**БЪЛГАРСКА АКАДЕМИЯ НА НАУКИТЕ**

ИНСТИТУТ ПО ИНФОРМАЦИОННИ И КОМУНИКАЦИОННИ ТЕХНОЛОГИИ

СЕКЦИЯ “ИНФОРМАЦИОННИ ПРОЦЕСИ И СИСТЕМИ ЗА ВЗЕМАНЕ НА РЕШЕНИЯ”

**Илиян Магдаленов Барзев**

# ИЗСЛЕДВАНЕ И АНАЛИЗ НА ВЪЗМОЖНОСТИТЕ ЗА ОТКРИВАНЕ НА ЗЛОНАМЕРЕН СОФТУЕР ЧРЕЗ СРЕДСТВАТА НА МАШИННО ОБУЧЕНИЕ

## ДИСЕРТАЦИЯ

за присъждане на образователна и научна степен “Доктор”
Професионално направление: 4.6. “Информатика и компютърни науки”
Докторска програма “Информатика”

**Научен ръководител**

проф. д.н. Даниела Борисова

2026

# СЪДЪРЖАНИЕ

# ИЗПОЛЗВАНИ СЪКРАЩЕНИЯ

- AI – Artificial Intelligence (Изкуствен интелект)
- API – Application Programming Interface (Приложен програмен интерфейс)
- AUC – Area Under the Curve (Площ под кривата)
- CatBoost – Библиотека за усилване на градиента, разработена от Yandex
- CLI – Command Line Interface (Команден интерфейс)
- CNN – Convolutional Neural Networks (Конволюционни невронни мрежи)
- CSV – Comma-Separated Values (Стойности, разделени със запетая)
- DDoS – Distributed Denial of Service (Разпределен отказ на услуга)
- DL – Deep Learning (Дълбоко обучение)
- DLL – Dynamic Link Library (Библиотека за динамично свързване)
- DT – Decision Trees (Дървета на решенията)
- EDA – Exploratory Data Analysis (Проучвателен анализ на данни)
- EMBER 2018 – Elastic Malware Benchmark for Empowering Researchers 2018 (Набор от данни за статичен анализ на зловреден софтуер, версия 2018)
- EMBER 2024 – Elastic Malware Benchmark for Empowering Researchers 2024 (Набор от данни за статичен анализ на зловреден софтуер, версия 2024)
- F1-Score – Средно хармонично на прецизност (Precision) и попълнота (Recall)
- FN – False Negative (Фалшиво отрицателен резултат)
- FP – False Positive (Фалшиво положителен резултат)
- FPR – False Positive Rate (Честота на фалшиво положителни резултати)
- GUI – Graphical User Interface (Графичен потребителски интерфейс)
- H1 – хибриден алгоритъм на базата на NB и KNN
- H2 – хибриден алгоритъм на базата на NB и RF
- H3 – хибриден алгоритъм на базата на NB, KNN и RF
- IDS – Intrusion Detection System (Система за откриване на проникванията)
- IoT-23 – A labeled dataset of IoT malware and benign traffic (Набор от данни със зловреден и доброкачествен трафик от Интернет на нещата)
- JSON – JavaScript Object Notation (Формат за обмен на данни)
- K-NN – K-Nearest Neighbors (К-най-близки съседи)
- LightGBM – Light Gradient Boosting Machine (Лека машина за усилване на градиента)
- LIME – Local Interpretable Model-agnostic Explanations (Локални интерпретируеми обяснения, независими от модела)
- LSTM – Long Short-Term Memory Networks (Мрежи с дълга краткосрочна памет)

- MIS – Mutual Information Score (Резултат за взаимна информация)
- NB – Naive Bayes (Наивен Бейсов класификатор)
- OOB – Out-of-Bag (Оценка на грешката извън обучителната извадка)
- PCA – Principal Component Analysis (Метод на главните компоненти)
- PE – Portable Executable (Формат на изпълними файлове в Windows)
- RF – Random Forest (Случайна гора)
- ROC – Receiver Operating Characteristic (Крива на оперативната характеристика на приемника)
- SAMC – Self-Aware Model Classifier (Самоосъзнат класификатор на модели)
- SHAP – SHapley Additive exPlanations (Обяснения чрез адитивни стойности на Шапли)
- SVD – Singular Value Decomposition (Разлагане по сингулярни стойности)
- SVM – Support Vector Machines (Машини с поддържащи вектори)
- TN – True Negative (Истински отрицателен резултат)
- TP – True Positive (Истински положителен резултат)
- t-SNE – Distributed Stochastic Neighbor Embedding (t-разпределено стохастично вграждане на съседи)
- VM – Virtual Machine (Виртуална машина)
- XAI – Explainable artificial intelligence (обясним изкуствен интелект)
- XGBoost – eXtreme Gradient Boosting (Екстремно усилване на градиента)
- МО – машинно обучение

# УВОД

Днес защитата от зловреден софтуер е важна по няколко причини. Едната се отнася до сигурността на данните, тъй като злонамереният софтуер може да компрометира лични и финансови данни, което води до кражба на самоличност или финансови загуби. Втората е свързана със състоянието на системите, тъй като инфекциите с вируси и троянски коне могат да повредят компютрите и мрежите, което води до загуба на информация и скъпи ремонти. Злонамереният софтуер може също да повлияе на производителността, като причинява сривове, нарушаващи ежедневните операции и загуба на време и ресурси. Всички тези ситуации се отразяват на репутацията на компаниите – компрометирането на данни може да повлияе на доверието на клиентите, което е трудно за възстановяване. Не на последно място е спазването на законите и разпоредбите, тъй като много организации са задължени по закон да защитават данните на своите клиенти. Нарушенията могат да доведат до сериозни санкции. Злонамереният софтуер може да се разпространява от едно устройство на друго, застрашавайки цялата мрежова сигурност.

Анализът на зловреден софтуер е процесът на локализиране и изследване на зловреден софтуер или код с цел разбиране на неговото действие и разработване на контрамерки. Зловреден софтуер може да приеме много форми, като вируси, червеи, троянски коне и ransomware, и може да причини значителни вреди на хора, организации и дори цели държави. За да се определи целта, потенциалните ефекти и възможности на даден зловреден софтуер, анализът на зловреден софтуер включва изследване на поведението, структурата и функционалностите на зловредния софтуер. Анализаторите на зловреден софтуер са от съществено значение за сектора на киберсигурността, защото се стремят да откриват опасности, да ги елиминират и да се защитават от онлайн атаки. Чрез използване на знанията, получени от анализа на зловреден софтуер, могат да се създадат решения за сигурност, които ще защитят по-добре бизнеса от опасен софтуер. Анализът на зловреден софтуер е ключова част от всяка успешна стратегия за киберсигурност в непрекъснато променящия се пейзаж на заплахите днес.

С еволюцията на дигиталните технологии се променят и методите за извършване на атаки, което налага разработване на нови алгоритми за ефективно и навременно откриване на зловредно поведение. Традиционните решения, базирани предимно на сигнатури, вече не са достатъчни за справяне със съвременните заплахи. Исторически антивирусните системи използват сигнатурно-базирани алгоритми, които сравняват файлове с база данни от вече известни вредители. Този подход има няколко ключови недостатъка: не идентифицира нова или мутирала заплаха; изисква непрекъснато обновяване на бази данни; не е ефективен срещу полиморфен, метаморфен и fileless malware. Развитието на зловредния софтуер значително изпреварва възможностите на сигнатурните системи. От друга страна, днешният malware използва усъвършенствани техники за укриване и адаптация – кодът се променя при всяко изпълнение; а атаките се изпълняват само в паметта, автоматично се генерират и нови варианти. Всички тези фактори повишават сложността на защитата и изискват нови методи за откриване.

Ръстът на атаките експоненциално нараства, като по данни на световни изследователски центрове ежедневно се появяват десетки хиляди нови модификации. Налице е неотложна необходимост от проактивна защита. Съвременните системи трябва да откриват заплахи преди да са вредили, а това би могло да се реализара чрез машинно обучение и поведенчески анализ, който изследване на поведението, структурата и функционалностите на malware.

Следователно, разработването на нови алгоритми за откриване на зловреден софтуер е критично за осигуряване на информационна сигурност в съвременната дигитална среда. Еволюцията на заплахите налага използването на интелигентни, адаптивни и мащабируеми подходи, които могат да реагират на динамиката на модерните кибератаки. Инвестициите в иновации и алгоритмични модели са ключови за изграждане на устойчиви защитни системи.

Като се взимат предвид всички тези аспекти, става ясно, че провеждането на изследвания и анализи на възможностите за откриване на злонамерен софтуер чрез средствата на машинно обучение е една актуална и бързо развиваща се научна област.

Изложението на дисертационния труд е структурирано в увод, три глави, заключение – резюме на получените резултати, приноси, списък на публикациите, списък на забелязаните цитирания, декларация за оригиналност и библиография.

В **Глава 1** е направен анализ на различни алгоритми на машинното обучение относно тяхното представянето за целите на откриване на зловреден софтуер. Представено е тестване на приложимостта на алгоритми за двоична класификация за откриване на зловреден софтуер, използвайки публичен набор от данни, заразен с 9 вида зловреден софтуер, чрез предложена методология. Следвайки тази методология и определените показатели, резултатите показват приложимост на изследвани алгоритми, като един от тях (Random Forest) демонстрира по-добра производителност. Показано е, че хибридният модел CNN-LSTM значително превъзхожда други тествани подходи, постигайки точност от 0,97 и също толкова високи стойности за прецизност, попълнота и F1-Score. Този хибриден модел комбинира силните страни на възможностите за извличане на характеристики на CNN със способността на LSTM да улавя времеви зависимости, което го прави особено ефективен за данни от IoT-23.

В **Глава 2** е описан предложен модел за избор на виртуална машина за целите на провеждането на експерименти за откриване на зловреден софтуер. Представен е подобрен подход на статичен анализ чрез оптимизиране извличането на характеристики, който комбинира различни алгоритми за машинно обучение за целите на откриване на зловреден софтуер. Представена е предложена рамка за статична класификация на зловреден софтуер, използваща оптимизация на функции и ансамблово обучение. С цел подобряване на класификация на зловреден софтуер е предложена самоосъзната рамка, интегрираща рутиране на модели на базата на система за доверие за избора и обяснимост на характеристиките. За прецизиране на класификация на зловредния софтуер е предложена адаптивна рамка, съобразена с доверието, позволяваща класификация на зловреден софтуер с възможност за корекции чрез обратна връзка. Тази адаптивна рамка е реализирана в разработена демо версия на софтуерно приложение под името „Shipka Guard“. Тя интегрира механизъм за рутиране, съобразен с доверието, който използва множество поколения модели, включително резервен механизъм. В нея се използва буфер за обратна връзка и слой за корекции за лека адаптация от потребителски корекции и синтетични инжекции.

В **Глава 3** са представени резултатите от проведеното тестване на предложените модели за избор на софтуер за виртуална машина. Описани са резултати от тестуването на предложения подобрен подход на статичен анализ чрез оптимизиране извличането

на характеристики, комбинирайки различни алгоритми за машинно обучение. Описани са числени експерименти, използвайки предложената рамка за статична класификация на зловреден софтуер, в която е направено оптимизиране на функциите и е използвано ансамблово обучение. Показани са също резултати от тестваната самоосъзната рамка, интегрираща рутиране на модели на базата на система за доверие за избора и обяснимост на характеристиките. Тук е представено разработеното и тествано приложение Shipka Guard, интегриращо предложената адаптивна рамка, съобразена с доверието, позволяваща класификация на зловреден софтуер с възможност за корекции чрез обратна връзка. Показано е, че буферът за обратна връзка дава възможност на потребителя съвместно да коригира модела, което му позволява да експортира/импортира файлове с обратна връзка. От друга страна, интеграцията на обяснимостта допринася за доверието в решенията. Всичко това е реализирано в разработена демо версия на софтуерно приложение под името „Shipka Guard“.

В заключението е направено обобщение на получените резултати в следствие на проведените изследвания, предмет на настоящия дисертационен труд. Показани са някои основни посоки, който да се използват в бъдещи изследвания.

# ГЛАВА 1. АНАЛИЗ НА ТЕХНИКИТЕ ЗА ОТКРИВАНЕ И КЛАСИФИЦИРАНЕ НА ЗЛОВРЕДЕН СОФТУЕР

В настоящата глава е направен анализ и сравнение на различни алгоритми на машинното обучение (МО) относно тяхното представянето за целите на откриване на зловреден софтуер. Представено е сравнение на различни алгоритми за двоична класификация и е анализирана производителността на някои алгоритми за откриване на зловреден софтуер в дейта сет от областта на интернет на нещата. На база на проведените анализи и сравнения е формулирана целта и задачите на дисертационното изследване.

## 1.1. Анализ и класификация на алгоритмите на машинно обучение за откриване на злонамерен софтуер

МО като подобласт на изкуствения интелект предоставя възможност за правене на прогнози въз основа на модели, научени директно от данни, без да са изрично програмирани за това (Ozkan-Okay et al., 2024). МО заема ключова роля в откриването и смекчаване на сложните заплахи на зловредния софтуер по различни начини. Чрез средствата на МО могат да анализират големи обеми от данни, да се класифицират файлове и програми, да се прогнозират заплахи и др. (Фиг. 1.1).


*Фиг. 1.1. Приложение на машинното обучение.*

- *Анализ на данни:* Алгоритмите за машинно обучение могат да анализират големи обеми от данни и да идентифицират модели и аномалии, свързани с поведението на зловредния софтуер (Kumar et al., 2026).
- *Класификация:* Системите за машинно обучение могат да класифицират файлове и програми като злонамерени или безопасни въз основа на предварително обучени модели вкл. подходи за машинно обучение, използващи квантови изчисления за анализ на големи данни (Singh et al.,2025).
- *Поведенчески анализ:* Вместо да разчита единствено на сигнатури (характеристики на известен зловреден софтуер), машинното обучение може да наблюдава как се държи софтуерът, идентифицирайки подозрителни действия.
- *Автоматично обучение:* Алгоритмите могат да се адаптират и подобряват, когато се сблъскват с нови заплахи, което им позволява да разпознават нови видове зловреден софтуер (Razaque et al., 2025).
- *Прогнозиране на заплахи:* Машинното обучение може да предсказва потенциални заплахи чрез анализ на исторически данни и тенденции, което позволява организиране на предварителна подготвка (Salman et al., 2025).
- *Семантично разбиране:* Някои от модели за машинно обучение могат да извличат информация от контекста на файловете, подобрявайки откриването на зловреден софтуер в сложни сценарии (Mohamed, 2025).
- *Класификация на имейли:* Алгоритмите за машинно обучение могат да откриват фишинг атаки, като анализират съдържанието на имейлите и идентифицират подозрителни модели (Ayeni, et al., 2024).

Машинното обучение предлага нови инструменти за преодоляване на предизвикателства, за които традиционните статистически методи не са добре подходящи. Поади това, интерес представлява анализа на специфичните особености на контролираното машинно обучение и на неконтролирано машинно обучение.

### 1.1.1. Контролирано обучение машинно обучение

При ситуации на използване на контролираното машинно обучение (Supervised Machine Learning), моделът първо се обучава върху вече етикетирани данни, при което

всеки вход е свързан с известен изход. Целта на този вид обучение е да се научи функция, която да свързва входовете с изходите. След това, моделът може да се използва за прогнозиране на нови, невидени входове. Съществуват два основни типа на контролирано обучение (Jiang et al., 2020):

- **Регресия**: Регресията е подходяща, когато е необходимо да се предвиди непрекъснат резултат, който попада в даден диапазон. Например, прогнозирането на цената на жилището въз основа на квадратурата, местоположението, и други параметри.
- **Класификация**: Приложението на класификация може да се илюстрира със ситуация, при която се опитваме да класифицираме дали даден резултат попада в две или повече категории. Например, детекторите за спам са модели за класификация (спам или не спам).

### 1.1.2. Неконтролирано машинно обучение

Моделът при неконтролираното обучение (Unsupervised Machine Learning) се обучава върху данни, които не са етикетирани, с което се цели да се открият скрити модели и структури. Същността при този вид обучение е да се разбере структурата на данните, без да се използват известни изходи. Този тип алгоритми могат да откриват общи модели в данните, без изрично да им се показват изходите (Usmani et al., 2022). Алгоритмите за неконтролирано обучение обикновено се използват за групиране и клъстериране на различни обекти и субекти.

#### Обучение с подсилване (Reinforcement Learning)

Обучението с подсилване е подмножество от алгоритми за машинно обучение, които използват награди, за да насърчат желано поведение или прогноза, и наказание в противен случай. Обучението с подсилване е техника за поведенческо моделиране, при която моделът се учи чрез механизъм на проба-грешка, докато продължава да взаимодейства със средата (Shakya et al., 2023).

#### Самостоятелно машинно обучение (Self-Supervised Machine Learning)

Самостоятелното обучение е ефективна по отношение на данните техника за машинно обучение, при която моделът се учи от немаркиран примерен набор от данни.

Самоконтролираното обучение е вид неконтролирано обучение, което помага при изпълнението на задачи надолу по веригата, свързани с компютърно зрение, като например откриване на обекти, разбиране на изображения, сегментиране на изображения и т.н (Rani et al., 2023).

### 1.1.3. Популярни алгоритми с контролирано обучение

Някои от популялните алгоритми с контролирано обучение са представени накратко:

1. Linear Regression е алгоритъм, който моделира линейна връзка между една или повече обяснителни променливи и непрекъсната числова изходна променлива. Линейна регресия е контролиран алгоритъм за обучение. Предимството му се крие в способността да обяснява и интерпретира прогнозите на модела (James et al., 2023).
2. Naive Bayes (NB) е контролиран алгоритъм за обучение често използван за класификационни задачи. Тези тип алгоритми предполагат независимост на характеристиките една от друга.
3. Logistic regression е също контролиран алгоритъм за обучение, който намира приложение при двоично класифицирани проблеми.
4. K-Nearest Neighbors (kNN) е алгоритъм, подходящ за решаването както на задачи за класификация, така и на регресия. Основната идея тук е, че алгоритъмът се стреми да класифицира неизвестна извадка въз основа на известната класификация на нейните съседи (Mucherino et al., 2009; Larose & Larose, 2014).
5. Дървета на решенията (Decision Trees) – дървовидна структура от правила за вземане на решения, които се прилагат към входните характеристики, за да се предскажат възможните резултати (Mienye & Jere, 2024). Може да се използва за класификация или регресия.
6. Random Forest (RF) – основава се на недостатъците на свръхнапасването, които се наблюдават в моделите на дърветата на решенията. Свръхнапасването е когато алгоритмите са обучени твърде добре върху данните за обучение и не успяват да обобщят или да предоставят точни прогнози върху невидими данни. Случайната гора решава проблема с

свръхнапасването, като изгражда множество дървета на решенията върху произволно избрани извадки от данните (Iranzad & Liu, 2025).

7. Support Vector Machines (SVM) е контролиран алгоритъм за обучение, който обикновено се използва за решаване на задачи, свързани с класификация. Това е техника за машинно обучение, основана на принципа за минимизиране на структурния риск (Roy & Chakraborty, 2023).

## 1.2. Сравнение на различни алгоритми за двоична класификация

В света на дигиталните комуникации, сигурността и поверителността на информацията са сред основните критерии, които всички се стремим да постигнем. Като доказателство за това е скорошен доклад за Европейската мрежа от центрове за киберсигурност и Център за компетентност за иновации и операции (Rajamaki et al., 2022). Зловреднният софтуер е един от най-съществените проблеми с киберсигурността, създаден от интелигентни нападатели, които се стремят да хакнат чувствителни данни, като по този начин причиняват щети на физически устройства и нарушават нормалните мрежови операции.

Точното откриване на зловреден софтуер е от съществено значение за защитата на компютрите от инфекция, както и за премахването на зловреден софтуер от компрометирана компютърна система. Следователно, усилията на много изследователи са насочени към разработването на различни подходи за справяне с откриването на зловреден софтуер, използващи техники за машинно обучение (Soja Rani & Reeja, 2020). Систематичен преглед на откриването на зловреден софтуер, използващ модели за дълбоко обучение, е представен в (Gopinath & Sethuraman, 2023). Показано е, че разработването на система за откриване на зловреден софтуер е предизвикателство. Сравнение на производителността на различни ансамбъл класификатори за откриване на зловреден софтуер е показано в (Dhanya et al., 2022). Някои автори се занимават с проблемите на откриването и анализа на зловреден софтуер в изпълними файлове и приложения за Android, включително безфайлов зловреден софтуер (Melvin & Kathrine, 2021; Patil et al., 2020; Singh & Singh, 2021; Razgallah et al. 2021; Khalid et al., 2023). Освен това е възможно да се използват методи за статистическо откриване на дрейф, за да се открие всяка промяна в моделите на данни и да се обучат класификаторите за машинно обучение да противодействат на

новоразработен зловреден софтуер (Amin et al., 2023). Трябва да се отбележи, че авторите на зловреден софтуер често използват криптографски инструменти, за да скрият част от зловредния софтуер, за да избегнат откриването му (Asghar et al., 2023).

