;
2. API participation flags, sections and their descriptions {0, 1}
3. Entropy of sections [13].

Using the methods and algorithms mentioned above, we got a set of exactly 50 features (Table 1) that we need, and with their help we can detect malware.

We will use these sets of extracted features to classify files into malicious or benign types. For this, we use several types of supervised machine learning algorithms.

Supervised learning is a branch of machine learning that combines algorithms and techniques for building models based on a set of pre-given examples containing input-output pairs.

In order for an algorithm to belong to the type of supervised learning, it must work with examples that contain not only a vector of independent variables (attributes, features), but also a value that comes out after model training (such a value is called a target value). The difference between the target and actual results of the model is reduced to a minimum during the learning process and is called the learning error (residual, residual), which acts as a "teacher". The output error value is then used to calculate model parameter corrections at each training iteration.

***Table 1*** – Features

| No | Feature name | № | Feature name |
|---|---|---|---|
| 1 | sha256 | 26 | minor_operating_system_version name |
| 2 | appeared | 27 | major_subsystem_version |
| 3 | label | 28 | minor_subsystem_version |
| 4 | file_size | 29 | sizeof_code |
| 5 | vsize | 30 | sizeof_headers |
| 6 | has_debug | 31 | sizeof_heap_commit |
| 7 | exports | 32 | imports |
| 8 | imports | 33 | exports |
| 9 | has_relocations | 34 | entry |
| 10 | has_resources | 35 | name_of_section |
| 11 | has_signature | 36 | size_of_section |
| 12 | has_tls | 37 | vsize_of_section |
| 13 | symbols | 38 | entropy |
| 14 | header | 39 | props |
| 15 | timestamp | 40 | histogram |
| 16 | machine | 41 | byte_entropy |
| 17 | characteristics | 42 | strings |
| 18 | subsystem | 43 | num_strings |
| 19 | dll_characteristics | 44 | avlength |
| 20 | magic | 45 | printabledist |
| 21 | major_image_version | 46 | printables |
| 22 | minor_image_version | 47 | paths |
| 23 | major_linker_version | 48 | urls |
| 24 | minor_linker_version | 49 | registry |
| 25 | major_operating_system_version | 50 | MZ |

Currently, many supervised learning algorithms have been developed, each of which has advantages and disadvantages depending on the setting of different problems. There is no single most efficient algorithm for solving all problems, each type of task has its own efficient algorithm.

– Supervised learning algorithms for classification tasks:
– decision tree;
– support vector machines;
– native Bayes classifier;
– linear discriminant analysis;
– k-nearest neighbor method;
– Supervised learning algorithms for regression tasks:
– linear regression;
– logistic regression;
– neural networks [14].

**Results.** The results of the experimental work, that is, the classification results, can be seen in the table 2.

Let's focus on the meaning of the terms whose values are given in the table.

Before describing the indicators obtained as a result of the machine learning algorithm, let us give information about the classification errors of these indicators, the confusion matrix.

We have two classes and an algorithm that predicts that each object belongs to one of the classes, where the classification error matrix is:

***Table 2*** – Confusion matrix

| | $y = 1$ | $y = 2$ |
|---|---|---|
| $\hat{y} = 1$ | True Positive (TP) | False Positive (FP) |
| $\hat{y} = 0$ | False Negative (FN) | True Negative (TN) |

Here $\hat{y}$ is the algorithm's response on the object, $y$ is the true class label on that object.

Classification errors are of two types: False Negative (FN) and False Positive (FP).

***Table 3*** – Classification report

| № | Algorithm | Accuracy | | Precision | Recall | F1-score | Support |
|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
| 1 | Logistic Regression | 0.83567 | 0 | 0.72 | 0.78 | 0.83 | 9778 |
| | | | 1 | 0.84 | 0.92 | 0.81 | 11968 |
| | | | avg/total | 0.78 | 0.85 | 0.82 | 21746 |

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|
| 2 | Decision Tree | 0.91258 | 0 | 0.90 | 0.89 | 0.94 | 9778 |
| | | | 1 | 0.91 | 0.94 | 0.87 | 11968 |
| | | | avg/total | 0.91 | 0.92 | 0.91 | 21746 |
| 3 | Forests of Decision Trees | 0.76874 | 0 | 0.78 | 0.76 | 0.83 | 9778 |
| | | | 1 | 0.84 | 0.67 | 0.71 | 11968 |
| | | | avg/total | 0.81 | 0.72 | 0.77 | 21746 |
| 4 | Support Vector Machine | 0.79621 | 0 | 0.74 | 0.78 | 0.76 | 9778 |
| | | | 1 | 0.65 | 0.73 | 0.71 | 11968 |
| | | | avg/total | 0.70 | 0.76 | 0.74 | 21746 |
| 5 | Naive Bayes Classifier | 0.69862 | 0 | 0.61 | 0.59 | 0.67 | 9778 |
| | | | 1 | 0.73 | 0.54 | 0.65 | 11968 |
| | | | avg/total | 0.67 | 0.57 | 0.66 | 21746 |
| 6 | k-nearest neighbors | 0.86668 | 0 | 0.90 | 0.78 | 0.83 | 9778 |
| | | | 1 | 0.84 | 0.94 | 0.89 | 11968 |
| | | | avg/total | 0.87 | 0.87 | 0.86 | 21746 |

***Accuracy*** is our most common evaluation metric and is easy to understand, i.e. the number of samples to be matched divided by the number of all samples.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

In general, the higher the accuracy, the better the classifier. The degree of accuracy is indeed a very good and intuitive measure of the estimate, but sometimes a high degree of accuracy does not reflect the algorithm.