Откриването на зловреден софтуер може да се счита за процес на вземане на решения, тъй като в края на този процес злонамерената програма трябва да бъде идентифицирана. Процесът на откриване на зловреден софтуер включва набор от различни защитни техники, които са способни да идентифицират, блокират и предотвратят вредните ефекти на зловреден софтуер. Тези техники могат да бъдат разделени най-общо на статични и динамични техники за откриване. Статичното откриване на зловреден софтуер разчита само на правилата или на фина настройка на правилата с течение на времето, за да се увеличи покритието. Следователно, обещаващата посока в откриването на зловреден софтуер се основава на динамични техники за откриване. Динамичните техники са базирани на изкуствен интелект и машинно обучение и биха могли да се използват в инструменти за сигурност, за да се научим да правим разлика между легитимни и злонамерени файлове и процеси.

Машинното обучение се отнася до процеса на обучение на алгоритми да учат модели от съществуващи данни, за да предсказват отговори на нови данни. Това се осъществява чрез различни техники чрез наблюдение на мрежовия трафик, честотата на процесите, моделите на внедряване и др. С течение на времето тези алгоритми за машинно обучение могат да научат как изглеждат „лошите“ файлове, което прави възможно откриването на нов и неизвестен зловреден софтуер. Откриването на зловреден софтуер чрез машинно обучение е известно като „поведенческо“ откриване поради факта, че прави анализ на поведението на подозрителните процеси. Като се има предвид наличието на различни алгоритми за машинно обучение, целта е да се сравни приложимостта и точността на често използваните за откриване на зловреден софтуер в различни обеми от данни.

### 1.2.1. Изследователски проблем

В резултат на значителното количество данни, генерирани във всеки аспект от живота ни, е важно да се разработят бързи и ефикасни алгоритми за обработка на данни в реално време. Техниките за машинно обучение и изкуствен интелект предоставят интелигентни алтернативи за анализ на големи обеми данни.

Процесът на машинно обучение разчита на два основни етапа: (1) фаза на обучение, където се обучава алгоритъмът за класификация, и (2) фаза на класификация. Задачата за класификация (известна още като контролирано обучение) е задачата, при която се извличат данни и при която данните се преобразуват в предварително дефинирани групи и класове. Седователно, класификацията чрез машинно обучение цели реализирано на автоматична класификация или категоризация на данни въз основа на определени критерии. Методи като логистична регресия, K-NN, DT и SVM са методи, базирани на техники за двоична класификация (Kotu & Deshpande, 2019).

Един от най-мощните инструменти на алгоритмите за контролирано обучение, използвани както за класификационни, така и за регресионни задачи, е DT (Hermawan et al., 2021; Saravanan & Gayathri, 2018). Този класификатор извършва многоетапни класификации, като използва серия от двоични решения, често използвани за управление на риска, класификация на изображения, диагностика в медицината и др. (Li et al., 2022; Al Mamun & Keikhosrokiani, 2022; Berikol & Berikol, 2020; Hou et al., 2022). В дърво на решения идеята е наборът от данни да се раздели въз основа на хомогенността на данните. Дървото се изгражда чрез разделяне на изходния набор на подмножества въз основа на набор от правила за разделяне, базирани на класификационни характеристики. При изграждането на дърво, алгоритъмът J48 игнорира липсващите стойности; стойността на този елемент може да се предскаже от стойностите на атрибутите за другите записи. Основната идея е данните да се разделят на диапазони въз основа на стойностите на атрибутите, намерени в обучаващата извадка. Алгоритъмът J48 (или C4.5) използва рекурсивна стратегия „разделяй и владей“ отгоре надолу, която използва мярка, наречена „информационно усилване“, за да избере атрибута на всеки етап. Например, ако всеки атрибут има свързано с него информационно усилване, е възможно да се дефинира намаляване на ентропията на атрибута, причинено от разделяне на инстанциите въз основа на стойностите, получени от този атрибут. В тази ситуация информационното усилване за определен атрибут $A$ може да се изчисли по следния начин (Puga et al., 2015):

$$IG(N, A) = E(N) - \sum_{value(A)} \frac{|N_i|}{|N|} E(N) \tag{1.1}$$

където $N$ е множеството от инстанции в даден възел и $|N|$ е неговата кардиналност, $N_i$ е подмножеството от $N$ а което атрибут $A$ има стойност $i$, а ентропията на $E$ на множеството $N$ се определя като:

$$E(N) = \sum_{i=1}^{No.ofClasses} -P_i \, log_2 P_i \tag{1.2}$$

където $P_i$ е пропорцията на инстанциите в $N$ които имат стойността на *i-тия* клас като изходни атрибути.

NB класификаторите имат силни предположения за независимост между характеристиките. Тази тип класификация, базирани на теоремата на Бейс, се използва за решаване на класификационни проблеми. Наивният Бейсов класификатор се използва широко в машинното обучение поради високата си мащабируемост, изискваща редица параметри, линейни по броя на променливите (характеристики/предиктори) в обучителния проблем. Теоремата на Бейс за дадена класова променлива се изразява чрез следната зависимост (Puga et al., 2015):

$$P(A|B) = \frac{P(A \cap B)}{P(B)} = \frac{P(B|A)P(A)}{P(B)} \tag{1.3}$$

където $P(A)$ и $P(B)$ са вероятностите за възникване на $A$ и съответно $B$ да се случат, $P(A|B)$ е вероятността за $A$ при дадено $B$, $P(B|A)$ е за вероятността $B$ при дадено $A$, и $P(A \cap B)$ е вероятността за възникване и на $A$, и на $B$.

Naive Bayes класификатори са набор от алгоритми за обучение, базирани на теоремата на Бейс с предположение за условна независимост между всички независими променливи. Теоремата на Бейс определя вероятността за настъпване на едно събитие, при условие че друго събитие вече се е случило, т.е. условна вероятност. Условната вероятност включва допълнителни условия и съответно повече данни, които допринасят за по-точни резултати.

SVM като контролиран алгоритъм за машинно обучение се използва за класифициране на точки от данни в два класа чрез намиране на разстоянието между групите точки от данни и максимизиране на разликата между тях. Той е изключително популярен поради относителната си простота и гъвкавост при справянето с редица проблеми с класификацията (Awad & Khanna, 2015; Pisner & Schnyer, 2020; Kecman, 2005). SVM работи на принципа на напасване на граница към област от точки, които са еднакви, т.е. които принадлежат към един клас (Kotu & Deshpande, 2019). SVM

класифицира данните, като намира най-добрата хиперравнина, която разделя всички точки от данни от един клас от тези от другия клас. Най-добрата хиперравнина за SVM е тази с най-голямо разстояние между двата класа. Например, ако са дадени обучаващ вектор $x_i \in R^p, i = 1,2, \dots, n$ в два класа и вектор $y \in \{1, -1\}^n$ целта е да се намерят $w \in R^p$ and $b \in R$ така че прогнозата дадена от знака $(w^t\phi(x) + b)$ да е правилна за повечето извадки. В тази връзка, SVM решава следния първичен проблем (Christmann & Steinwart, 2008):

$$\min_{w,b,\zeta} \frac{1}{2} w^T w + C \sum_{i=1}^{n} \zeta_i \tag{1.4}$$

при ограничения

$$y_i(w^T\phi(x_i) + b) \geq 1 - \zeta_i \tag{1.5}$$

$$\zeta_i \geq 0, i = 1,2, \dots, n \tag{1.6}$$

Тъй като проблемите обикновено не винаги са напълно разделими с хиперравнина, може да се позволи някои проби да бъдат на разстояние $\zeta_i$ от правилната граница на полето. Параметърът $C$ контролира компромиса между правилното класифициране на точките от тренировъчния набор и поддържането на широк марж и действа като обратен параметър на регуларизация от двойствената задача към първичната, както следва:

$$\min_{\alpha} \frac{1}{2} \alpha^T Q_\alpha - e^T \alpha \tag{1.7}$$

при ограничения

$$0 \leq \alpha_i \leq C, i = 1,2, \dots, n \tag{1.8}$$

Тук $e$ е вектор от всички единици, $Q$ е положителна полудефинирана матрица, и $Q_{ij} \equiv y_i y_j K(x_i, x_j)$ където $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$ е ядрото (kernel).

### 1.2.2. Методология на експеримента

За откриването на зловреден софтуер, използваните алгоритми и подходящата предварителна обработка на данните са от голямо значение. В тази връзка, изследователският проблем, разгледан тук, има за цел да сравни приложимостта и точността на различни алгоритми за откриване на зловреден софтуер в един и същ обем

данни. За опростяване ще бъде следвана следната диаграма, за да се сравни приложимостта на някои алгоритми за двоична класификация за откриване на зловреден софтуер, както е показано на Фиг. 1.2 (Barzev et al., 2024).


*Фиг. 1.2. Блок-схема на методологията на експеримента.*

Стъпка 1 е свързана с дефинирането на проблема – в конкретния случай това е сравняване на различни алгоритми за откриване на зловреден софтуер в различни обеми. Следващата стъпка 2 е събиране и придобиване на данни, където се идентифицира и събира набор от данни за съответния проблем. Данните могат да идват от различни източници, включително бази данни, API, сензори или ръчно събиране. В конкретния случай се използва **BIG2015 Dataset**.

Стъпка 3 се отнася до пречистване на данните преди употреба. Това включва обработка на липсващи стойности, премахване на дубликати и справяне с отклонения. Пречистването на данните гарантира, че наборът от данни е надежден и без грешки, които могат да повлияят на производителността на модела.

Стъпка 4 е свързана с проучвателен анализ на данни (Exploratory Data Analysis – EDA), за да се разберат характеристиките на набора от данни, включително неговото разпределение, корелации между променливите и всякакви модели или тенденции. Инструменти за визуализация и статистически техники могат да помогнат в този процес като се използва библиотеката Seaborn на Python за топлинни карти.

Стъпка 5 се отнася до предварителната обработка на данни чрез извършване на различни задачи за тази обработка, включително: 1) мащабиране на характеристики; 2) инженерство на характеристики; 3) кодиране на категорични данни; и 4) разделяне на данни. Мащабирането на характеристиките има за цел да нормализира или стандартизира характеристиките, за да се гарантира, че те имат подобен мащаб. Инженерингът на характеристиките създава нови характеристики или трансформира съществуващи, за да улови подходяща информация. Кодирането на категорични данни преобразува категорични променливи в числов формат, използвайки техники като еднократно кодиране или кодиране с етикети. Разделянето на данните има за цел да раздели набора от данни на обучителни, валидационни и тестови набори, за да се оцени производителността на модела. За конкретния случай на употреба експериментите са подготвени с различни обеми от един и същ набор от данни, в мащаб 1:10, 1:8, 1:6, 1:4, 1:1, и обработени в среда на Python, използвайки библиотеката Sklearn на Python.

Изборът на характеристики (стъпка 6) определя най-подходящите характеристики за конкретния проблем с машинното обучение. Тази стъпка е една от важните, тъй като изборът на характеристики може да помогне за намаляване на размерността и подобряване на производителността на модела.

Изборът на алгоритъм за машинно обучение на стъпка 7 има за цел да избере подходящ алгоритъм за машинно обучение въз основа на типа на проблема (класификация, регресия, клъстеризация и др.) и характеристиките на набора от данни. Започва с прости модели и постепенно преминава към по-сложни. За нашия случай ще бъде използван следният двоичен алгоритъм за класификация: Наивен Бейсов алгоритъм, J48 дървета на решенията и машини с поддържащи вектори. Обучението на модела се отнася до използването на набора от тренировъчни данни за обучение на избрания модел.

На стъпка 8 моделът изучава особености и връзки между входните характеристики и целевите резултати. Оценяването на модела има за цел да оцени производителността на модела, използвайки показатели за оценка, отнасящи се за конкретен проблем.

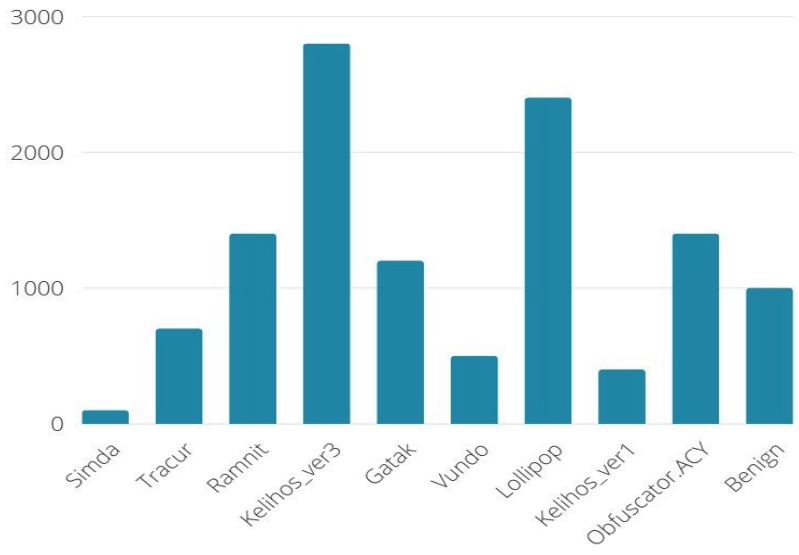
При стъпка 9 наборът от данни за валидиране може да се използва за фина настройка на модела. За нашия случай целта е точността на F1 резултата.

Стъпка 10 от методологията е свързана с тестване на модела и има за цел да го оцени върху тестовия набор, за да се получи окончателна оценка на неговата обща производителност.

Последната стъпка 11 се отнася до документиране на целия процес, включително източници на данни, стъпки за предварителна обработка, подробности за модела и показатели за производителност. Тази документация е от съществено значение за възпроизводимостта и бъдещите справки.

### 1.2.3. Сравнение между различните алгоритми

Наборът от данни, използван в този експеримент, се нарича BIG2015 Dataset. Той съдържа етикетирани доброкачествени и злонамерени PE и OLE файлове. Изтеглена и изследвана е извадката заедно с етикетираната информация, предоставена от създателите на BIG2015 Dataset във файлов формат csv. Избран е този набор от данни, тъй като той предоставя добра възможност да изследваме различните методи на алгоритмите за машинно обучение, без да е необходимо да разполагаме с високопроизводителни ресурси. Друго предимство са относително новите семейства зловреден софтуер, съдържащи се в набора от данни. Разпределението на зловредния софтуер в набора от данни е показано на Фиг. 1.3.




*Фиг. 1.3. Представяне на разпространението на зловреден софтуер в набора от данни.*

За да добием по-ясна представа, в Таблица 1.1 са показани различните видове зловреден софтуер, с които е бил заразен наборът от данни.

*Таблица 1.1. Брой и видове зловреден софтуер, с който е заразен наборът от данни*

| No | Вид злонамерен софтуер | Брой |
|---|---|---|
| **1** | Simda | 103 |
| **2** | Tracur | 712 |
| **3** | Ramnit | 1411 |
| **4** | Kelihos_ver3 | 2840 |
| **5** | Gatak | 1224 |
| **6** | Vundo | 498 |
| **7** | Lollipop | 2452 |
| **8** | Kalihos_ver1 | 439 |
| **9** | Obfuscator.ACY | 1385 |
| **10** | Benign | 1083 |

Доброкачествените файлове не съдържат никакъв зловреден софтуер, но са включени в наборите от данни за зловреден софтуер по няколко причини. Например, чрез тях се осигурява по-реалистична тестова среда и се използват за анализ на фалшиво положителни резултати.

**Извличане на характеристики**

За да се стартират алгоритмите за откриване на зловреден софтуер, първоначалната стъпка включва избиране на специфични характеристики, които да бъдат извлечени от файловете на набора от данни, наречени *features*. Впоследствие е необходим предварителен анализ на набора от данни, за да се определи кои характеристики трябва да бъдат извлечени и използвани като входни данни за алгоритмите за машинно обучение. Най-значимите характеристики, които влияят върху резултатите при откриването на зловреден софтуер, са: размер на двоични файлове, байтове n-грама, ентропия, n-грами на опкод, брояч на опкодове, n-грама на API и проверка на API.

**Избор на алгоритми за машинно обучение:**

За целите на проведеното на изследване е използвана библиотеката на Python, наречена scikit-learn. Трябва да се отбележи, че всеки алгоритъм може да има леко различни имплементации в различните налични библиотеки. Scikit-learn поддържа различни класификатори, обхващащи опции като Random Forest, Support Vector Machines – SVM, Gradient Boost и други. Относно работния процес на алгоритъма, първоначално се започва с разделяне на набора от данни на два отделни сегмента:

тренировъчен набор и тестов набор. Scikit-learn предоставя специализирана функция, която рационализира процеса на разделяне на пробите.

За конкретния разглеждан случай е взето решение наборът от данни да се раздели в следното съотношение: 75% от пробите са използвани в тренировъчния набор, а останалите 25% съставляват тестовия набор. Освен това, и обучението е разделено на две подмножества. Това е важно, за да се предотврати пренареждането, което може да доведе до по-малко точни резултати. Пренареждането се получава, когато моделът стане изключително успешен в класифицирането на пробите в рамките на обучителния набор, но се затруднява да обобщи и точно класифицира невидими данни. Накрая получаваме три различни извличания на набори от данни: (1) тренировъчен набор, (2) валидационен набор и (3) тестови набор. Това води до 68% тренировъчен набор, 12% валидационен тест и 20% тестов набор.

Избраните алгоритми за провеждане на сравнението са Random Forest, K-Nearest Neighbors и Support Vector Machines.

**Метрики**

Преди да се стигне до резултата от експеримента, е важно да се определят метриките или показателите, които ще се използват за определяне точността на алгоритмите, т.е. тяхното представяне спрямо набора от данни. Избраните показатели са: 1) Точност (Accuracy), 2) Прецизност (Precision), 3) Попълнота (Recall) и 4) F-1 резултат.

**Анализ на резултатите**

Получените стойности на изследваните метрики за избраните алгоритми Random Forest, K-Nearest Neighbors и Support Vector Machines върху изследвания набор от данни са показани в Таблица 1.2.

*Таблица 1.2. Резултати от използваните показатели*

| Алгоритъм | Метрики | | | |
|---|---|---|---|---|
| | Accuracy | Precision | Recall | F-1 |
| **RF (Random Forest)** | 0.9825 | 0.9812 | 0.9999 | 0.9893 |
| **K-NN (K-Nearest Neighbors)** | 0.8632 | 0.8911 | 0.9938 | 0.9731 |
| **SVM (Support Vector Machines)** | 0.7819 | 0.9653 | 0.7309 | 0.8487 |

От Таблица 1.2 може да се установи, че най-добрият алгоритъм за използваната метрика е Random Forest. Това се дължи на точността на Random Forest, която е най-добра със стойност 0.9825, следвана от K-NN със стойност 0.8632 и последна е Support Vector Machines с 0.7819. Относно вторият показател – прецизност, резултатите са аналогични и най-добра производителност показва алгоритъмът Random Forest, също със стойност 0.9812, следван от Support Vector Machines и накрая от K-NN. Това се дължи на получените резултати за избраните метрики. За останалите метрики, като изчерпаемост и F-1 оценка, най-добри резултати се получават от алгоритъма Random Forest (Таблица 1.2).

Като недостатък може да се отбележи обема на използвания набор от данни, който не е достатъчно голям. Използвайки по-голям обем данни, би било възможно да се оцени разликата при тестване на различни по обем данни с подобни характеристики. Въпреки това, всеки опит за противодействие на зловредния софтуер би спомогнало за запазване на данните на потребителите в ерата на дигиталната общност. А това е предпоставка за постигане на висока сигурност в комуникациите и по-добра икономическа устойчивост.

За да се защити огромното количество данни както на компаниите и техните клиенти, така и на обикновените потребители, е необходимо да се предприемат навременни мерки. Това е наложително, тъй като прогнозите сочат за увеличаване на кибер инцидентите. Един от основните елементи на защитата срещу подобни инциденти е разработването на усъвършенствани алгоритми за навременно откриване на зловреден софтуер. Следователно, тестването на приложимостта на алгоритми за двоична класификация за откриване на зловреден софтуер представлява интерес. За провеждането на тестовете, използвайки публичен набор от данни, заразен с 9 вида зловреден софтуер, предложена блок-схема на методологията, чрез която се определя производителността на алгоритмите. Следвайки тази методология и определените показатели, резултатите показват приложимост на всичко изследвани алгоритми, като един от тях демонстрира по-добра производителност.

## 1.3. Анализ на производителността на алгоритми за откриване на зловреден софтуер в дейта сет от областта на интернет на нещата

Мрежовата сигурност трябва да ограничава външния достъп по такъв начин, че да гарантира поверителността и целостта на данните и ресурсите. Днес, съществува постоянна заплаха от кибератаки поради разнообразието от приложения на IoT (Internet of Things – Интернет на нещата) устройствата в здравеопазването (Almotairi, 2023), дигиталната икономика (Khang et al., 2024), интелигентния град (Alahi et al., 2023) морската индустрия (Aslam et al., 2023), селското стопанство (Smmarwar et al., 2022) и др. IoT устройствата се използват не само в компанията, в която работим, но и у дома (Cvitić et al., 2023,). Едно от критичните предизвикателства, свързани с проникването на зловреден софтуер, е получаването на неоторизиран достъп до IoT устройства, където зловредният софтуер се опитва да копира оторизирани устройства, като имитира техните хардуерни и софтуерни спецификации (Praveen et al., 2023). Веднъж попаднал в мрежата, зловредният софтуер може да зарази цялата мрежа и да остане неактивен в продължение на дни или седмици. Най-добрите продукти за сигурност не само сканират за зловреден софтуер при влизане в системата, но и непрекъснато наблюдават файловете след това, за да откриват аномалии и да елиминират зловредния софтуер.

Компаниите полагат усилия да се защитят от мрежови заплахи, но с появата на нов зловреден софтуер това е постоянно предизвикателство. За да се гарантира надеждна сигурност, е необходимо да се използват както различни хардуерни, така и софтуерни решения. Поради тези причини е важно да се предприемат защитни мерки, най-често срещаната от които е инсталирането на антивирусен софтуер.

Чрез използването на различни алгоритми за машинно обучение може да се увеличи способността за откриване на зловреден софтуер, което е предпоставка за подобряване на цялостната киберсигурност. Машинно обучение се фокусира върху разработването на алгоритми и модели, които позволяват на компютрите да се учат от данни и да подобряват производителността без изрично програмиране. Изискват се големи количества данни, от които алгоритмите могат да извличат информация и модели. Моделите се обучават с помощта на исторически данни. По този начин обучените модели могат да предсказват или класифицират нови данни въз основа на това, което са научили. С течение на времето, с появата на нови данни, моделите могат

да бъдат адаптирани и подобрени. Преглед на техниките и алгоритмите за откриване на зловреден софтуер може да се намери в (Aslan & Samet, 2020). Тези техники за анализ на зловреден софтуер могат да бъдат определени като статични, динамични, хибридни и анализ на паметта (Sihwail et al., 2018). Две от тях са основни техники за анализ на зловреден софтуер – статичен и динамичен. Статичният анализ има за цел да изследва кода и структурата на зловреден софтуер, без да се изпълнява кодът, докато динамичният анализ изследва поведението по време на изпълнение на код (Raghuraman et al., 2020). Много изследователи се опитват да предложат различни техники за справяне с предизвикателствата при откриване и анализ на зловреден софтуер (Baptista et al., 2019; Nobakht et al., 2024). Например, авторите предлагат рамка за откриване на зловреден софтуер, комбинираща дълбоко обучение и машинно обучение (Shaukat et al., 2023). Ефективността на откриването на зловреден софтуер зависи от това колко ефективно се извличат отличителните характеристики на зловредния софтуер чрез техники за анализ. Съществуват различни методи за анализ, използващи различни статични и динамични инструменти, както е показано в (Singh & Singh, 2021; Gopinath & Sethuraman, 2023; Ling et al., 2023).

Имайки предвид трудността, свързана с откриването на зловреден софтуер, от особена важност е анализа на алгоритмите по отношение на няхната ефективност в дейта сет от областта на интернет на нещата. В тази връзка, в следващите няколко раздела са представине накратко модели, които са обект на анализ по отншение на тяхната ефективност.

### 1.3.1. Long Shorm-Term Memory Network

Мрежите с дълга краткосрочна памет (Long Shorm-Term Memory Networks – LSTM) могат да бъдат представени като последователна невронна мрежа със способността да съхранява информацията за произволни интервали от време. LSTM е дълбока невронна мрежа, способна да се справя с информация от времеви редове, подходяща за прогнозиране на дългосрочни нелинейни редове (Hochreiter & Schmidhuber, 1997). Моделът за прогнозиране, базиран на LSTM, извлича нелинейни и динамични характеристики на данните от процеса, за да постигне задоволителна прогнозна производителност.

### 1.3.2. Support Vector Machines

Support Vector Machines като набор от методи за контролирано обучение, могат да анализират данни за прогнозиране и класификация. SVM имат за цел да намери такава хиперплоскост в N-мерно пространство, която ясно може да класифицира точките от данните (Suthaharan, 2016).

### 1.3.3. Convolutional Neural Network

Convolutional Neural Network като вид алгоритъм за дълбоко обучение често се използва за анализ на визуални изображения. Основната идея на CNN е да се използва серия от конволюционни слоеве за извличане на характеристики от входните данни, (Gu et al., 2018). CNN използва специална техника, известна като „конволюция“, и разчита на матрични умножения, за да комбинира две функции, за да покаже как едната променя формата на другата.

### 1.3.4. CNN-LSTM

Чрез хибридния метод за дълбоко обучение CNN-LSTM, който комбинира CNN и LSTM, се цели да се подобри точността на прогнозиране (Shoorkand et al., 2024).

### 1.3.5. Методолигия на проведените експерименти

Използва се структурирана методология за анализ и класифициране на данни за мрежовия трафик от извадка от набора от данни IoT-23, използвайки различни модели за машинно обучение, включително LSTM, SVM, CNN и CNN-LSTM. Наборът от данни IoT-23 включва както доброкачествени, така и злонамерени мрежови дейности, което го прави надежден бенчмарк за оценка на ефективността на подходите за машинно обучение при откриване на зловреден софтуер в IoT. Използваната методология включва 11 основни стъпки. Този систематичен подход осигурява цялостно разбиране на данните, оптимален избор на модел и точна оценка на производителността. Всяка стъпка е описана подробно по-долу, с акцент върху процесите и срещаните предизвикателства.

1) *Описание на проблема.* Целта е да се идентифицират и класифицират злонамерени дейности в трафика на мрежата на Интернет на нещата, използвайки набора от данни IoT-23. Тази стъпка подготвя почвата за целия

експеримент, като очертава целите, обхвата и желаните резултати. Точното формулиране на проблема може да бъде предизвикателство, тъй като изисква цялостно разбиране на контекста на набора от данни, потенциалните заплахи и очакваните резултати.

2) *Събиране и придобиване на данни.* Тази стъпка включва събиране на набора от данни IoT-23, който съдържа данни за мрежовия трафик, включително както доброкачествени, така и злонамерени дейности. Осигуряването на целостта и пълнотата на данните може да бъде проблем. Освен това, получаването на достъп до изчерпателни и добре обозначени данни за трафика на IoT може да бъде отнемащо време и сложно.

3) *Пречистване на данни.* Пречистването на данни се фокусира върху премахване на шум, обработка на липсващи стойности и адресиране на несъответствия в набора от данни, за да се подобри качеството на данните. Това гарантира, че наборът от данни е подходящ за анализ и моделиране. Наборът от данни на IoT-23 може да съдържа повредени данни, дублирани записи или липсващи стойности, които трябва да бъдат адресирани. Правилното пречистване е от решаващо значение, тъй като пренебрегнатите грешки могат да повлияят негативно на производителността на модела.

4) *Проучвателен анализ на данни.* Извършва се, за да се разбере основната структура и разпределение на данните. Техники като визуализация на данни, описателна статистика и анализ на корелацията на характеристиките помагат за идентифициране на модели и потенциални аномалии. Анализирането на сложни данни за мрежовия трафик с голям брой характеристики може да бъде обезсърчително. Идентифицирането на смислени модели при наличие на шум и отклонения може да изисква значителни усилия.

5) *Предварителна обработка на данни.* Предварителната обработка включва трансформиране на данните във формат, подходящ за машинно обучение. Стъпките могат да включват кодиране на характеристики, нормализиране и намаляване на размерността. Изборът на правилните техники за предварителна обработка на данни за мрежовия трафик може да бъде

труден. Например, запазването на важни темпорални връзки, като същевременно се намалява шумът, изисква внимателно обмисляне.

6) *Избор на характеристики.* Изборът на характеристики се фокусира върху идентифицирането на най-подходящите характеристики, които допринасят за предсказващата сила на модела. Това намалява размерността, подобрява интерпретируемостта на модела и минимизира пренапасването. Изборът на оптимално подмножество от характеристики не е лесен, тъй като включва балансиране между релевантността и излишъка на характеристиките. Пренебрегването на критични характеристики може да влоши производителността на модела.

7) *Избор на алгоритъм за машинно обучение.* В тази стъпка се избират подходящи алгоритми за машинно обучение за задачата за класификация. В конкретното проучване бяха избрани LSTM, SVM, CNN и CNN-LSTM, за да се уловят различни характеристики и модели на данните в набора от данни IoT-23. Изборът на най-добрия алгоритъм изисква задълбочено разбиране на набора от данни и силните страни на всеки модел. Гарантирането, че избраните модели съответстват на характеристиките на данните, е от решаващо значение за оптимална производителност.

8) *Обучение на модела.* Избраните модели се обучават с помощта на предварително обработен набор от данни. Това включва въвеждане на данни в моделите, настройване на хиперпараметри и минимизиране на функциите за загуба чрез итеративна оптимизация. Обучението на сложни модели като CNN-LSTM може да бъде изчислително интензивно и да изисква значително време и ресурси. Осигуряването на конвергенция и избягването на пренареждане са ключови предизвикателства по време на тази стъпка.

9) *Оценка на модела.* Обучените модели се оценяват с помощта на показатели за ефективност, като например точност, прецизност, извикване и F1-оценка, за да се оцени тяхната ефективност при класифицирането на доброкачествен и злонамерен мрежов трафик. Оценяването на модели върху сложни данни, като например набора от данни IoT-23, може да бъде предизвикателство

поради потенциален дисбаланс в данните и необходимостта от надеждни техники за валидиране.

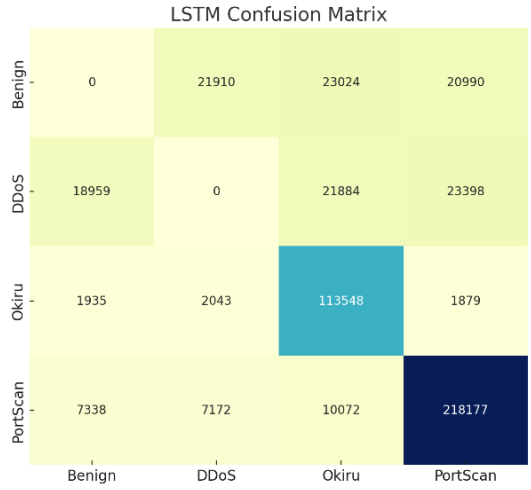
10) *Тестване на модела.* Моделите се тестват върху невидими данни, за да се оценят техните възможности за обобщение и реална производителност. Тестването върху нови данни може да разкрие проблеми като пристрастност на модела, свръхадаптиране или липса на обобщение. Осигуряването на ефективното пренасяне на производителността на модела в различни среди е от решаващо значение.

11) *Документиране.* Последната стъпка включва документиране на целия процес, включително стъпките за предварителна обработка на данните, конфигурациите на модела, резултатите и всички предизвикателства, възникнали по време на експеримента. Изчерпателната документация изисква внимание към детайлите и ясна комуникация, за да се гарантира възпроизводимост и прозрачност. Тя може да отнеме много време, но е от съществено значение за съвместните и бъдещи изследователски усилия.

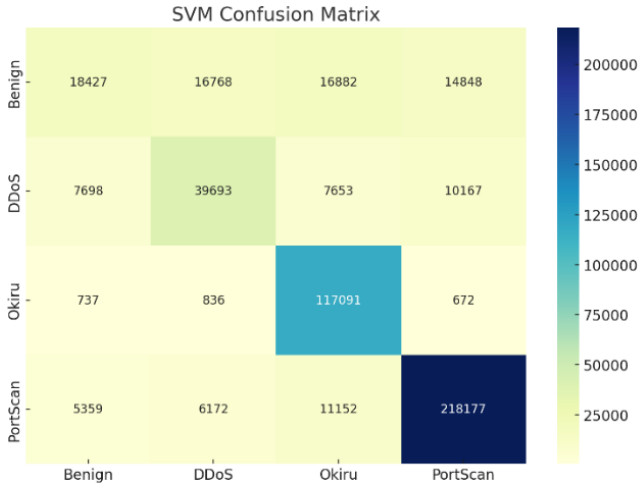
Тази методология предоставя структуриран подход за анализ на данните за трафика на мрежата на IoT, използвайки различни модели за машинно обучение. Въпреки че всяка стъпка представлява уникални предизвикателства, те заедно осигуряват строг и прозрачен работен процес, водещ до смислени прозрения за заплахите за сигурността на IoT. Комбинацията от LSTM, SVM, CNN и CNN-LSTM предлага разнообразен набор от възможности за улавяне на различни модели на данни и подобряване на откриването и класифицирането на злонамерени дейности.

**Анализ на резултатите**

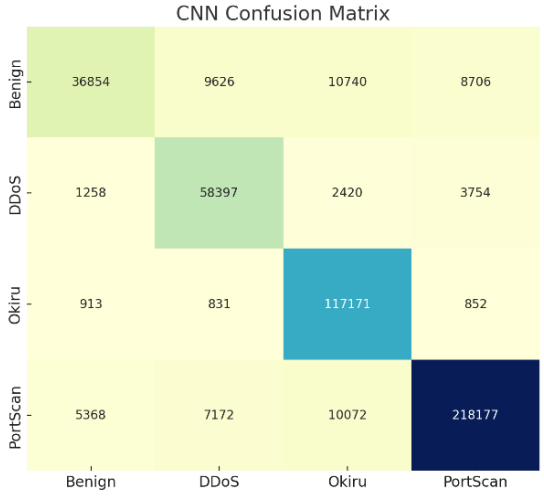
За да се определи производителността на всеки алгоритъм за класификация, използван в настоящото изследване, съответните матрици на объркване, които визуализират и обобщават производителността на алгоритмите за класификация, са показани на фигурите от 1.4 до 1.7.



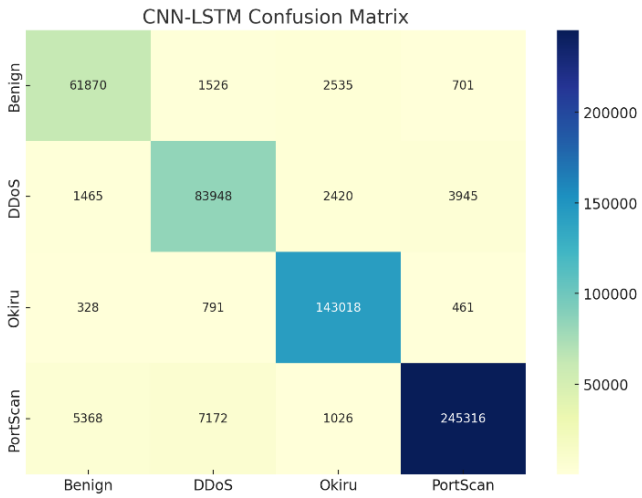

*Фиг. 1.4. Матрица на объркване на LSTM.*




*Фиг. 1.5. Матрица на объркване на SVM.*

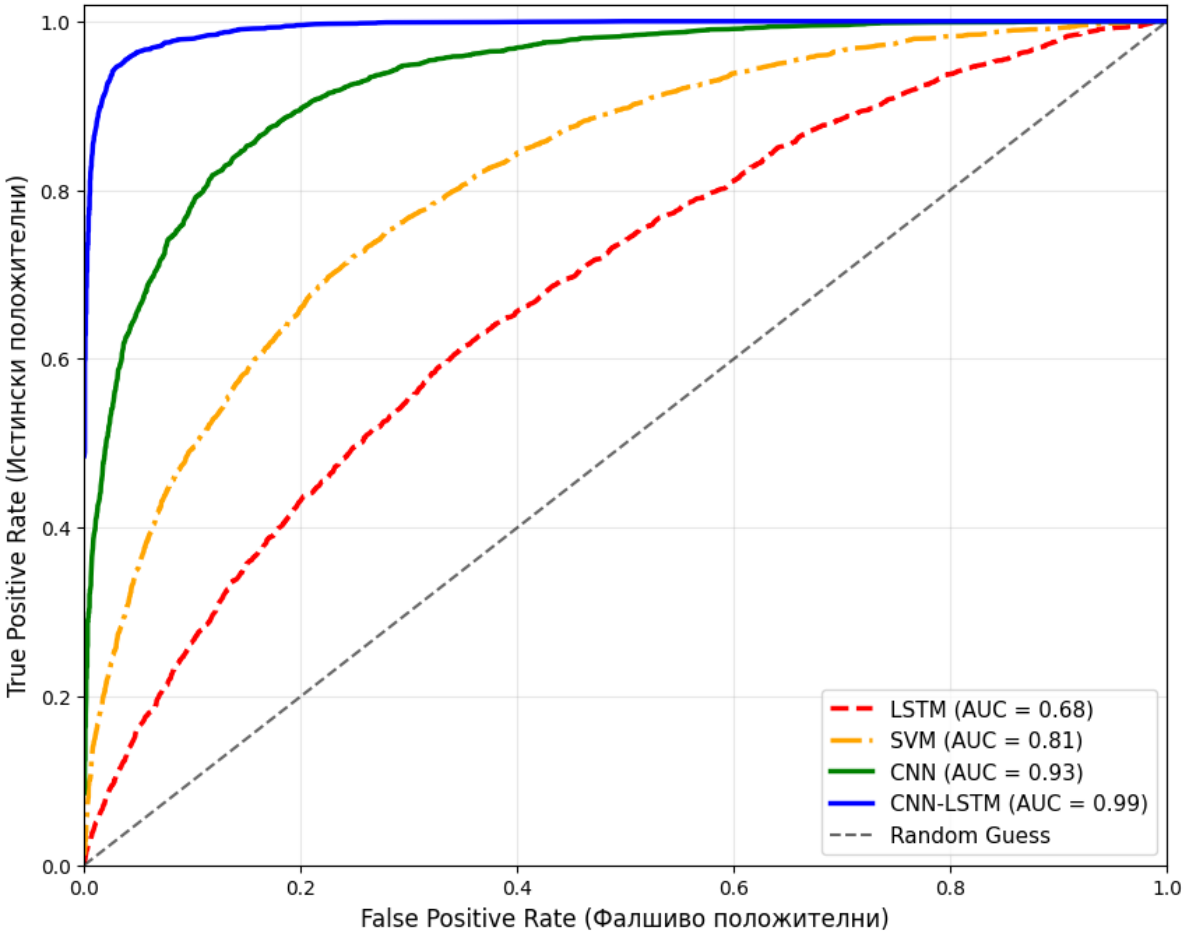



*Фиг. 1.6. Матрица на объркване на CNN.*




*Фиг. 1.7. Матрица на объркване на CNN-LSTM.*

За да се оцени производителността на разгледаните по-горе четири алгоритъма за класификация LSTM, SVM, CNN и CNN-LSTM, е използван анализ на кривата на оперативните характеристики на приемника (Receiver Operating Characteristic – ROC). ROC кривите графично илюстрират дискриминационните способности на всеки модел чрез отразяване на честотата на истински положителни резултати (True Positive Rate - TPR) спрямо честотата на фалшиво положителни резултати (FP) за различни прагови стойности, както е показано на Фиг. 1.8.




*Фиг. 1.8. ROC криви за LSTM, SVM, CNN, CNN-LSTM.*

Чрез сравняването на ROC кривите на всеки модел може да се оценят компромисите между чувствителност и специфичност. Този анализ дава представа за способността на всеки алгоритъм правилно да идентифицира положителни случаи, като същевременно минимизира фалшиво положителните резултати, предлагайки цялостен поглед върху ефективността на класификацията при различни прагове.

По-високата крива с по-голяма AUC предполага по-добра производителност при класифицирането на случаи. Формата и разположението на всяка крива отразяват компромисите между правилното идентифициране на положителни случаи и избягването на фалшиви положителни резултати.

За да се оцени цялостно класификационната ефективност на четирите модела – LSTM, SVM, CNN и CNN-LSTM – се анализираха няколко ключови показателя, включително точност, прецизност, попълнота на повторение и F1 резултат, както е показано в Таблица 1.3.

*Таблица 1.3. Резултати от производителността на алгоритмите*

| Модел | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| **LSTM** | 0.74 | 0.72 | 0.74 | 0.71 |
| **SVM** | 0.84 | 0.86 | 0.84 | 0.84 |
| **CNN** | 0.91 | 0.92 | 0.91 | 0.91 |
| **CNN-LSTM** | 0.97 | 0.97 | 0.97 | 0.97 |

Тези показатели за производителност предоставят подробен преглед на способността на всеки модел правилно да класифицира случаите, да минимизира фалшиво положителните и отрицателните резултати и да балансира между прецизност и изчерпаемост.