***Precision*** is for the results of our predictions and shows how many samples whose predictions are positive are correct. Then there are two possibilities to predict the positive class, one is to predict the positive class as class positive (TP) and the other is to predict the negative class as class positive (FP).

$$Precision = \frac{TP}{TP + FP} \tag{3}$$

***Recall*** is for our original sample and indicates how many positive examples in the sample are predicted correctly. There are also two possibilities, one is to predict the original positive class as class positive (TP) and the other is to predict the original positive class as class negative (FN). The recall rate is a measure of coverage.

$$Recall = \frac{TP}{TP + FN} \tag{4}$$

***F1-score***. Indicators Precision and Recall sometimes have contradictions, so they need to be considered comprehensively. The most common method is F-Measure (also known as F-Score). F-Measure is the weighted harmonic mean of precision and recall.

$$F1 = \frac{2 * P * R}{P + R} \tag{5}$$

***Support*** is the number of instances of each class[15].

From the experimental results presented in table 3, we will see that the most effective machine learning algorithm for our situation is the decision tree. But the results obtained through this practical work may not be suitable for all cases. As the number, value, and training data of various conditions and parameters change, the results of different algorithms may be different.

**Conclusion:**

A number of results were achieved during the work:

1. An overview of malicious software. Detailed information about PE files, its structure and nature has been provided.

2. Prepared data for analysis (more than 50,000 files). A method of separating files into clean and malicious files has been developed.

3. The most important features are selected. A complete list of obtained features can be seen in Table 1.

4. The use of modern machine learning algorithms and their comparative analysis were carried out within the framework of this task under the conditions of real resources and time. The most optimal, that is, the algorithm that showed the best result for the given task was selected.

A valuable practical result of the work is the creation of an efficient machine learning engine in terms of speed and quality.

## REFERENCES

1 Michael Sikorski, Andrew Honig (2018). *Practical Malware Analysis. The Hands-On Guide to Dissecting Malicious Software*. San Francisco, DC: No Starch Press.

2 Clarence Chio, David Freeman (2018). *Machine Learning and Security: Protecting Systems with Data and Algorithms*. Sebastopol, DC: O'Reilly.

3 Ucci D., Aniello L., Baldoni R. (2017). Survey on the Usage of Machine Learning Techniques for Malware Analysis. *ACM Transactions on the Web*, 1, 3-58.

4 B.A.S. Dilhara (2021). Classification of Malware using Machine learning and Deep Learning Techniques. *International Journal of Computer Applications,* (0975 – 8887) 183 – 32.

5 Damin Moon, JaeKoo Lee, MyungKeun Yoon (2021). Compact feature hashing for machine learning based malware detection. *Information & Communications Technology Express,* DOI: 10.1016/j.icte.2021.08.005.

6 Microsoft. (2022, June 23). PE Format. Microsoft website: https://learn.microsoft.com/en-us/windows/win32/debug/pe-format

7 Virus Total malware samples dataset (2022). Virus Total website: https://www.virustotal.com/gui/home/upload

8 Malware Traffic Analysis (2022). A source for packet capture (pcap) files and malware samples. Malware Traffic Analysis website: https://www.malware-traffic-analysis.net/

9 VirusShare (2022). VirusShare website: https://virusshare.com/

10 VX Heaven (2010). VX Heaven Virus Collection. VX Heaven website: https://web.archive.org/web/20170611163424/http://vxheaven.org/

11 Kaggle and Microsoft (2018). Microsoft Malware Prediction. Kaggle website: https://www.kaggle.com/c/microsoft-malware-prediction

12 Shannon C. E. (1948) A Mathematical Theory of Communication. *Bell System Technical Journal.* 27, 379-423.

13 Hyrum S. Anderson, Phil Roth (2018). *Research Gate*. An Open Dataset for Training Static PE MalwareMachine Learning Models. Research Gate website: https://www.researchgate.net/publication/324492745_EMBER_An_Open_Dataset_for_Training_Static_PE_Malware_Machine_Learning_Models

14 Henrik Brink, Joseph W. Richards, Mark Fetherolf (2017). Real-World Machine Learning. Shelter, Island. DC: Manning

15 Alexey Michurin (2022). Metrics in machine learning: precision, recall and more. Michurin website: http://www.michurin.net/computer-science/ml-precision-recall.html