Моделът LSTM демонстрира скромна обща производителност с точност от 0,74 и подобни стойности за прецизност, попълнота и F1-Score. Въпреки че възможностите за последователно моделиране на LSTM му позволяват да улавя времеви модели, присъщи на данните за мрежовия трафик, неговата производителност беше ограничена по отношение на прецизност и чувствителност в сравнение с други модели. Това може да се дължи на трудности при разграничаването между определени класове злонамерен софтуер за IoT в набора от данни.

SVM постигна по-висока точност от 0,84, със стойности за прецизност, попълнота и F1-Score от 0,84 до 0,86. Способността на SVM модела да разделя класове с помощта на хиперплоскост работи ефективно за набора от данни IoT-23. Производителността му обаче е ограничена от липсата на възможности за дълбоко извличане на характеристики, което го прави по-малко подходящ за силно сложни взаимовръзки в данните за мрежовия трафик.

Моделът CNN допълнително подобри производителността, постигайки точност от 0,91 и постоянно висока прецизност, попълнота и F1-Score. Способността на CNN да извлича пространствени характеристики от моделите на мрежовия трафик му позволи да идентифицира сложни характеристики в набора от данни IoT-23, предоставяйки силни резултати от класификацията за откриване на зловреден софтуер.

Хибридният модел CNN-LSTM значително превъзхожда другите подходи, постигайки точност от 0,97 и също толкова високи стойности за прецизност, попълнота и F1-Score. Този хибриден модел комбинира силните страни на възможностите за извличане на характеристики на CNN със способността на LSTM да улавя времеви

зависимости, което го прави особено ефективен за данни от IoT-23. Този набор от данни често показва както пространствени, така и времеви модели в поведението на мрежовия трафик, а моделът CNN-LSTM се отличава с улавянето на тези сложни взаимодействия.

Въпреки че всички модели показват различна степен на ефикасност при класифицирането на данни от набора от данни IoT-23, хибридният подход CNN-LSTM се откроява благодарение на изключителната си точност и балансирана производителност по всички показатели. Комбинацията от извличане на характеристики и възможности за последователно обучение го прави изключително ефективен за анализ на сложни данни за трафика на мрежата на IoT за откриване на зловреден софтуер. Въпреки това, при внедряването в реалния свят трябва да се вземат предвид практическите съображения относно изчислителните изисквания и използването на ресурси.

## 1.3. Изводи

Машинното обучение като област от изкуствения интелект, използва алгоритми, за да се учи от данни, да идентифицира модели и да прави прогнози или решения с минимална човешка намеса. Следователно, могат да се направят някои конкретни изводи от проведените анализи и сравнения, а именно.

- Съществува разнообразие от модели, базирани на машинно обучение, които са способни за откриване на злонамерен софтуер;
- Важен етап от прилагането на машинно обучение и по-специално на контролираното обучение е процесът на самото обучение, при което се цели обучаване с данни за конкретна предметна област;
- Комбинирането на различни модели на машинното обучение може да допринесе за преодоляване недостатъците на самостоятелно използваните модели.

Следователно, изброените по-горе предпоставки са основа за разработване на подобрени модели, рамки и приложения, водещи до по-добри резултати при откриването и класифицирането на зловреден софтуер.

## 1.4. Цел и задачи

Цел на дисертационният труд е да се изследват и анализират възможностите за откриване на злонамерен софтуер чрез средствата на машинно обучение, на база на които да се предложат подходящи хибридни модели, рамки и приложения за получаване на по-добри резултати при откриването на зловреден софтуер. За постигането на тази цел е необходимо да се изпълнят следните задачи:

1) да се направи анализ на различни алгоритми на машинното обучение относно тяхното представянето за целите на откриване на зловреден софтуер;
2) да се определи подходяща виртулна машина, която да се използва при провеждане на тестове за откриване и класификация на зловреден софтуер;
3) да се предложи подобрен подход за статичен анализ за откриване на зловреден софтуер чрез оптимизиране извличането на характеристики и комбинирайки различни алгоритми за машинно обучение;
4) да се предложи рамка за статична класификация на зловреден софтуер, използваща оптимизация на функции и ансамблово обучение;
5) да се предложи самоосъзната класификация на зловреден софтуер чрез рутиране на модели на базата на система за доверие за избора и обяснимост на характеристиките;
6) да се предложи адаптивна рамка, съобразена с доверието, за класификация на зловреден софтуер с корекции за обратна връзка.

# ГЛАВА 2. МОДЕЛИ ЗА ИЗБОР НА ВИРТУАЛНА МАШИНА, ХИБРИДНИ АЛГОРИТМИ И РАМКИ ЗА ОТКРИВАНЕ НА ЗЛОНАМЕРЕН СОФТУЕР ЧРЕЗ МЕТОДИТЕ НА МАШИННОТО ОБУЧЕНИЕ

В тази част от дисертацията е описан предложен модел за избор на виртуална машина за целите на провеждането на експерименти за откриване на зловреден софтуер. Представен е подобрен подход на статичен анализ чрез оптимизиране извличането на характеристики, комбинирайки различни алгоритми за машинно обучение за откриване на зловреден софтуер. В тази глава е описан и предложената рамка за статична класификация на зловреден софтуер, използваща оптимизация на функции и ансамблово обучение. За постигане на по-добро откриване на зловреден софтуер е предложена самоосъзната класификация, използваща рутирането на модели на базата на система за доверие за избор и обяснимост на характеристиките. Представена е и адаптивна рамка, интегрираща доверие за класификация на зловреден софтуер позволяваща корекции чрез обратна връзка. Тя интегрира механизъм за рутиране, съобразен с доверието, който използва множество поколения модели, включително резервен механизъм. В нея се използва буфер за обратна връзка и слой за корекции за лека адаптация от потребителски корекции и синтетични инжекции. Благодарение на комбинацията от адаптивност, интерпретируемост и мащабируемост се решават предизвикателствата, свързани с непрекъснатото разработване на системи за откриване на зловреден софтуер от следващо поколение, които постоянно остават „верни“ за дадените модели на заплахи, вместо да се променят постоянно.

## 2.1. Избор на софтуер за виртуални машини за целите на откриване на зловреден софтуер

Дигиталната трансформация, ускорена от Covid-19 пандемията, промени не само използваните технологии, но и начина по който приложенията се проектират, внедряват и поддържат. Чрез използване на данни от различни източници и подходящ анализ е възможно да се постигне интелигентно вземане на решения, което да

рационализира управлението на ресурсите, да се подобри оперативната ефективност, да се оптимизира разработването на продукти и да се генерират нови приходи (Borissova et al, 2020). При обработката на данни е наложително да се използват не само различни методи и техники, но и принципи, които отчитат неопределеността, изразена по подходящ начин (Borissova & Dimitrova, 2021).

Водена от новите технологии, дигиталната трансформация прави възможно подпомагането на различни бизнес дейности в различни области чрез електронна търговия (Dimitrova et al., 2023). Наред с предимствата, заслужава да се споменат и някои уязвимости, свързани с киберсигурността. Справянето със злонамерен софтуер става все по-предизвикателно поради сложността на откриването му в големи файлове (Ucci et al., 2019). От друга страна, откриването на злонамерена активност е от съществено значение за подобряване на сигурността на облака и виртуалните машини (virtual machines (VM), но все още има проблеми с по-ниската точност при откриване на атаки, високия процент на фалшиви прогнози и грешки (Vaza et al., 2022). Виртуалните машини са се превърнали в неразделна част от изчислителната индустрия с приложения не само за бизнеса, но и с приложения в облачните изчисления. Основното предимство на работата с виртуална машина е възможността за стартиране на приложения, които не биха били достъпни поради много различни системни изисквания (Agache et al., 2020). Това е една от причините, поради които виртуализацията е станала толкова важна. Сложният зловреден софтуер може да е насочен към инфраструктура, базирана на виртуализация, и да навреди на виртуалните ресурси, превръщайки се в заплаха за индустриалните приложения и данните, хоствани в облака (Mishra et al., 2022).

Напоследък много автори се занимават с проблемите на откриването и анализа на зловреден софтуер в изпълними файлове и Android приложения, включително безфайлов зловреден софтуер (Melvin & Kathrine, 2021; Patil et al., 2020; Singh & Singh, 2021; Razgallah et al., 2021; Khalid et al., 2023). За симулиране на идеална среда, виртуална машина може да се използва за изучаване на това как извадка от зловреден софтуер взаимодейства с всичко – от файловата система до системния регистър. Възможността за симулиране на множество екземпляри на операционна система на една физическа машина прави виртуализацията изключително полезна за анализ, базиран на поведение. Използването на виртуални машини позволява лесно създаване

на идеални условия за тестване, като се има предвид гъвкавият характер на виртуализацията, която е подходяща за тестване на всякакви приложения, включително зловреден софтуер. Възможно е да се разпредели цялата виртуална RAM памет, място за съхранение и процесорна мощност, необходими на тестовата среда. Благодарение на независимата от платформи виртуална машина, анализът на зловреден софтуер в множество системи ще даде представа за това как той взаимодейства с различни операционни системи. Виртуализацията прави възможно създаването на няколко виртуални системи, като например Windows, Linux и Mac OS X, като по този начин се елиминира необходимостта от обемисти физически машини. Използването на виртуална машина позволява провеждането на тестови дейности вътре в една виртуална машина, като по този начин се защитава останалата част от системата от повреда.

За да се защитят авангардни технологии като Интернет на нещата (Sadhu at al, 2022), киберфизически системи (Tsochev & Yoshinov, 2020) и системи за управление на бизнеса (Stankov & Tsochev, 2020) от различни кибератаки, механизмите за разпространение на зловреден софтуер трябва да бъдат добре проучени. В тази дръзка, автори са направили цялостно проучване на проблемите, предизвикателствата и бъдещите насоки за откриване на зловреден софтуер (Aboaoja et al., 2022). За защита от атаки на зловреден софтуер в облака са предложени различни подходи, базирани на виртуализация, като повечето от тях са базирани на техники за дълбоко обучение (deep learning) (Gopinath & Sethuraman, 2023). Други подходи разчитат на техники за машинно обучение (Gibert et al., 2020; Liu et al., 2020), генетичен алгоритъм (GA) (Vivekanandam, 2021), задача за двустепенна оптимизация (Jerbi et al., 2022), комбинация от дълбоко обучение с хибриден BP-PSO (Al-Andoli et al., 2022) и др. За да се справят с тези проблеми, ново решение е предложено за откриване на зловреден софтуер за виртуализационни среди в облака, използващо хардуерно проследяване и дълбоко обучение (Tian et al., 2021).

Като се има предвид важността на виртуалната машина не само за бизнес цели, но и за откриване на зловреден софтуер, в настоящото дисертационно изследване са предложени два модела за групово вземане на решения за избор на виртуална машина. Отличителна черта на предложените подходи е фактът, че и двата модела могат да използват различни оценки. Например, оценки от вече дадени отзиви или

нарочно определени оценки за всеки от критериите, участващи в избора. Освен това, група експерти участва в процеса на избор и влияе върху формирането на окончателното групово решение.

### 2.1.1 Модели за групово вземане на решение при избор на софтуер за виртуална машина

Конкретният разглеждан проблем се отнася се до избор на виртуална машина, която да работи на десктоп с Windows. В тази връзка е необходимо да се определи най-подходящия софтуер за виртуализация спрямо някои критерии. Като достатъчно общи показатели могат да се използват критериите „лекота на използване“, „обслужване на клиенти“ и „съотношение цена-качество“, които могат лесно да бъдат разбрани. Въз основа на дадени оценки е възможно да се направи подходящ избор. За да бъде изборът прозрачен и убедителен, в този раздел са предложени два оптимизационни модела за избор. Тези модели за групово вземане на решения разчитат на адекватното мнение на компетентни и оторизирани лица, вземащи решения.

Описаните по-горе четири критерия за оценка се използват за избор на софтуер за виртуални машини за целите на тестване откриването на зловреден софтуер. Всички тези критерии, заедно с вече дадените оценки, се използват в следния предложен модел за групово вземане на решения (Borissova et al., 2023):

$$\max\{A_j^*\} = \sum_{q=1}^{Q} \lambda^q \sum_{i=1}^{I} w_i \, e_i^q \tag{2.1}$$

$$\sum_{i=1}^{I} w_i = 1 \tag{2.2}$$

$$\sum_{q=1}^{Q} \lambda^q = 1 \tag{2.3}$$

където индекс $j$ е използван за обозначаване на набор от дадените алтернативи ($j = 1, \dots, J$), с индекс $i$ е означено множеството на критериите за оценка ($i = 1, \dots, I$), а индекс $q$ е използван за означаване на множеството на лицата, вземащи решение ($q = 1, \dots, Q$) които ще формират крайното групово решение. Коефициентите $w_i$ изразявят важността на критериите за оценка и трябва да спазват отношението (2.2). Другият вид коефициенти $e_i^q$ изразяват оценките на *i-ия* критерий съгласно гледната точка на *q-ия* експерт. В зависимост от компетентността на всеки експерт за всеки се задават съответни тегла $\lambda^q$, които ще участват във формирането на крайното групово решение.

Трябва да се отбележи, че използваните в (2.1) мерни единици трябва да бъдат изразени в сравнима скала. Това означава, че оценките $e_i^q$ и теглата $\lambda^q$ трябва да бъдат в интервала между 0 и 1, като по-голямата стойност означава по-добро представяне.

Възможно е също така, вместо избор на един единствен тип софтуер за виртуална машина, даденият набор от алтернативи да се редуцира до подмножество, а изборът да се извърши по други критерии. Например, може да се направи последващо класиране, използвайки специфичните характеристики. В тази ситуация може да се използва следният комбинаторен оптимизационен модел (Borissova et al., 2023):

$$max\{\sum_{j=1}^{J} x_j \sum_{q=1}^{Q} \lambda^q \sum_{i=1}^{I} w_i \, e_i^q\} \tag{2.4}$$

При ограничения

$$\sum_{j=1}^{J} x_j = C, x_j \in \{0,1\} \tag{2.5}$$

$$x_j \in \{0,1\} - binary\ integers \tag{2.6}$$

$$C < J \tag{2.7}$$

$$\sum_{i=1}^{I} w_i = 1 \tag{2.8}$$

$$\sum_{q=1}^{Q} \lambda^q = 1 \tag{2.9}$$

Използваните двоични променливи $x_j$ са свързани с всяка алтернатива, а константата $C$ изразява броя на алтернативите, до които се стремим да редуцираме даденото множество алтернативи. Очевидно е, че тази константа трябва да е по-малка от броя на алтернативите, изразени с (2.7). Целевата функция (2.4) ще избере точно тези алтернативи с по-добри резултати, съобразно оценките $e_i^q$.

## 2.2. Подобрен подход на статичен анализ чрез оптимизиране извличането на характеристики, комбинирайки различни алгоритми за машинно обучение за откриване на зловреден софтуер

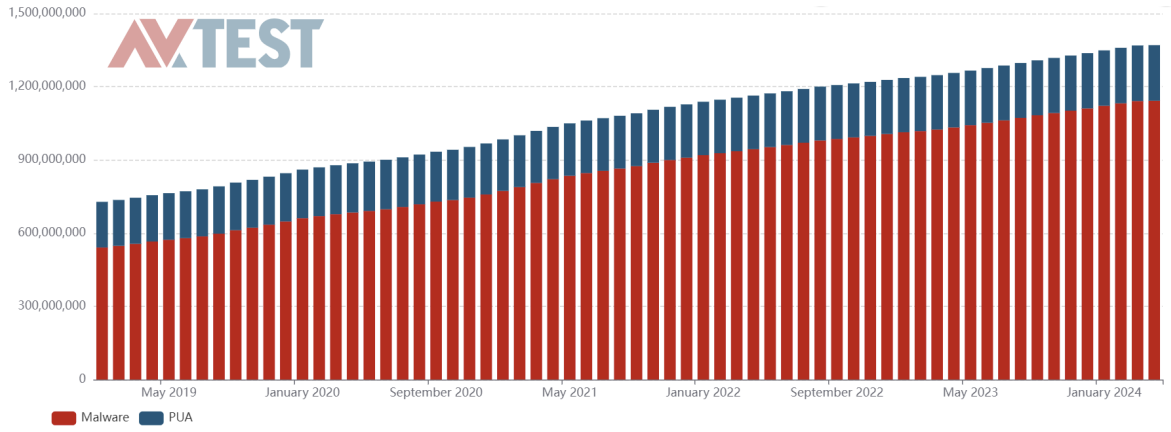
Статичният анализ на зловреден софтуер, като проактивен подход, включва анализ на характеристиките и компонентите на подозрителен файл, без той да се изпълнява. Този метод предоставя информация за структурата, поведението и потенциалните рискове на файла, като анализира различни атрибути, като заглавка на файла, метаданни, низове, ресурси и код. Този безопасен статичен подход се отличава с бързо сканиране

и маркиране на често срещани заплахи, което го прави идеален за антивирусен софтуер и първоначално сортиране. Следователно, статичната класификация може да се разглежда като полезен инструмент за прогнозиране на резултата от откриването на зловреден софтуер. Като се има предвид пример за зловреден софтуер, който не се наблюдава в обучаващия набор от данни със същия набор от атрибути на характеристиките, с изключение на набора за прогнозиране, алгоритъмът няма да може да предскаже правилния клас. Точността на прогнозирането определя колко добър е алгоритъмът.

Зловредният софтуер е бързо нарастваща заплаха за различни услуги като социални мрежи и облачно съхранение, тъй като бързо се размножава в интернет и понякога води до големи атаки, например атака от ботнет мрежа (Dutta и др., 2022). Той използва различни техники, за да прикрие присъствието си, което затруднява откриването му с помощта на конвенционални методи за сигурност (Mazurczyk & Caviglione, 2015). Докладът на Digital SME показва, че през 2023 г. е имало 57% скок в кибератаките в Европа и сериозно увеличение на атаките с ransomware от 112 през 2022 г. до 175 през 2023 г., последвани от фишинг кампания, проведена през този годишен период (Swascan, 2023). С нарастващата устойчивост на дигитализацията и интелигентните устройства, инцидентите със сигурността, включително неоторизиран достъп, zero-day експлойти, нарушения на данните, атаки за отказ на услуги (Denial of Service – DoS), социално инженерство и фишинг, са се увеличили експоненциално през последните години от 744 милиона през 2019 г. до 1,361 милиона през 2024 г., както е показано на Фиг. 2.1 [1].

Ето защо е необходимо да се внедрят подходящи технологии за киберсигурност (Blagoev, 2024). Извличането на модели или прозрения от данни за киберсигурност и изграждането на съответни модели, базирани на данни, са от решаващо значение за автоматизирането и подобряването на интелигентността на системите за сигурност. За да се разберат и анализират данните, се използват различни научни методи, техники за машинно обучение, процеси и системи.

[1] About malware and PUA, https://portal.av-atlas.org/malware, last access 2024/04/03




*Фиг. 2.1. Общо количество злонамерени и потенциално нежелани приложения.*

Интегрирането на машинно обучение, освен компонент на изкуствения интелект (ИИ), има потенциала да разкрие прозрения от данните (Mathews, 2019). За да се разграничи софтуерът за злонамерени приложения от злонамерени, могат да се използват два различни, но понякога допълващи се подхода: откриване на базата на сигнатури или/и аномалии. Решението, базирано на сигнатури, идентифицира злонамереното поведение като последващо събитие, защото генерира добре дефинирани модели (Wressnegger et al., 2017; Syrris & Geneiatakis, 2021). Тези модели са ядрото на двигателя за откриване, който наблюдава приложенията в реално време. Всеки от тези анализи се основава на различни характеристики от въпросния файл или система и може да варира от детайли на заглавката на файла в статичния анализ до холистични показатели за производителност на ниво операционна система в случай на онлайн анализ. Могат да се разграничат няколко подхода за анализ на зловреден софтуер, като например статичен (Patil, et al., 2020), динамичен (Singh & Singh, 2021) и онлайн анализ (Niu et al., 2022; Kimmell et al., 2021). Използването на специфичен анализ зависи от случая на употреба и наличието на данни.

Основно предизвикателство е мрежата на Интернет на нещата (IoT), която позволява на интелигентните устройства в организационна информационна система да се свързват и обменят данни с централно хранилище, създавайки огромен потенциал за всякакви атаки от зловреден софтуер. За да се справи с такъв проблем, е предложен модел на двупосочно-гейтирана рекурентна единично-конволюционна невронна мрежа (bidirectional-gated recurrent unit-convolutional neural network), базиран на дълбоко обучение (Deep Learning), за откриване на зловреден софтуер на IoT и класифициране на семействата зловреден софтуер на IoT, използвайки изпълними

формати на двоични файлови байтови последователности като входна характеристика (Chaganti et al., 2022). Операционната система Android е предпочитана в много IoT устройства, които се използват в различни области на ежедневието, като интелигентни домове и интелигентни градове, включително в интелигентния градски транспорт (Garvanov, & Garvanova, 2021). В тази връзка, в (Smmarwar et al., 2024) се съдържа цялостен преглед на подходите за откриване на зловреден софтуер на Android, групирани в три категории: техники за избор на характеристики за откриване на зловреден софтуер, техники, базирани на машинно обучение, и техники, базирани на DL, за откриване на зловреден софтуер. Друг преглед на литературата показва, че подходът на дълбоко обучение може да бъде обещаващо решение на проблема с откриването на различен зловреден софтуер (Aslan & Yilmaz, 2021). Подробно проучване на нововъзникващите алгоритми за машинно обучение за откриване на зловреден софтуер в бизнес информационни системи, задвижвани от Интернет на нещата, е представено в (Gaurav et al., 2023). Систематичен преглед на литературата и таксономия, текущи предизвикателства и бъдещи насоки на методите за машинно обучение за откриване на зловреден софтуер, които разглеждат тези проблеми чрез анализ на изследователски статии, свързани с откриването на зловреден софтуер с помощта на алгоритми за машинно обучение, са предложени в (Gorment et al., 2023).

Откриването на зловреден софтуер е труден въпрос поради недостатъци в точността на производителността, вида анализ и подходите за откриване на зловреден софтуер. Сравнителен анализ на производителността на алгоритмите за откриване на зловреден софтуер, базиран на различни текстурни характеристики и класификатори, може да бъде намерен в (Ahmed et al., 2024). Машинното обучение и обратното инженерство са вероятно най-добрите техники за справяне със зловреден софтуер и много изследователи работят върху този тип проблем (Ismael & Thanoon, 2022). Възможно е да се използват функции за обратно инженерство и алгоритми за машинно обучение, за да се открият уязвимости, присъстващи в приложенията за смартфони. Авторите предлагат модел, който включва по-иновативни набори от статични функции с най-големите текущи набори от данни за проби от зловреден софтуер, отколкото конвенционалните методи (Urooj et al., 2022).

Киберсигурността привлича много изследователи обединени от идеята за проектиране на ефективни модели за откриване на зловреден софтуер, базирани на

машинно обучение или дълбоко обучение. Подробен преглед на откриването на зловреден софтуер в Windows относно техники, изследователски проблеми и бъдещи насоки е представен в (Maniriho et al., 2024). За да се справят с такъв тип проблеми, виртуалните машини са подходящ инструмент за тестване на разработени алгоритми за откриване на зловреден софтуер (Borissova et al., 2023). Използвайки такива виртуални машини, някои алгоритми за двоична класификация за откриване на зловреден софтуер са сравнени по отношение на четири показателя: точност, прецизност, попълнота и F1-Score. Трябва да се отбележи, че получените резултати за всички алгоритми са окуражаващи, но Random Forest показа по-добра производителност срещу девет вида зловреден софтуер (Barzev et al., 2024).

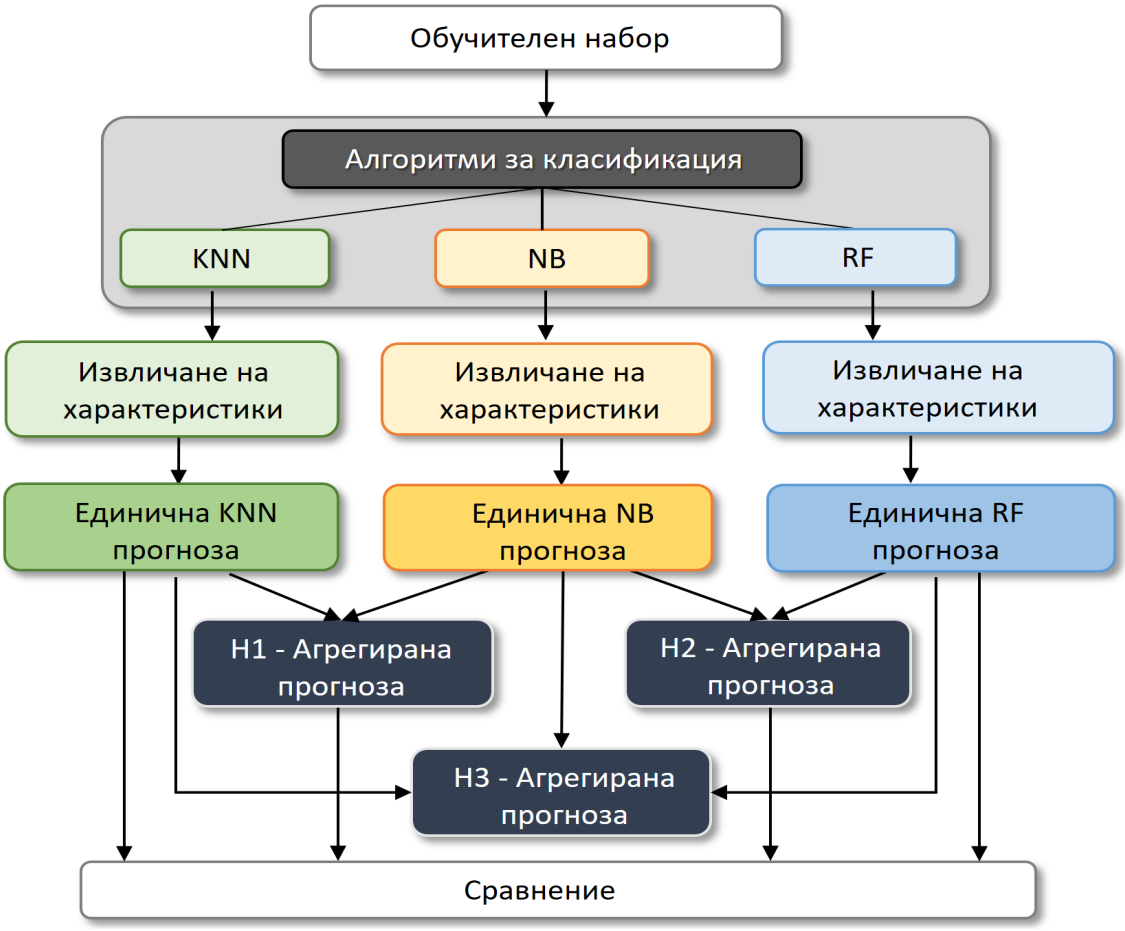
За да се подобри точността на статичния анализ за откриване на зловреден софтуер, в настоящото дисертационно изследване се предлагат три хибридни алгоритъма, базирани на комбинация от различни алгоритми за машинно обучение, за да се постигне по-добра точност чрез оптимизиране на извличането на характеристики. За да се провери приложимостта на тези алгоритми, се използва набор от данни IoT-23 като цялостна колекция от записи на мрежов трафик, специално проектирани за IoT среди. Този набор от данни се използва за обучение и оценка на предложените хибридни модели за машинно обучение, за да се направи по-точно разграничаване между доброкачествени и злонамерени дейности в IoT мрежите.

### 2.2.1. Подобрен подход за откриване на зловреден софтуер чрез комбиниране на различни алгоритми за машинно обучение

K-NN е доста интуитивен и прост и не изисква обучение (мързелив обучаващ алгоритъм), но е бавен, тъй като е изчислително интензивен и производителността може да бъде чувствителна към избора на разстояние, мащаба на характеристиките и неподходящите характеристики. Алгоритъмът RF може да обработва големи набори от данни, да обработва липсващи стойности и да предоставя информация за важността на характеристиките, но се нуждае от по-дълго време за обучение и е труден при интерпретирането на модела. NB алгоритъмът има ниска сложност на обучение и добра класификация, като може да обработва както дискретни, така и непрекъснати данни, но разчита на независимост между входните променливи. Следователно, за да се преодолее недостатъкът от използването само на един класификатор, е възможно те

да се комбинират в хибридни алгоритми. Методите NB, KNN и RF са избрани, защото използват различни подходи за прогнозиране на етикетите на класовете.

Обобщеното алгоритмично представяне на предложения подход за подобрено откриване на зловреден софтуер чрез комбиниране на различни алгоритми за машинно обучение в 3 хибридни алгоритъма е илюстрирано на Фиг. 2.2 (Barzev & Borissova, 2025).




*Фиг. 2.2. Обобщено алгоритмично представяне на предложения подход.*

В началната фаза се сравняват индивидуалните точности на класификаторите, а в следващата фаза – точността на формираните хибридни алгоритми:

1) H1 като комбинация на NB и KNN;

2) H2 като комбинация на NB и RF;

3) H3 като комбинация на NB, KNN и RF.

NB като директен вероятностен класификатор работи при предположението за силна независимост на признаците. Той разчита на теоремата на Бейс, за да изчисли априорната вероятност $P(H)$ и апостериорната вероятност $P(H|E)$ за събитие,

наблюдавано съответно преди и след доказателствата, и се изразява като правило на Бейс:

$$P\,(H|E)\ =\ (P(E|H)\ \mathrm{x}\ P(H))/P(E) \tag{2.10}$$

Псевдокодът за наивния алгоритъм на Бейс е следният:

```
Step 1: Training:
      For each feature
        For each feature-value E
            For each class-label H
            P(E/H) = total number of occurrences of feature value with
class label)/(total number of occurrences of class label)
            End for
        End for
      End for
Step 2: Testing:
      For each instance in test data
        Calculate probability using P(H/E)=(P(E/H)*P(H))/P(E)
      End for
Step 3: Assign the class label with the maximum probability to the test
instance.
```

По подобен начин могат да бъдат изразени алгоритмите KNN и RF

**Оптимизиране на извличането на характеристики**

Извличането на характеристики е критичен етап, от решаващо значение за повишаване на точността на предложения подход. Извличането на характеристики в машинното обучение се отнася до трансформиране на сурови данни в набор от характеристики, които могат да бъдат използвани за обучение на модел. Проучване относно избора на характеристики може да се намери в (Chandrashekar & Sahin, 2014; Rida et al., 2020). Характеристиките са специфични измерими свойства или характеристики на данните, свързани с текущата задача. Целта на извличането на характеристики е да представи данните по по-компактен, информативен и дискриминативен начин, улеснявайки процеса на обучение на алгоритмите за машинно обучение. Могат да се използват различни техники, в зависимост от естеството на данните и конкретния проблем, който се разглежда. Някои често срещани методи за извличане на характеристики включват:

- *Намаляване на размерността*: Това включва намаляване на броя на характеристиките чрез трансформиране на данните в по-нискоразмерно

пространство, като същевременно се запазва по-голямата част от важната информация. Анализът на главните компоненти (Principal Component Analysis – PCA), разлагането на сингулярни стойности (Singular Value Decomposition – SVD) и t-разпределеното стохастично вграждане на съседи (t-Distributed Stochastic Neighbor Embedding – t-SNE) са популярни техники за намаляване на размерността (Anowar et al., 2021).

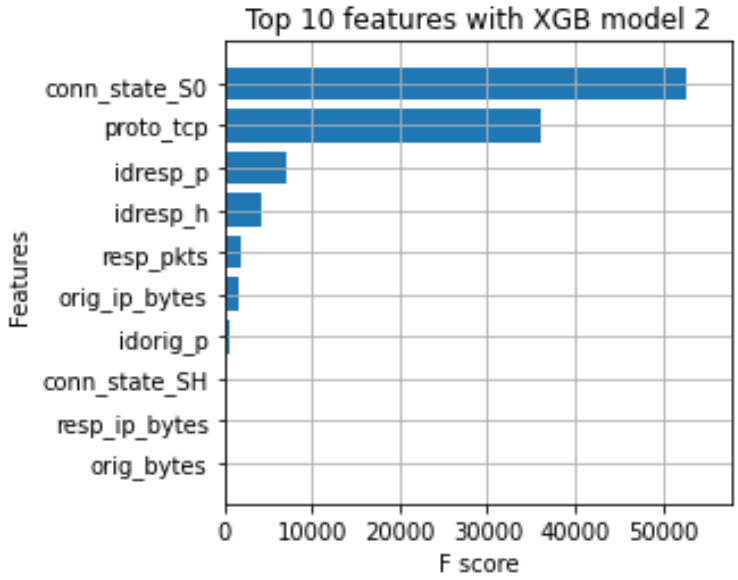
- *Избор на характеристики*: Вместо трансформиране на данните, изборът на характеристики включва избиране на подмножество от оригиналните характеристики, които са най-подходящи за задачата за прогнозиране. Това може да се направи с помощта на статистически методи, като например взаимна информация или оценки за важност на характеристиките от дървовидни модели, или чрез техники за регуларизация, като например ласо регресия (Lee et al., 2022).
- *Трансформация и мащабиране*: Техники за трансформация на данни, като нормализиране или стандартизация, могат да се приложат за мащабиране на характеристиките до подобен диапазон или разпределение. Това гарантира, че характеристиките с по-големи мащаби не доминират в процеса на обучение или не въвеждат отклонения в модела.
- *Инженеринг на характеристиките*: Това включва създаване на нови характеристики, базирани на знания за областта или прозрения за данните. Инженерството на характеристиките има за цел да улови релевантна информация, която може да не присъства в оригиналните сурови данни, потенциално подобрявайки производителността на моделите за машинно обучение (Mumuni & Mumuni, 2024).

Оптимизирането на извличането на характеристики включва намиране на най-ефективното представяне на данните, което максимизира производителността на модела за машинно обучение. Това обикновено изисква експериментиране и итеративно усъвършенстване, където се прилагат различни техники за извличане на характеристики и се оценяват с помощта на подходящи показатели за производителност. Процесът на оптимизация включва:

- *Разбиране на данните*: Анализиране на характеристиките и разпределението на данните, за да се определи кои техники за извличане на характеристики са най-подходящи.
- *Експериментиране*: Тестване на различни методи за извличане на характеристики, включително различни комбинации от техники, за да се види кои от тях дават най-добри резултати.
- *Валидация*: Оценка на производителността на модела за машинно обучение чрез кръстосана валидация или валидация с ограничение, за да се гарантира, че избраните характеристики се обобщават добре за невидими данни.
- *Итеративно усъвършенстване*: Фина настройка на процеса на извличане на характеристики въз основа на показателите за производителност и итеративно подобряване на производителността на модела.

Чрез оптимизиране на процеса на извличане на характеристики, моделите за машинно обучение могат да бъдат обучавани по-ефективно, което води до по-добра производителност и обобщение върху невиждани данни. Библиотеката Featurewiz на Python се използва за извличане на висококачествени характеристики от оригиналния набор от данни, благодарение на изключителната ѝ производителност в големи състезания по наука за данни.

Процедурата за избор на характеристики се състои от два етапа. Методът SULOV, изграден върху алгоритъма MRMR, първоначално се използва за идентифициране на променливи с висок взаимен информационен резултат и минимална корелация. Алгоритъмът SULOV определя характеристиките, които ще бъдат запазени, въз основа на техния взаимен информационен резултат (Mutual Information Score – MIS) с предварително определената цел. След това 3 от 27-те променливи се премахват в началото на алгоритъма, тъй като те са довели до променливи с ниска информация. Характеристиките се избират въз основа на по-висок MIS и по-ниска корелация с други характеристики. Впоследствие, използвайки характеристиките, получени от предишната стъпка, се прилага рекурсивен XGBoost за изграждане на модел и облекчаване на проблемите с недостатъчното напасване и свръхнапасването. Процесът на изпълнение на тази стъпка е илюстриран на Фиг. 2.3.




*Фиг. 2.3. Най-важните характеристики с XGB модел на целевия набор от данни за IoT-23.*

Накрая, Featurewiz генерира избраните характеристики, класирани по тяхната MIS спрямо предварително дефинираната цел. След процеса на избор на характеристики, броят на определените такива в двата набора от данни е съответно тринадесет и петнадесет. Крайният резултат съдържа списък със 7 важни характеристики, както е показано в Таблица 2.1.

*Таблица 2.3. Характеристики, използвани от XGB модела, с тяхното описание.*

| No | Характеристика | Описание |
|---|---|---|
| **1** | conn_state_50 | Connection state |
| **2** | proto_tcp | Transaction protocol |
| **3** | idresp_p | Destination port |
| **4** | idresp_h | Destination IP address |
| **5** | resp_pkts | Destination packets |
| **6** | orig_ip_bytes | Source2destination transaction bytes |
| **7** | Idorig_p | Source port |

**Агрегиране на прогнозираната точност**

Важно условие за постигане на по-добра точност при комбиниране на класификатори е използването на различни набори от характеристики или различни обучителни набори (Ho et al., 1994; Kittler et al., 1998). Съществена характеристика на многостепенния класификатор е фактът, че той може да стабилизира обучението на

класификатори въз основа на малък размер на извадката. Следователно, стратегиите за комбиниране на класификатори биха могли да смекчат грешките от равномерно разпределени класове или погрешни класификации в отделни класификатори, като по този начин увеличат точността. Агрегирането включва обединяване на вероятностите за класове, генерирани от един или повече класификатори. В настоящата система агрегирането се използва само когато NB не успява да класифицира точно или не се доверява на етикета на класа. Следователно, ползите от агрегирането се реализират само частично. Когато NB дава нестандартни резултати, съществуващият метод се затруднява да подобри значително резултатите.

В предложения подход, се стремим да се справим с недостатъците на настоящата система, като агрегираме NB и KNN по различен начин, наред с RF. За целта използваме средната стойност на вероятностите за класове като метод за агрегиране. Нашите експерименти се провеждат върху набора от данни IoT-23, реализиран с помощта на езика Python. Първоначално се имплементира по отделно класификаторите (NB, KNN и RF). За KNN целият набор от данни се преобразува в числов формат. Тъй като нормализираните данни дават по-добри резултати за изчисления на разстояния, използваме евклидовата мярка за разстояние. Впоследствие преминаваме към хибридни алгоритми, комбинирайки вероятности за класове от NB с KNN, NB с RF и NB както с KNN, така и с RF.

Псевдокодът на предложения хибриден алгоритъм H1, базиран на агрегирането на NB и KNN, е следният:

```
Step 1: Aggregate the probabilities from NB and KNN
    For each test instance Ti
      For each class label Ci
          Probability of Ci= (probability of Ci obtained from NB +
probability of Ci obtained from KNN)/2
      End for
  End for
Step 2: Assign the class label with highest probability to the test
instance.
```

Псевдокодът на предложения хибриден алгоритъм H2, базиран на агрегирането на NB и RF, е следният:

```
Step 1: Aggregate the probabilities from NB and RF
  For each test instance Ti
      For each class label Ci
      Probability of Ci= (probability of Ci obtained from NB + probability
of Ci obtained from RF)/2
      End for
  End for
Step 2: Assign the class label with highest probability to the test
instance.
```

Псевдокодът на предложения хибриден алгоритъм H3, базиран на агрегирането на NB, RF и KNN, е следният:

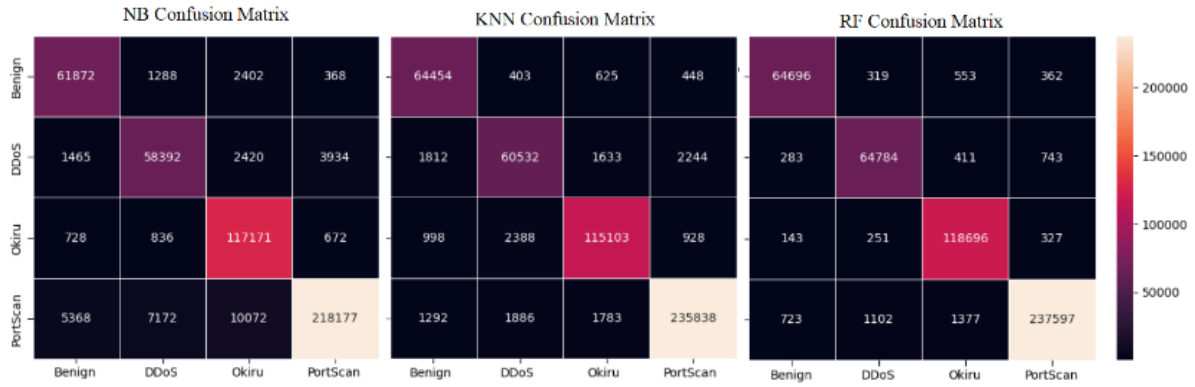
```
Step 1: Aggregate the probabilities from NB, RF and KNN
  For each test instance Ti
      For each class label Ci
      Probability of Ci= (probability of Ci obtained from NB + probability
of Ci obtained from RF + probability of Ci obtained from KNN)/3
      End for
  End for
Step 2: Assign the class label with highest probability value to the test
instance.
```

Предлага се среднопретеглената точност при комбиниране на резултати от различни алгоритми за машинно обучение да се изрази чрез следното съотношение:

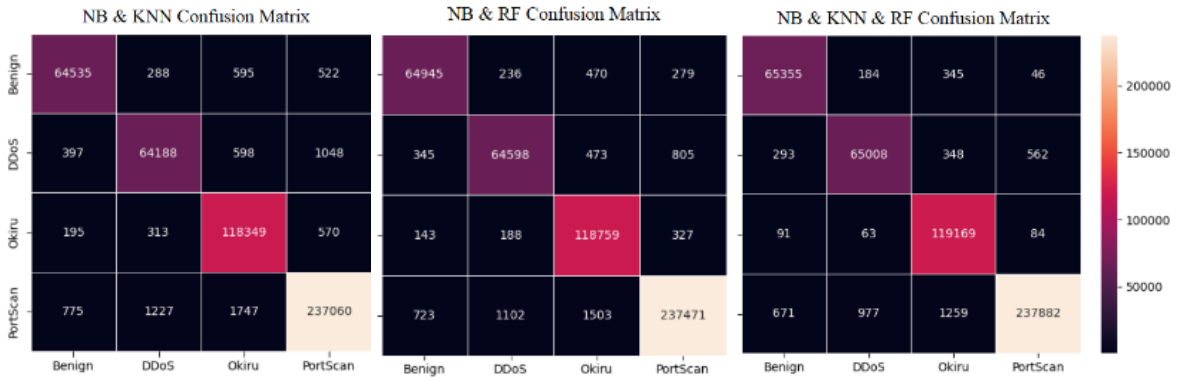
$$Conbined\_Accuracy = \frac{\sum_{i=1}^{n} Accuracy_i * Count_i}{\sum_{i}^{n} Count_i} \quad (2.11)$$

където $Accuracy_i$ точността на *i*-тия алгоритъм докато $Count_i$ представлява броя пъти, в които е постигната *i*-тата точност (т.е. броят на алгоритмите, които са довели до *i*-тата точност), а $n$ е общият брой уникални точности.