***Г. АЛПЫСБАЙ, А. БЕДЕЛЬБАЕВ, А. УСМАНОВА, Ж. ДҮЙСЕНБЕКҚЫЗЫ***

*Әл-Фараби атындағы Қазақ ұлттық университеті, Алматы қ.,Қазақстан*

## ЗИЯНДЫ БАҒДАРЛАМАЛЫҚ ЖАБДЫҚТАРДЫ ТАЛДАУДА ҚОЛДАНЫЛАТЫН МАШИНАЛЫҚ ОҚЫТУ АЛГОРИТМДЕРІНЕ САЛЫСТЫРМАЛЫ ТАЛДАУ

*Заманауи ғылым әлеміндегі киберқауіпсіздік саласының жетістігіне және оның әдістерінің үздіксіз дамуына қарамастан, зиянды бағдарламалар әлі де ақпараттық қауіпсіздікке төнетін ең үлкен қатерлердің бірі болып табылады. Зиянды бағдарлама күн сайын дамып келеді және оның түрлері мен әрекеттері күннен-күнге артып, дамып келеді. Ал мұндай күрделі және алуан түрлі зиянды бағдарламаларды анықтау және олармен күресу үшін заманауи, күрделі технологияларды қолданудың маңыздылығы орасан. Осы орайда ақпараттық қауіпсіздік саласында интеллектуалды жүйелерді пайдаланудың артықшылықтарын атап өтуге болады. Бұл мақалада біз Windows операциялық жүйесіндегі PE (Portable Executable) файлдарын, яғни компьютерде орындалатын процестерді және машиналық оқыту алгоритмдерін пайдаланып зиянды бағдарламаларды талдауды қарастырамыз. Сонымен бірге біз әртүрлі машиналық оқыту алгоритмдерінің жұмысына назар аударамыз және біздің мысалымызда қай әдісті қолдану тиімдірек екенін көрсетеміз.*

*Бұл мақалада біз келесі тапсырмаларды орындаймыз:*

*1. Зиянды бағдарламалық қамтамасыз ету туралы ақпарат беру. PE файлдарының анықтамасы, оның құрылымы мен сипаты.*

*2. Тәжірибелік жұмысқа мәліметтерді дайындау (таза және зиянды кодтары бар файлдарды жинау). Файлдарды таза және зиянды файлдарға бөлу әдістеріне шолу.*

*3. Алдын ала дайындалған файлдар бойынша оқытуға қажетті белгілерді сұрыптау, яғни жаттығу кезінде ең дәл нәтижеге қол жеткізуге мүмкіндік беретін белгілерді ғана алу.*

*4. Бірнеше машиналық оқыту алгоритмдерін енгізу және олардың ішінен ең тиімдісін таңдау.*

***Түйін сөздер**: ақпараттық қауіпсіздік, зиянды бағдарламалық жабдық, PE файлдары, зиянды бағдарламаларды талдау, машиналық оқыту, белгілер.*

*Г. АЛПЫСБАЙ, А. БЕДЕЛЬБАЕВ, А. УСМАНОВА, Ж. ДУЙСЕНБЕККЫЗЫ*

*Казахский национальный университет им. аль-Фараби,
г. Алматы, Казахстан*

# СРАВНИТЕЛЬНЫЙ АНАЛИЗ АЛГОРИТМОВ МАШИННОГО ОБУЧЕНИЯ, ИСПОЛЬЗУЕМЫХ ДЛЯ АНАЛИЗА ВРЕДОНОСНОГО ПО

*Несмотря на достижения области кибербезопасности в современном мире науки и постоянное развитие ее методов, вредоносное ПО по-прежнему остается одной из самых больших угроз информационной безопасности. Вредоносное ПО развивается каждый день, а количество его типов и моделей поведения увеличивается день ото дня. И важность использования современных, совершенных технологий в выявлении и борьбе с такими сложными и разнообразными вредоносными программами возрастает. В связи с этим можно отметить преимущества использования интеллектуальных систем в сфере информационной безопасности. В этой статье мы проанализируем PE (Portable Executable) файлы в операционной системе Windows, то есть процессы, запущенные на компьютере, и проанализируем вредоносные программы с помощью алгоритмов машинного обучения. При этом мы сосредоточимся на работе разных алгоритмов машинного обучения и покажем, какой метод наиболее эффективно использовать для нашего примера.*

*В этой статье у нас будут следующие задачи:*

*1. Предоставление информации о вредоносном ПО. Определение PE-файлов, их структура и природа.*

*2. Подготовка данных для практической работы (сбор файлов с чистым и вредоносным кодом). Обзор методов разделения файлов на чистые и вредоносные.*

*3. Сортировка необходимых для обучения признаков по заранее подготовленным файлам, то есть получение только тех признаков, которые позволяют добиться наиболее точного результата при обучении.*

*4. Реализация нескольких алгоритмов машинного обучения и выбор среди них наиболее эффективного.*

***Ключевые слова**: информационная безопасность, вредоносное ПО, PE-файлы, анализ вредоносных программ, машинное обучение, признаки.*