Накрая, матрицата на объркване се изчислява за всички гореспоменати индивидуални (NB, KNN, RF) и хибридни H1 (NB & KNN), H2 (NB & RF) и H3 (NB & KNN & RF) алгоритми, както е показано на Фиг. 2.4 и Фиг. 2.5.




*Фиг. 2.4. Матрици на объркване: a) NB, b) KNN, c) RF.*




*Фиг. 2.5. Матрици на объркване: a) H1 (NB & KNN), b) H2 (NB & RF), c) H3 (NB & KNN & RF).*

Матрица на объркване, използвана в задачи за класификация, оценява производителността на моделите за машинно обучение. Тя дава информация за това колко добре моделът може да предсказва различни класове. За този пример е използвана извадка от 492337 записа от набора от данни IoT-23, който включва 4 различни класа – „Benign“, „Okiru“, „DDoS“ и „PortScan“. По диагонала могат да се видят истинските положителни резултати, докато от другата страна са записите, които не са довели до положителна прогноза (вж. Фиг. 2.4 и Фиг. 2.5).

## 2.3. Рамка за класификация на зловреден софтуер, използваща оптимизация на функции и ансамблово обучение

Изкуственият интелект в системите за киберсигурност може да се разпознае в използването на интелигентни алгоритми и техники за машинно обучение за подобряване на откриването, предотвратяването и реагирането на киберзаплахи. Чрез включването на тези подходи, системите за киберсигурност стават способни да анализират огромни количества данни, да идентифицират модели и да помагат за вземането на информирани решения, които надхвърлят човешките възможности. Авторите показват, че изкуственият интелект има значителен ефект върху киберсигурността и обратно, за подизвадки, водени от повишена автоматизация, сложността на кибератаките, която изпреварва защитните възможности (Khan et al., 2024). Поради това се разработват различни нови и хибридни алгоритми за подобряване на защитата срещу зловреден софтуер (Barzev & Borissova, 2025).

Киберсигурността и зловредният софтуер са тясно свързани теми, фокусирани върху защитата на компютърни системи, приложения, устройства, данни, финансови активи и хора от зловреден софтуер и кибератаки. Атаките със зловреден софтуер продължават да нарастват и експертите прогнозират, че честотата им само ще се увеличава през следващите години. Сред разнообразието от начини за откриване на зловреден софтуер, най-често срещаният е сканирането на компютъра за злонамерени файлове или програми.

Няма универсален начин за премахване на зловреден софтуер, тъй като това зависи от конкретния инсталиран зловреден софтуер. Някои често срещани методи за справянето със зловредния софтуер са добре познатите ни антивирусни програми. В тази връзка, усилията на много изследователи са насочени към разработването на различни подходи за предотвъртяване на проблемите, които зловредния софтуер може да причини (Baghirov, 2025). Някои от възможноте подходи внедряват различни алгоритми за дълбоко обучение за класификация на зловреден софтуер (Aslan & Yilmaz, 2021; Brown et al., 2024). Други изследователи се стремят да открият невидими атаки на зловреден софтуер чрез API базирана рамка за Windows и техники за дълбоко обучение (Maniriho et al., 2023). Предложена е също така и нова рамка за откриване на злонамерени действия като механизъм за разпознаване и предотвратяване на атаки

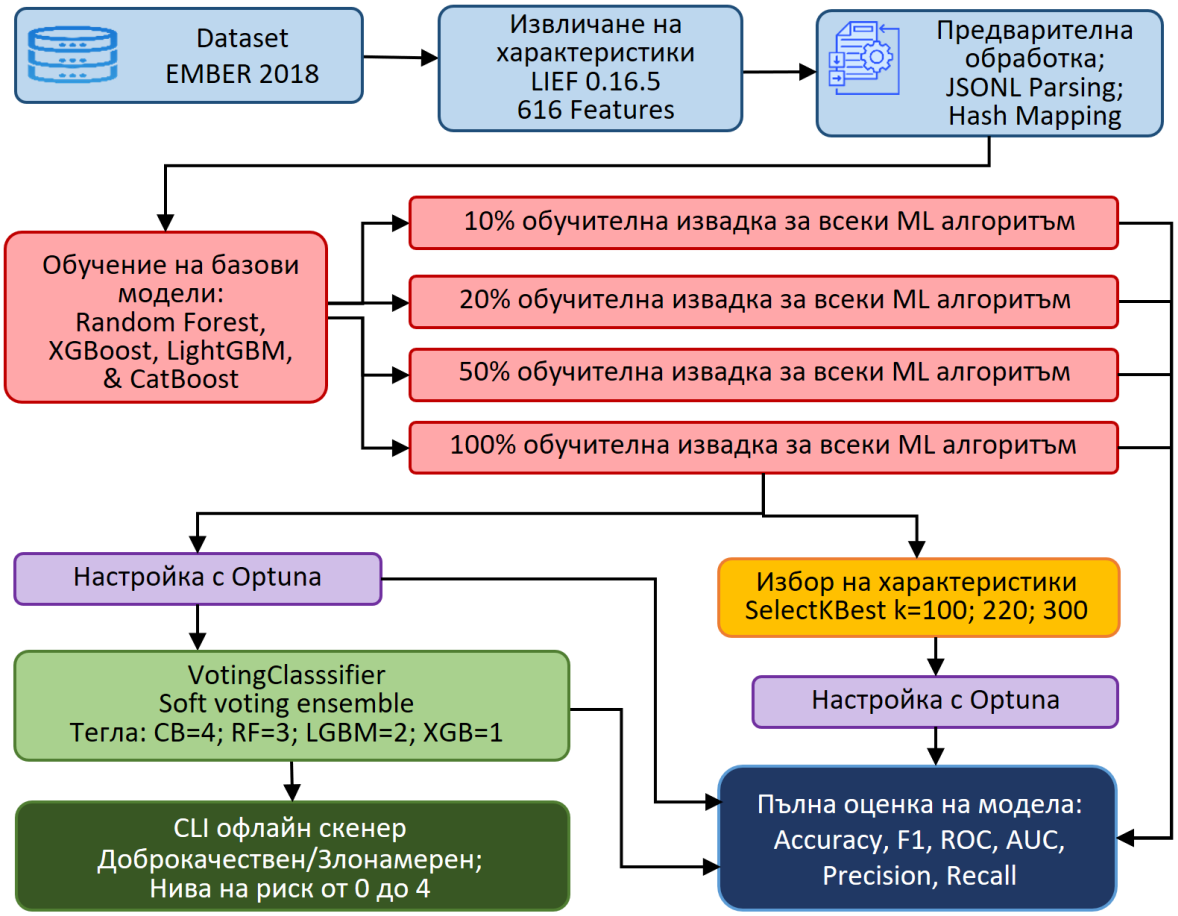
(Parmar & Brahmbhatt, 2023), както и рамка за откриване на зловреден софтуер за Android устройства (Smmarwar et al., 2022).

За да се реализира софтуерна система, способна да открива и идентифицира зловреден софтуер, е необходимо да се анализира неговото поведение. Съществуват две основни категории анализ на зловреден софтуер: статичен анализ и динамичен анализ (Arora et al., 2024). Статичният анализ на зловреден софтуер има няколко ключови предимства – ранно откриване, базиран е на сигнатури и може да използва различни техники, за да разбере естеството на заплахата. Статичният анализ оценява свойствата на зловреден софтуер, като метаданни, низове и структура. Този процес може да бъде автоматизиран чрез използване на дизасемблер, декомпилиране и дебъгери за бърз анализ на голям брой проби от зловреден софтуер. Статичният анализ разчита на обратно инженерство (Reverse Engineering), проверка на кода и подходи, базирани на цифров подпис, за откриване на зловреден софтуер (Omar et al., 2022).

Имайки предвид всички тези съображения, в следващия раздел е описана предложена рамка за откриване и анализ на зловреден софтуер, която има за цел да защити данните, тъй като киберсигурността остава критична не само за отделните лица, но и за бизнеса от всякакъв мащаб.

### 2.3.1. Методология на Python-базирана софтуерна рамка за класификация на статичен зловреден софтуер, използваща оптимизация на характеристики и ансамбълно обучение

За реализиране на рамката за откриване и класифициране на зловреден софтуер се предлага методология, базирана на използването на библиотеки на Python. Тази рамка комбинира няколко алгоритми за машинно обучение, като Random Forest, XGBoost, LightGBM и CatBoost (Thai, 2022; Hancock & Khoshgoftaar, 2020). Предложението за рамката се основана на архитектура от модулен тип, което позволява лесно да се добавят допълнителни модули с различни алгоритми за машинно обучение. За разработването на Python-базирана рамка за автоматична класификация на зловреден софтуер се предлага използването на методология, която е графично илюстрирана на Фиг. 2.6.




*Фиг. 2.6. Методология за разработване на Python-базирана рамка за класификация на зловреден софтуер.*

Предложената рамка за реализиране на класификацията на зловреден софтуер се базира на използването на следните Python библиотеки и модули за конкретната цел:

- *Набор от данни*: Публично достъпен, добре структуриран набор от данни - EMBER 2018 (Anderson, & Roth, 2018), съдържащ 616 предварително извлечени характеристики от PE файлове. Данните са разделени на 80% към 20% за обучение и тестване. Извършихме набор от експерименти съответно върху 10%, 20%, 50% и 100% от данните за обучение, за да оценим влиянието на размера на данните за обучение върху производителността на модела.
- *Обработка на данни*: pandas, numpy – за обработка на структуриране на данни, трансформиране и числен анализ; json – за .jsonl файлове.
- *Обработка на файлове и пътища*: os, pathlib, re, hashlib, sys – за обработка на файлове, регулярни изрази, хеширане и директории.

- *Машинно обучение и модели*: (a) scikit-learn – за базови модели, избор на характеристики и оценка (VotingClassifier, RandomForestClassifier, SelectKBest, mu-tual_info_classif, permutation_importance, FeatureHasher, accuracy_score, pre-cision_score, recall_score, f1_score, roc_auc_score, confusion_matrix, roc_curve); (b) xgboost (XGBClassifier), lightgbm (LGBMClassifier), catboost (CatBoostClassifier) – за модели с градиентно усилване (Rizkallah, 2025); (c) optuna – за хиперпараметрична оптимизация.
- *Визуализация*: matplotlib.pyplot, seaborn, plotly.graph_objects – за графично представяне на резултатите.
- *PE (Portable Executable) анализ и извличане на характеристики*: lief – за извличане на статистически и структурни характеристики от PE файлове.
- *Помощни библиотеки за поддръжка на връзки между модели, като*: joblib, tqdm, logging, argparse, concurrent.futures, time – сериализация, логване, ленти за прогрес, аргументи от командния ред за различни функции, паралелна обработка.

### 2.3.2. Обучение на моделите

За всеки експеримент за всеки алгоритъм (по един за всеки тренировъчен набор) се обучават четири модела, което води до общо 16 базови модела. Всички модели са запазени като .pkl файлове за възпроизводимост и сравнение.

Тествани са четири алгоритми за машинно обучение: Random Forest, XGBoost, LightGBM и CatBoost, обучени върху 4 различни пропорции на обучителни данни: 100%, 50%, 20% и 10%.

Един от основните въпроси, на които трябва да се отговори, е дали производителността на модела се влияе от количеството обучителни данни, както и какви показатели биха били използвани за количествено определяне на този въпрос. Показателите за оценка на тези 45 обучени модела са представени в следващата Таблица 2.2.

*Таблица 2.2. Резултати от моделите с различни тренировъчни набори от данни.*

| # | Model | K | Accuracy | F1-Score | ROC AUC | Precision | Recall | Optuna |
|---|---|---|---|---|---|---|---|---|
| | | | | **100% Train Size** | | | | |
| **1** | RandomForest | 616 | 0.9391 | 0.9392 | 0.9796 | 0.9353 | 0.9433 | Yes |
| **2** | RandomForest | 616 | 0.9245 | 0.9241 | 0.9542 | 0.9285 | 0.9198 | No |
| **3** | RandomForest | 300 | 0.9117 | 0.9116 | 0.9426 | 0.9129 | 0.9093 | No |
| **4** | RandomForest | 300 | 0.9362 | 0.9369 | 0.9777 | 0.9332 | 0.9387 | Yes |
| **5** | RandomForest | 200 | 0.9371 | 0.9371 | 0.962 | 0.9344 | 0.9394 | Yes |
| **6** | RandomForest | 200 | 0.9188 | 0.9184 | 0.9435 | 0.9193 | 0.9175 | No |
| **7** | RandomForest | 100 | 0.9357 | 0.9356 | 0.9506 | 0.9328 | 0.9382 | Yes |
| **8** | RandomForest | 100 | 0.9154 | 0.9154 | 0.9454 | 0.9161 | 0.9145 | No |
| **9** | XGBoost | 616 | 0.9173 | 0.9184 | 0.9728 | 0.9063 | 0.9308 | Yes |
| **10** | XGBoost | 616 | 0.9082 | 0.9171 | 0.9482 | 0.8926 | 0.9281 | No |
| **11** | XGBoost | 300 | 0.9283 | 0.9287 | 0.9586 | 0.9207 | 0.9368 | Yes |
| **12** | XGBoost | 300 | 0.7762 | 0.8027 | 0.9124 | 0.7176 | 0.9106 | No |
| **13** | XGBoost | 200 | 0.9268 | 0.9269 | 0.9735 | 0.9201 | 0.9336 | Yes |
| **14** | XGBoost | 200 | 0.7604 | 0.7781 | 0.9007 | 0.7246 | 0.8412 | No |
| **14** | XGBoost | 100 | 0.9255 | 0.9258 | 0.9407 | 0.9191 | 0.9324 | Yes |
| **15** | XGBoost | 100 | 0.8294 | 0.8215 | 0.8394 | 0.8616 | 0.7852 | No |
| **16** | LightGBM | 616 | 0.9389 | 0.9391 | 0.9691 | 0.9351 | 0.9432 | Yes |
| **17** | LightGBM | 300 | 0.9441 | 0.9438 | 0.9853 | 0.9445 | 0.9437 | Yes |
| **18** | LightGBM | 200 | 0.9424 | 0.9425 | 0.9841 | 0.9399 | 0.9451 | Yes |
| **19** | LightGBM | 100 | 0.9427 | 0.9425 | 0.9576 | 0.9432 | 0.9418 | Yes |
| **20** | LightGBM | 200 | 0.9011 | 0.9036 | 0.9283 | 0.8815 | 0.9268 | No |
| **21** | LightGBM | 616 | 0.8970 | 0.8993 | 0.9645 | 0.8795 | 0.9201 | No |
| **22** | LightGBM | 300 | 0.8257 | 0.8408 | 0.9232 | 0.7739 | 0.9203 | No |
| **23** | CatBoost | 616 | 0.9405 | 0.9406 | 0.9706 | 0.9388 | 0.9424 | Yes |
| **24** | CatBoost | 616 | 0.9292 | 0.9298 | 0.9597 | 0.9215 | 0.9382 | No |
| **25** | CatBoost | 300 | 0.9341 | 0.9342 | 0.9757 | 0.9299 | 0.9385 | Yes |
| **26** | CatBoost | 300 | 0.8447 | 0.8503 | 0.8795 | 0.8206 | 0.8823 | No |
| **27** | CatBoost | 200 | 0.9355 | 0.9359 | 0.9608 | 0.9307 | 0.9411 | Yes |
| **28** | CatBoost | 200 | 0.9237 | 0.9236 | 0.9689 | 0.9246 | 0.9226 | No |
| **29** | CatBoost | 100 | 0.9315 | 0.9313 | 0.9464 | 0.9275 | 0.9351 | Yes |
| **30** | CatBoost | 100 | 0.8507 | 0.8517 | 0.8665 | 0.8457 | 0.8578 | No |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **31** | VotingClassifier (XGB, LGBM, CB, All Features) | 616 | 0.9461 | 0.9461 | 0.9864 | 0.9468 | 0.9454 | No |
| | | | | **50% Train Size** | | | | |
| **32** | RandomForest | 616 | 0.8652 | 0.8533 | 0.9253 | 0.9338 | 0.7856 | No |
| **33** | XGBoost | 616 | 0.8668 | 0.8621 | 0.9216 | 0.8934 | 0.8329 | No |
| **34** | LightGBM | 616 | 0.8659 | 0.8605 | 0.9352 | 0.8967 | 0.8272 | No |
| **35** | LightGBM | 100 | 0.8827 | 0.8841 | 0.8989 | 0.8735 | 0.8951 | No |
| **36** | CatBoost | 616 | 0.8891 | 0.8846 | 0.9543 | 0.9223 | 0.8498 | No |
| | | | | **20% Train Size** | | | | |
| **37** | RandomForest | 616 | 0.7855 | 0.7375 | 0.9247 | 0.9497 | 0.6028 | No |
| **38** | XGBoost | 616 | 0.8381 | 0.8244 | 0.9256 | 0.9003 | 0.7603 | No |
| **39** | LigghtGBM | 616 | 0.8349 | 0.8141 | 0.9244 | 0.9314 | 0.7234 | No |
| **40** | CatBoost | 616 | 0.8625 | 0.8487 | 0.9408 | 0.9433 | 0.7713 | No |
| | | | | **10% Train Size** | | | | |
| **41** | RandomForest | 616 | 0.5972 | 0.3336 | 0.7517 | 0.9663 | 0.2016 | No |
| **42** | XGBoost | 616 | 0.6626 | 0.5165 | 0.8121 | 0.9112 | 0.3604 | No |
| **43** | LigghtGBM | 616 | 0.7463 | 0.6683 | 0.9034 | 0.9658 | 0.5112 | No |
| **44** | CatBoost | 616 | 0.6458 | 0.4664 | 0.7613 | 0.9458 | 0.3096 | No |

От базовите модели (без Optuna оптимизация), CatBoost показа най-добрите стойности със значителна разлика, използвайки целия набор от данни. Времето за обучение за 100% набор от тренировъчни данни е около 15-30 минути за модел (в зависимост от модела). За по-малките размери на обучение се използва значително по-малко време за обучение. Увеличаването на количеството обучени данни от 10% до 100% подобри производителността по отношение на точността за всеки модел. XGBoost и CatBoost осигуриха най-голямо подобрение. LightGBM е най-бързият за обучение, докато точността му в режим по подразбиране е малко по-ниска от останалите.

### 2.3.3. Избор на характеристики и оптимизация на хиперпараметри

Всички модели и оптимизации в този раздел използваха набора от характеристики, създаден с адаптирания скрипт features.py и съвместим с lief версия 0.16.5 и Python версия 3.8, за да се установи контролирана и последователна среда с EMBER 2018. Прилагат се два важни метода за оптимизиране и систематизиране на

производителността: намаляване на размерността чрез SelectKBest и оптимизация на хиперпараметри с помощта на Optuna.

Общо 45 модела са обучени и оценени със следните комбинации: 4 алгоритми за машинно обучение (Random Forest, XGBoost, LightGBM, CatBoost), 4 различни размера на тренировъчния набор (10%, 20%, 50%, 100%), Optuna включена и изключена и различни стойности на SelectKBest, k = 100; 200; 300; всеки модел се оценява от независим тестов набор по 5 основни показателя (вж Таблица 2.2).

Алгоритъмът за избор на характеристики SelectKBest, използващ mutual_info_classif, се използва за идентифициране на най-информативните характеристики. При сравняване на броя характеристики и производителността, параметърът k се тества съответно със 100, 200 и 300 характеристики. Има малка загуба на точност, дължаща се на по-малко избрани характеристики, но както k = 200, така и k = 300 не се различават много от целия набор от характеристики, което означава, че по-малко функции могат да бъдат полезни в ситуации, където изчислителните ресурси са фактор. Пълното оптимизиране на различни модели с Optuna изисква значително количество изчислителни и времеви ресурси: всеки от 4-те алгоритъма (CatBoost, XGBoost, LightGBM, Random Forest) е изпълнил 50 опита плюс различни измерения на характеристиките и дълбочини на модела, което е отнело приблизително от 100 до 140 часа чисто изчислително време. Работата включва дълги цикли на валидиране, регистриране, преглед на резултатите и запазване на най-добре представящите се модели. Вложеното време отразява прецизността и ангажимента за създаване на надеждна високопроизводителна архитектура, интегрираща алгоритми за машинно обучение.

### 2.3.4. Оптимизирани хиперпараметри от най-добре представящите се конфигурации

Тъй като настройката на хиперпараметрите е от съществено значение за оптимизиране на производителността на алгоритмите за машинно обучение (Lujan-Moreno et al., 2018), тук са описани най-добрите хиперпараметри, постигнати за всеки от приложените алгоритми. Всеки параметър е избран въз основа на селекцията на Optuna от общо 50 опита за всеки параметър, така че те да могат да осигурят добра производителност, обобщаемост и да съкратят времето за обучение. Те представляват

най-добрите конфигурации, базирани на F1-оценка на валидационния набор, а крайните сравнения на моделите са използвали тези стойности. Описаните хиперпараметри са добре балансирани мерки за производителност/скорост и са валидирани с помощта на пълния набор от функции (616 функции) с трансформации SelectKBest (k = 100; 200; 300). Кратки описания за всяка от съответните характеристики на параметрите са включени в скоби.

- LightGBM:

  $max\_depth = 6$ (максимална дълбочина на дървото, за да се елиминира рискът от прекомерно нагаждане), $learning\_rate = 0.07$ (моделира скоростта на адаптация към данните), $n\_estimators = 420$ (броят на итерациите достатъчен за средна скорост на обучение)

- CatBoost:

  $depth = 8$ (максимална дълбочина на дървото, достатъчна за улавяне на непряки връзки), $learning\_rate = 0.05$ (средната скорост на обучение е по-бавна, но се сближава постоянно), $iterations = 500$ (брой итерации, необходими за ниска скорост на обучение)

- XGBoost:

  $n\_estimators = 468$ (брой усилващи рундове), $max\_depth = 10$ (дълбоки дървета за моделиране на сложни зависимости), $learning\_rate = 0.0942$ (стандартна стойност за стабилно обучение), $subsample = 0.7039$ (нагаждане, контролирано чрез използване на една извадка от всички данни от извадките), $colsample\_bytree = 0.8392$ (броят на използваните характеристики на дърво)

- Random Forest:

  $n\_estimators = 132$ (брой дървета в гората за стабилни прогнози), $max\_depth = 28$ (може да създава относително дълбоки дървета, като същевременно обещава да позволи само разумно пренареждане), $min\_samples\_split = 5$ (прекомерни клони), $min\_samples\_leaf = 2$ (има твърде чувствителни клони), $max\_features = 'log2'$ (броят на характеристиките, когато дърветата на решенията изграждат разделянията, е във всички измерения, така че извадките увеличават разнообразието на дърветата.)

Струва си да се отбележи, че времето за изпълнение на това обучение - оптимизацията на Optuna с пълна конфигурация на всички данни за обучение (100%) / SelectKBest (k = 300) – е изключително времеемко. Единично изпитание за CatBoost или XGBoost в пълния му обхват по време на изпитание може да отнеме от 20 до 90 минути, в зависимост от дълбочината на модела, броя на итерациите и входните данни. Следователно, в идеалния случай до 50 изпълнения за един модел отнемат приблизително от 20 до 50 часа чисто изчислително време.

Optuna предостави ясни подобрения спрямо базовите конфигурации и очаквано подобри производителността за CatBoost и Random Forest. Най-голямото подобрение в точността е за LightGBM, което се подобри от 0.8970 до 0.9389. С различни стойности на k=100;200;300, SelectKBest даде на някои модели с по-ниска размерност на данните производителност, която се конкурира или надвишава производителността при използване на всички функции. Например, XGBoost с k = 300 и Optuna успяват да постигнат точност = 0.9283 и F1-оценка = 0.9287, което е по-добре от същия модел, използващ общо 616 функции (Точност = 0.9173, F1 = 0.9184). LightGBM с k = 300 показа гранична точност (Точност = 0.9392) по-добра от пълната си версия (0.9389). CatBoost с k = 300 показа, че може да се представи отлично (Точност = 0.9341, F1 = 0.9342), като същевременно се доближава до версията си с резултати с пълен вход (0.9405). По този начин SelectKBest може да бъде ефективен като стратегия в ситуации с ограничени ресурси или време. За крайния модел на ансамбъла (описан в следващия раздел) обаче се използва пълният набор от данни от 616 характеристики, така че се постига едно единствено бенчмарк число, за да се осигури балансиран вход за всички модели в ансамбъла.

### 2.3.5. Ансамблов метод с класификатор за гласуване (Voting Classiffier)

Предложеният ансамблов метод (Voting Classiffier), който комбинира оптимизирани модели с уникални архитектури, даде най-благоприятния резултат в анализа (вж. Тблица 2.2). Моделът VotingClassifier, който прилага меко гласуване (Soft Voting) с определени тегла, направи прогнози по-силни от отделните модели, включително CatBoost, който имаше най-добра производителност. Всичко това е доказателство за оползотворяване на силата на уникални модели с уникални полезни алгоритми.

След като отделните експерименти са извършени и хипернастройката е завършена на всеки модел, конфигурацията на ансамбъла се формира с помощта на VotingClassifier. Целта на ансамбъла е да може да комбинира всички силни страни на целия потенциал на алгоритъма и да минимизира рисковете за отделните модели, използвайки меко гласуване като опора.

Съставът на VotingClassifier е следният: CatBoost (Optuna, 616 характеристики); LightGBM (Optuna, 616 характеристики); Random Forest (Optuna, 616 характеристики); и XGBoost (Optuna, 616 характеристики). Мекото гласуване (Soft Voting) се използва за обединяване на вероятностни резултати от модела. Мекото гласуване е предпочитано пред твърдото гласуване, защото то позволява обединяването на прогнозите по много по-информативен начин, тъй като прогнозираните вероятности могат да бъдат обединени и разгледани, а не само обединени и твърдо определени класификации. Освен това, използването на меко гласуване позволява на моделите да изразят своята увереност, което е особено важно при модели, които се различават по своите фалшиво положителни/отрицателни резултати, смесени с чувствителност и специфичност.

Всички модели са обучени върху целия 100% набор от данни за обучение. Всеки модел използва пълния вектор от 616 характеристики, за да се гарантира, че входната структура е сходна. Претеглените класификации за VotingClassifier (базирани на производителност) са следните: $\alpha_1$ за CatBoost; $\alpha_2$ за Random Forest; $\alpha_3$ LightGBM; и $\alpha_4$ за XGBoost. Тези тегла представляват относителната сила на всеки модел след настройката на Optuna, с цел да се позволи на силата на най-добрите възможни предиктори да осигурят най-добрия резултат. И накрая, директната Optuna не се прилага към VotingClassifier, тъй като той е ансамбъл от базови модели, които вече са оптимизирани, и прилагането на ансамблово ниво на настройка би довело до значителни изчислителни разходи с много малко очаквани печалби от производителността, които да се оправдаят. Експертно дефинираните тегла и мекото гласуване са едновременно фина настройка и помогнаха гласуването да бъде малко по-изчислително. Резултатите от VotingClassifier е показан в Таблица 2.2.

VotingClassifier постигна най-високите стойности за всички основни показатели за оценка. Той надмина всички модели без гласуване; дори най-добрият като цяло модел CatBoost (0.9405). Ансамбъл, моделиран с предварително настроени базови модели, също така позволи намаляване на индивидуалните слабости на моделите.

Практическите предимства са, че моделът има висока стабилност и надеждност, когато се използва в реални условия. Приложението на модела би било подходящо за вграждане в действителен CLI (Command Line Interface) скенер (подробно описан в следващия раздел) със силен баланс между точност и интерпретируемост.

Претегленият ансамбъл с меко гласуване не само превъзхожда всички модели без гласуване, но също така има силна стабилност и адаптивност в реалистични среди. Настройката на ансамбъла с гласуване използва изключително добре оптимизирани базови модели и контролирано гласуване, за да осигури най-високата точност на класификация сред изследването (точност: 94,61%).

## 2.4. Самоосъзната класификация на зловреден софтуер чрез рутиране на модели на базата на система за доверие за избора и обяснимост на характеристиките

Широко разпространената дигитализация на различни бизнес области, както и голямото количество данни, произтичащи от Интернет на нещата, са предпоставка за растежа на кибератаките. Според доклада за първото тримесечие на 2025 г. средният брой атаки на организация на седмица се е увеличил до 1925[2]. Това представлява 47% увеличение на броя на атаките за същия период на предходната 2024 г. Този факт ясно показва нарастващото предизвикателство, свързано с осигуряването на стабилна позиция за киберсигурност на фона на постоянно променящия се пейзаж на заплахите. Оказва се, че образователният сектор е сред най-уязвимите с 4484 атаки на седмица, което е 73% увеличение в сравнение с предходната година. Тази статистика е достатъчно доказателство за спешната нужда от средства и инструменти за справяне с кибератаките. Това е и причината много изследователи да са съсредоточили усилията си не само върху развиването на ИКТ компетенции (Shalamanov et al., 2020) и кибер полигони (Blagoev & Shalamanov, 2023), съпътстващи дигиталната трансформация, но и върху разработването както на подходящи алгоритми, така и на цялостни решения за откриване на зловреден софтуер (Guliashki et al., 2023).

[2] https://blog.checkpoint.com/research/q1-2025-global-cyber-attack-report-from-check-point-software-an-almost-50-surge-in-cyber-threats-worldwide-with-a-rise-of-126-in-ransomware-attacks/

Откриването на зловреден софтуер, базирано на машинно обучение, се основава на два основни етапа, свързани с извличането на характеристики и класификацията на зловреден софтуер. Използването на модели за машинно обучение води до подобрено откриване на зловреден софтуер, като се взема предвид не само изборът на характеристики, но и техниките за мащабиране (Hasan et al., 2025). Доказано е, че техниките за мащабиране допринасят за постигане на равен принос на всяка характеристика към специфичния модел за откриване. Други автори използват дълбок анализ за намаляване на характеристиките и внедряват леки алгоритми, за да постигнат ефективност на работата на всички устройства (Farfoura et al., 2025). Показано е, че предварително обучените CNN модели могат да се използват като инструмент за извличане на характеристики (Bakır, 2025). В резултат на това е предложен нов подход, който е способен да оптимизира конфигурациите на CNN модели с подобрени нива на откриване и е устойчив на обфускация. Чрез използване на комбинации от различни алгоритми за машинно обучение може да се постигне оптимизация на извличането на характеристики, което води до подобрено откриване на зловреден софтуер (Barzev & Borissova, 2025). Освен това, използването на комбинация от паралелно дълбоко обучение и хибридна BP-PSO рамка също е подходящо за извличане на характеристики (Al-Andoli et al., 2022). Всичко това показва, че машинното обучение и дълбокото обучение са приложими при откриване на зловреден софтуер и тяхната ефективност произтича от анализа, чрез който се извършва изборът на характеристики. Въпреки обещаващите резултати, тези модели изискват подобрена обяснимост за внедряване в реални приложения. В тази връзка се предлага използването на обясним изкуствен интелект (Explainable AI - xAI), за да се разберат и оценят решенията, взети от моделите за машинно обучение при откриване на зловреден софтуер (Baghirov 2025).

Съществуват различни техники за откриване на зловреден софтуер (Inayat et al., 2021; Alsmadi & Alqudah, 2021; Dutta et al., 2022), сред които са откриване, базирано на сигнатури, статичен и динамичен анализ (Lebbie et al., 2022; Palsa et al., 2022), откриване по евристични правила (Čisar et al., 2025; Kalla et al., 2024) и онлайн откриване на зловреден софтуер (Manthena et al., 2023). Техника, базирана на сигнатури, се използва в предложен скенер за зловреден софтуер за Android приложения (Pardhi et al., 2021). Статичният анализ на зловреден софтуер разчита на извличане на данни и техники за машинно обучение и е най-често използваният при анализа на изпълними файлове за

откриване на зловреден код. Това може да се реализира чрез проектирането и внедряването на различни класификатори за машинно обучение и отделни категории характеристики, статично извлечени от изпълняеми файлове (Balram et al., 2019). За да се подобри класификацията на зловреден софтуер, е възможно да се комбинират както статични, така и динамични техники (Chanajitt et al., 2021; Thakur et al., 2024). Въпреки всичко това, откриването на зловреден софтуер остава предизвикателство поради нарастващата сложност на заплахите.

Класическите системи за откриване на зловреден софтуер използват предимно статични мерки, базирани на правила, или статични модели за машинно обучение, получени от стари набори от данни. Те могат да доведат до висока точност на класификация на известни данни, но ако тези характеристики се отклоняват с времето, както обикновено се случва с примери за зловреден софтуер, се получава влошаване на производителността. Най-често това се случва поради преобучение, ограничена адаптивност или надценяване на достоверността при предишно обучение на модела към вход, който сега е неизвестен или нов, въпреки че векторът на атака е същият.

В няколко публикации (Augello et al., 2025; Khan & Nauman, 2024; Jia et al., 2023) е описано използването на набора от данни EMBER 2018 (Anderson & Roth, 2018) за целите на откриването на статичен зловреден софтуер, и които демонстрират обещаваща основа за разработване на модели. Характеристиките на зловредните софтуери се променят с течение на времето до точката, в която статичните модели могат да бъдат крехки, а въпросът за откронението на характеристиките става сериозен фактор за ефективността на моделите. Нови публикации изследват обяснимия изкуствен интелект (XAI), маршрутизирането на модели и системите за вземане на решения, базирани на доверие, но доколкото е известно, имплементации и/или системи, използващи тези аспекти, са рядкост, предимно за приложения за статична двоична класификация с вградена резервна логика и логика на доверие (Manthena et al., 2025).

Проведеният анализ в настоящото дисертационно изследване използва както стари, така и нови набори от данни, за да се направят сравнения и също така се предлага ансамблов метод за осигуряване на справедлива производителност в различните времеви вектори на развиващите се заплахи.

В отговор на тези предизвикателства, в дисертационното изследване се предлага рамка, която позволява самоосъзната еволюция в класификацията на двоичен зловреден софтуер (злонамерен срещу доброкачествен). Тази интелигентна рамка е едновременно адаптивна и устойчива, тъй като включва самоосъзнат класификатор на модели (SAMC), който използва адаптивна логика за автоматичен избор между традиционни и съвременни слоеве на моделите чрез измерване на надеждността на прогнозирането. Приложимостта на предложения подход е проверена чрез използване на модели, обучени върху наборите от данни EMBER 2018 и EMBER 2024, работещи със същия статичен, 616-мерен вектор от характеристики, получен от статичен анализ на PE файлове.

### 2.4.1. Набор от данни и извличане на характеристики

EMBER 2024 е широкомащабен отворен набор от данни за откриване на статичен зловреден софтуер, разработен от Future Computing Lab и публикуван в началото на 2024 г. EMBER 2024 е официалният наследник на бенчмарк набора от данни EMBER 2018. EMBER 2024 включва общо над 5 милиона примера, в много етикети, включително:

- 4 000 000 примера за обучение (Win32, .NET и неизвестни PE)
- 1 000 000 тестови примера (Win32 и други PE)
- Множество типове класове/архитектура, включително специфични за .NET и подписани с Authenticode файлове.

Всяка извадка се съхранява във формат JSONL с полета както за сурови, така и за производни характеристики, включително байтови хистограми, ентропия, анализ на низове, импортиране, заглавки, секции, коефициент на откриване (от гласовете на доставчиците за антивирусна диагностика) и размити хешове като TLSH. Наборът от данни е структуриран с много повече от 128 отделни .jsonl файла с много доброкачествени и злонамерени извадки.

В това изследване се фокусираме върху подмножеството Win32 PE и извлякохме общо 1 560 000 обучителни и 360 000 тестови извадки. След като базовите характеристики бяха извлечени, ние анализирахме и конвертирахме извадките, така че те да съответстват на допълнителните характеристики от набора EMBER 2018, използвайки персонализиран Python скрипт, който нормализира JSONL файловете и

подравни всички полета. Нормализацията беше от решаващо значение, за да се гарантира, че можем да дублираме оригиналната логика на PEFeatureExtractor, за да осигурим съвместимост между наборите от данни и всички да работят на общ ML пайплайн.

Извличането на характеристики беше извършено с помощта на модифициран PEFeatureExtractor конвейер с lief==0.16.5 и Python 3.8, за да се осигури ретросъвместимост със стари PE формати. Всяка извадка се извлича в 616-измерен вектор, който включва байтови хистограми, ентропийни части и низови характеристики.

В Таблица 2.3 са показани основните разлики между наборите от данни EMBER 2018 и EMBER 2024.

*Таблица 2.3. EMBER 2018 и EMBER 2024 набори от данни.*

| Атрибут | EMBER 2018 | EMBER 2024 |
|---|---|---|
| **Release year** | 2018 | 2024 |
| **Training samples** | 900 000 | 1 560 000 |
| **Test samples** | 200 000 | 360 000 |
| **Feature count** | 616 | 616 (harmonized) |

### 2.4.2. Методология

**Обучение и настройка на модела**

Обучихме четири алгоритми за машинно обучение (RandomForest, XGBoost, CatBoost, LightGBM), използвайки 100% от обучителния набор EMBER 2024, всички 616 характеристики. Проведена беше оптимизация на хиперпараметри използвайки Optuna:

- LightGBM: 100 Optuna опити,
- XGBoost, CatBoost, RandomForest: 30 Optuna опити за всеки.

Продължителността на обучение се различава в зависимост от модела:

- RandomForest: 15 ÷ 40 min
- LightGBM: 2 ÷ 14 min
- CatBoost: 8 ÷ 40 min
- XGBoost: 56 ÷ 180 min

Регистрираните най-добри параметри за всеки модел са оценени, използвайки различни показатели като точност, F1-оценка, ROC-AUC и прецизност, попълнота. Резултатите са представени в Таблица 2.4.

*Таблица 2.4. Сравнение на производителността на отделни класификатори, оптимизирани за Optuna варианти и ансамбъла VotingClassifier за EMBER 2024 (Win32).*

| Модел | Accuracy | F-1 score | ROC-AUC | Precision | Recall |
|---|---|---|---|---|---|
| **CatBoost** | 0.9397 | 0.9382 | 0.9881 | 0.9622 | 0.9153 |
| **LightGBM** | 0.9438 | 0.9425 | 0.9898 | 0.9650 | 0.9210 |
| **LightGBM (Optuna)** | 0.9540 | 0.9531 | 0.9925 | 0.9710 | 0.9359 |
| **RandomForest** | 0.9545 | 0.9535 | 0.9927 | 0.9746 | 0.9333 |
| **CatBoost (Optuna)** | 0.9546 | 0.9538 | 0.9924 | 0.9710 | 0.9372 |
| **RandomForest (Optuna)** | 0.9552 | 0.9542 | 0.9928 | 0.9747 | 0.9346 |
| **XGBoost** | 0.9575 | 0.9568 | 0.9936 | 0.9730 | 0.9411 |
| **XGBoost (Optuna)** | 0.9664 | 0.9659 | 0.9955 | 0.9795 | 0.9527 |
| **VotingClassifier (All models)** | 0.9654 | 0.9649 | 0.9953 | 0.9793 | 0.9509 |

Когато разгледаме внимателно всеки отделен алгоритъм, може да се установи следното:

- CatBoost постигна най-ниската попълнота (0.9153) от всички модели, но също така имаше много добра прецизност от 0.9622. Това означава, че CatBoost е по-малко вероятно да генерира фалшиви положителни резултати, но вероятно е пропуснал някои злонамерени файлове. В допълнение към подобрението на попълнотата до 0.9372 в оптимизираната за Optuna версия, ясно е, че CatBoost реагира положително на оптимизацията на хиперпараметрите.
- LightGBM е най-бързият за обучение и се представи добре по всички показатели. Оптимизираната за Optuna версия на LightGBM успя да постигне забележими подобрения в F1 и ROC-AUC, както се очакваше, но все пак показа по-ниска попълнота от другите модели, което показва по-консервативно поведение при класификация.
- RandomForest показа един от най-добрите баланси между прецизност и попълнота. Оптимизираната за Optuna версия даде по-висок F1 и прецизност (0.9747), което означава, че е добър алгоритъм за балансирано откриване, но все пак е по-нисък от XGBoost при попълнота.

- XGBoost успя да постигне отлични резултати дори без коригиране на хиперпараметри с F1-оценка от 0.9568 и ROC-AUC от 0.9936, които бяха по-добри от всички други неоптимизирани модели. След коригиране на производителността на XGBoost с оптимизация на хиперпараметри с Optuna, XGBoost даде най-високите резултати за показателите за оценка: 0.9664 Точност, 0.9659 F1-оценка и 0.9955 ROC-AUC. Оптимизираният XGBoost се утвърди като най-добре представящия се и най-стабилен индивидуален модел като цяло, както с висока способност за обобщаване, така и с изключителна преизност 0.9795 и попълнота 0.9527.
- VotingClassifier, който е ансамблов модел, използващ претеглено меко гласуване на другите модели: Експериментирахме с 2 ансамблови модела – всичките 4 модела комбинирани и XGB, RF, LightGBM. VotingClassifier (XGB, RF, LightGBM) даде почти идентична производителност с оптимизирания XGBoost: 0.9661 точност, 0.9656 F1-оценка и 0.9955 ROC-AUC. Този ансамблов модел успя да стабилизира прогнозите и да компенсира някои слабости на отделните класификатори (като по-ниска попълнота за CatBoost и малко по-ниска прецизност за LightGBM), но не даде по-добра производителност от оптимизирания модел XGBoost.

Както бе споменато по-горе, XGBoost осигурява най-висока производителност, така че бе избран за новия основен модел. В сравнение с публикувани по-рано резултати, използващи EMBER 2018, беше важно да се видят осезаеми печалби както във времето, така и в точността при използване на същите алгоритми за машинно обучение (Random Forest, XGBoost, CatBoost и LightGBM) върху по-малък набор от данни с по-малко извадки и по-малко финозърнести метаданни. Моделите RandomForest, обучени върху публикуваните данни на EMBER 2018, отнеха приблизително 15-30 минути като моделите, обучени върху EMBER 2024, но дадоха по-ниска точност между 0.91-0.94, заедно с по-ниска попълнота. LightGBM беше и най-бързият модел и в двете проучвания, но имаше най-голямо подобрение от среден ~0.90 F1-оценка на EMBER 2018 до 0.95+ след оптимизация на хиперпараметри с EMBER 2024 (вторият експеримент). Моделите XGBoost и CatBoost също така постигнаха значително подобрение както в прецизността, така и в попълнотата на данни, използвайки по-богат набор от функции от версията EMBER 2024, а по-представителните данни за обучение

вероятно биха довели до подобрени модели, както показват някои от откритията. Резултатите потвърдиха, че EMBER 2024 е по-добър не само за предоставяне на по-разнообразни извадки, но и за позволяване на съвременните модели да се обобщават по-добре, особено когато те са съчетани с нови процеси за настройка на хиперпараметри, като например използването на Optuna.

**Ансамблов модел: VotingClassifier**

Създаден е ансамблов модел с помощта на две комбинации от оптимизираните класификатори чрез VotingClassifier, с меко гласуване и тегла за всеки класификатор в зависимост от неговата надеждност.

Първа комбинация (с всички модели):

- XGBoost: 10;
- RandomForest: 2;
- CatBoost: 1;
- LightGBM: 4.

Втора комбинация (с XGBoost, Random Forest, LightGBM):

- XGBoost: 15;
- RandomForest: 1;
- LightGBM: 3.

Въпреки че се постарахме да изградим силен алгоритъм на VotingClassifier, оптимизираният алгоритъм XGBoost превъзхожда другите алгоритми по отношение на обобщаване и е основният класификатор в SAMC рамката.

**Самоосъзнаващ се класификатор на модели)**

Признавайки, че зловредният софтуер непрекъснато се развива с течение на времето и трябва да се адаптира към него, разработихме самоосъзнат класификатор на модели, който да има възможност за динамично насочване към подходящия модел въз основа на нивото на доверие на прогнозата. Целта на логиката на този класификатор е да се намери баланс между доверие, обобщаемост и устойчивост при използване на стари и съвременни модели на машинно обучение, като и двата модела са обучени върху два различни набора от данни: EMBER 2018 (стар) и EMBER 2024 (нов).

**Логика за избор на модел**

Всеки .exe файл първо се анализира статично чрез PEFeatureExtractor пайплайн, което води до 616-измерен вектор от характеристики. След това векторът се въвежда в следните два класификатора, които са обучени независимо:

- Наследен (Legacy) модел: VotingClassifier трениран върху EMBER 2018, който е съставен от моделите RandomForest, CatBoost, LightGBM, и XGBoost с меко гласуване (тежести: RF=3, CB=2, LGBM=1, XGB=4), постигайки следните най-добри показатели върху legacy данните: Accuracy 0.9646 0.9461, F1-Score 0.9646 0.9461, ROC-AUC 0.9951 0.9864, Precision 9468 , Recall 0.9454
- Нов (New) модел: един XGBoost (Optuna) класификатор трениран върху 2024 Win32 проби от EMBER. Той също така постига всички най-добри показатели: Accuracy 0.9583 0.9664, F1-Score 0.9576 0.9659, and ROC-AUC 0.9938 0.9527, Precision 0.9795, Recall 9527.

**Рутиране, базирано на доверие**

Нека с *maxproba* означим максималната прогнозирана вероятност за клас на даден модел. В логиката на предложения самоосъзнат класификатор се използва условие, на база на което да определи рутирането следвайки следните предположения:

- Ако $max\_proba(new\_model) \geq t$ – довери се на прогнозата на новия модел (подходяща стойност може да бъде например t = 0.85).
- В противен случай, ако $max\_proba(new\_model) \geq t$ – довери се на прогнозата на стария модел
- В противен случай: комбинирай прогнозираните вероятности за класове и на двата модела като резервен вариант, като предложите среднопретеглена стойност, както следва:

$$Pfinal = 0.6 \cdot Pnew + 0.4 \cdot Plegacy \quad (2.11)$$

Най-високата вероятност от тази крайна вероятност ще бъде избрана като окончателен резултат, взет от SAMC. Механизмът за рутиране, съобразено с доверието, в рамките на SAMC гарантира, че прогнозите ще бъдат приемани само когато и двата отделни модела са достатъчно уверени, ограничавайки вероятността от приемане на прекалено уверени погрешни класификации на нови или двусмислени нови данни.

**Внедряване и регистриране на операциите с логове**

Пълната имплементация е достъпна в Python 3.8, използвайки joblib и numpy, с описани подробности. Всеки от .exe файловете ще бъде сканиран и логът на всеки файл ще съдържа:

- Избрания модел,
- Прогнозиран етикет (Malicious or Benign),
- Увереност в прогнозата,
- ако е необходим резервен вариант, крайните вероятности се осредняват в претеглена комбинация, за разлика от старомодното „средно аритметично“.

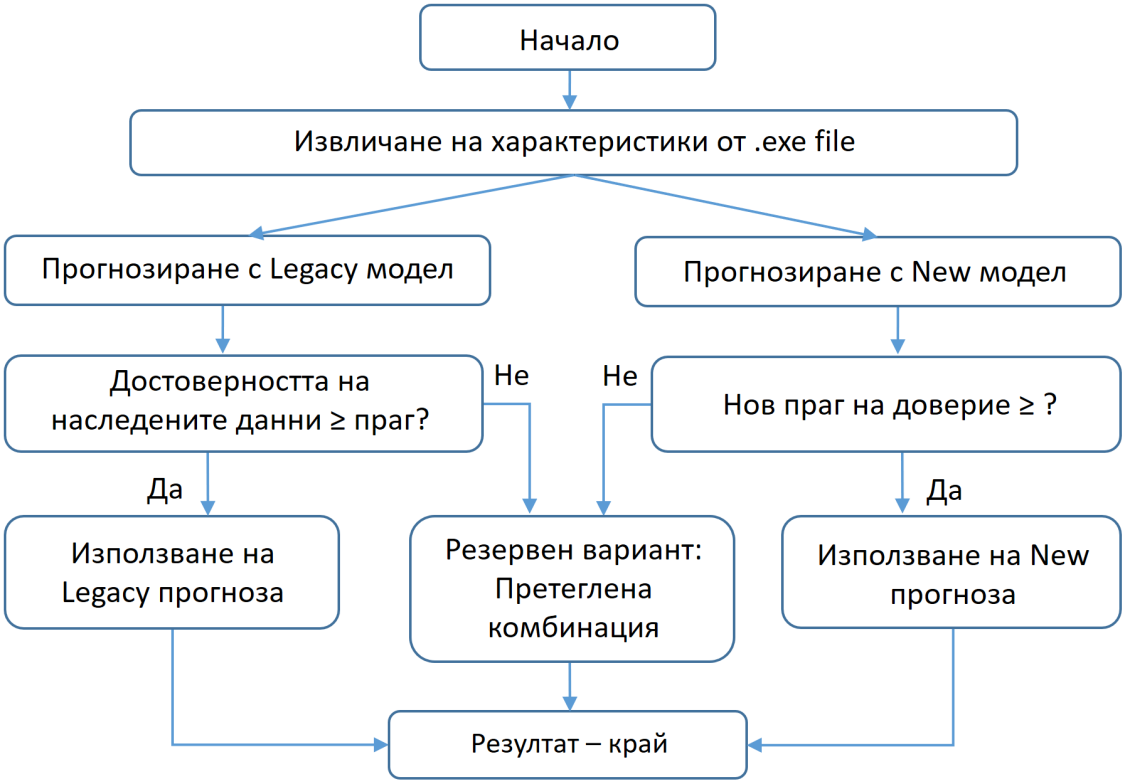
Този метод води до наличието на одитируема история на рутирането, което е от първостепенно значение за обяснимостта, отстраняването на грешки и смекчаването на риска. Цялостната карта на всички режими за сканиране и условие за вземане на решение е показана в Таблица 2.5.

*Таблица 2.5. Режими на решения за сканиране, налични модели, SHAP съвместимост и резервен метод на маршрутизиране в SAMC.*

| Режим | Модел | Праг | SHAP Enabled | Резервен вариант |
|---|---|---|---|---|
| **Legacy Mode** | EMBER 2018 | — | НЕ | Не |
| **New Model Mode** | EMBER 2024 | — | Да | НЕ |
| **Auto Mode (SAMC)** | New или Legacy или и двата | 0.85 | Да (приеминаване към нов режим!) | Weighted Combo if needed |

**Диаграма на рутиране на SAMC**

Логиката на рутиране на SAMC е визуализирана на Фиг. 2.7:



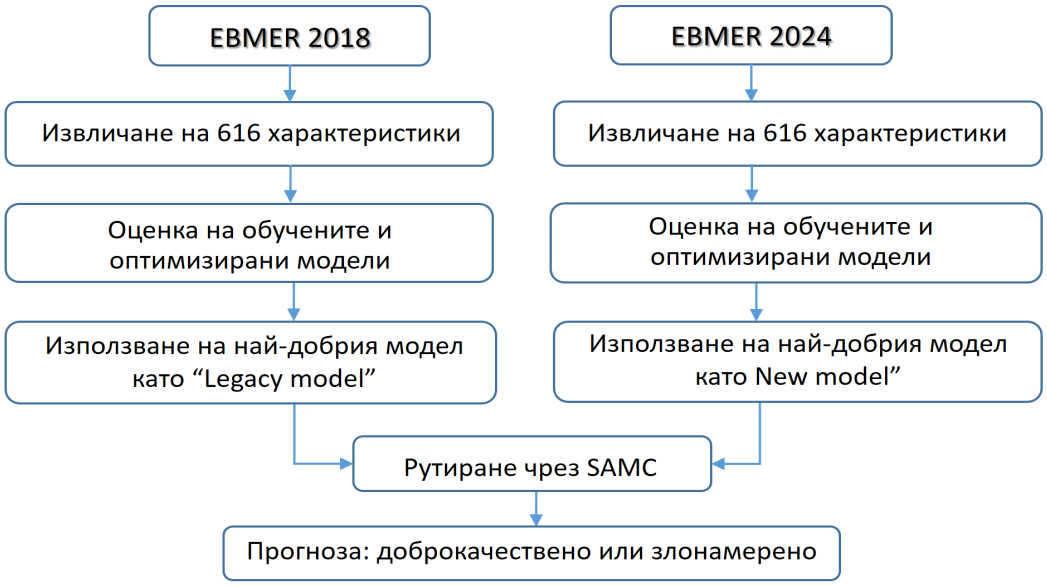

*Фиг. 2.7. Логика на рутиране на SAMC с избор, базиран на доверие, и претеглена резервна стратегия.*

От Фиг. 2.7 се вижда условието, когато рутирането се осъществява между моделите, както и решението за резервен подход в случай, че доверието не е достатъчно.

Предимства:

- Справя се с отклонението на характеристиките, като дава предпочитание на по-новия модел.
- Използва историческата устойчивост чрез наследения ансамблов модел.
- Избягва сляпото доверие: и двата модела трябва да демонстрират достатъчна сигурност.
- Проверим, обясним и модулен дизайн – разширяем до многокласов или графичен потребителски интерфейс.

Освен решенията за рутиране, SAMC внедрява опростена форма на логика за вземане на решения, основана на доверие както е показано на Фиг. 2.8.




*Фиг 2.8. Извеждане на моделите и рутиране в SAMC, използвайки EMBER 2018 и EMBER 2024.*

Тази логика проследява достоверността на прогнозите за наследените и новите модели, приемайки техния изход само когато достоверността надвишава определен праг (напр. 0,85). Когато нито един от моделите няма достатъчна сигурност в изхода си, тя използва резервно решение, базирано на техните претеглени прогнози, като на новия модел се присвоява 60% тегло, а на наследения модел – 40%. Това се прави, така че прогнозите, които показват повече от определена степен на несигурност, да не доминират в крайния резултат от класификацията. По този начин SAMC се справя по-добре с намаляването на пристрастията от погрешна класификация в крайни случаи. Този вид логика е в съответствие със съвременната работа в системи с изкуствен интелект, калибрирани въз основа на доверие, осигурявайки баланс между решителност или предпазливост при работа в реалния свят при дефиниране на характеристиките на зловредния софтуер.

**Анализ на отклонението на характеристиките (Feature Drift)**

За да предоставим пример за еволюционните характеристики на зловредния софтуер във времето, е проведен анализ на отклонението на характеристиките за наборите от данни EMBER 2018 и EMBER 2024. Въпреки че и двата набора от данни имат идентично статично пространство от характеристики с 616 измерения (характеристики на байтова хистограма, характеристики на сегменти, базирани на ентропия, характеристики на низове), абсолютните разпределения могат да варират значително поради еволюцията на зловредния софтуер, промените в използваните компилатори и

поведението на опаковане. Такива промени в разпределенията могат да навредят на прогнозите от модели, обучени върху по-стари данни, и могат да мотивират разработването на адаптивни стратегии, като SAMC.

За да се оцени степента на отклонение на характеристиките, е използван теста на Колмогоров-Смирнов (Kolmogorov-Smirnov – KS). Този тест представлява непараметричен метод за сравняване на две емпирични разпределения като измерва разлиакта между тях. За това проучване е използвана извадка от 10 000 от EMBER 2018 и Win32 частта на EMBER 2024 и са сравнени стойностите на характеристиките. KS статистиката DDD измерва максималното разстояние, наблюдавано между двете емпирични кумулативни функции на разпределение. Тоест, по-високата стойност на DDD показва по-голямо отклонение, докато p-стойностният тест показва статистическата значимост на KS теста.

В резултат на проведено изследване се установи статистически значимо отклонение по много характеристики. В Таблица 2.6 са обобщени резултатите за 10-те характеристики с най-висока статистика на KS теста.

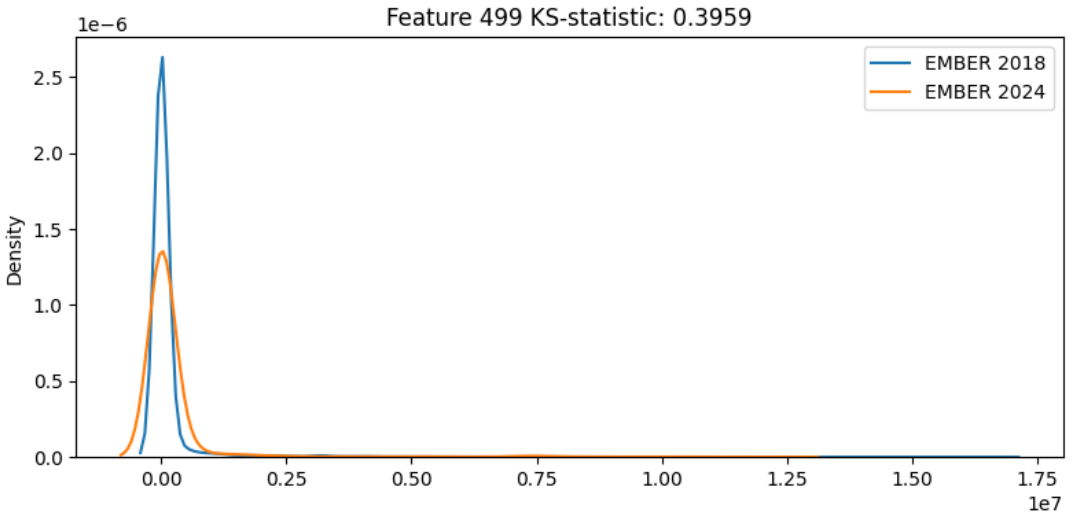
Тези резултати показват, че на ниво на статистическа значимост, се променят в много измерения на пространството от характеристики, което е особено вярно за характеристики с по-голям индекс, които вероятно представляват разпределения на низове или разпределения на ентропия с различни стойности. Стойностите на KS от около 0.39–0.40 показват значителна разлика в характеристиките за двата набора от данни.

*Таблица 2.6. Топ 10 характеристики със статистика за тестове на KS.*

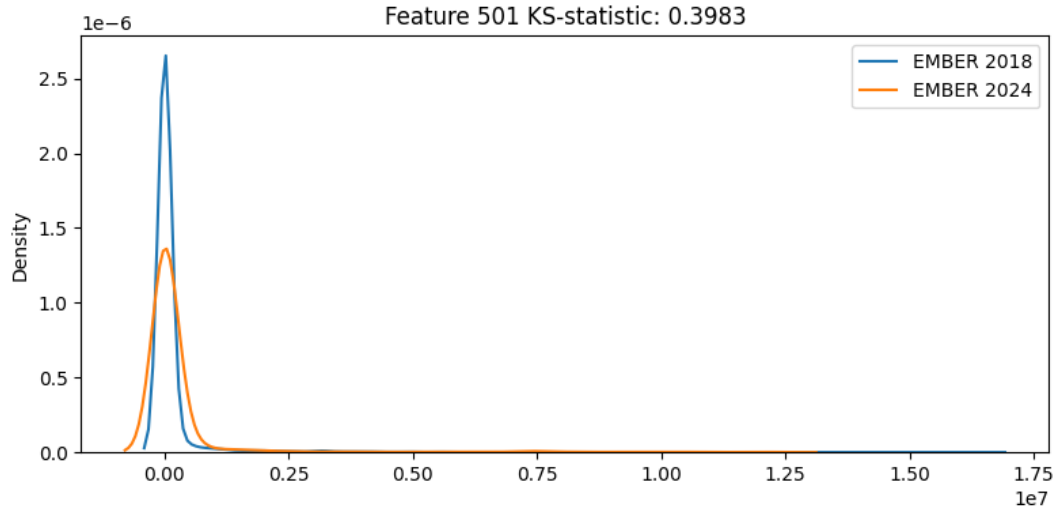
| Feature Index | KS Statistic | p-value |
|---|---|---|
| **Feature 501** | 0.3983 | < 1e-10 |
| **Feature 507** | 0.3976 | < 1e-10 |
| **Feature 506** | 0.3971 | < 1e-10 |
| **Feature 499** | 0.3959 | < 1e-10 |
| **Feature 508** | 0.3959 | < 1e-10 |
| **Feature 511** | 0.3959 | < 1e-10 |
| **Feature 510** | 0.3958 | < 1e-10 |
| **Feature 498** | 0.3950 | < 1e-10 |
| **Feature 505** | 0.3945 | < 1e-10 |
| **Feature 509** | 0.3943 | < 1e-10 |

**Визуални доказателства**

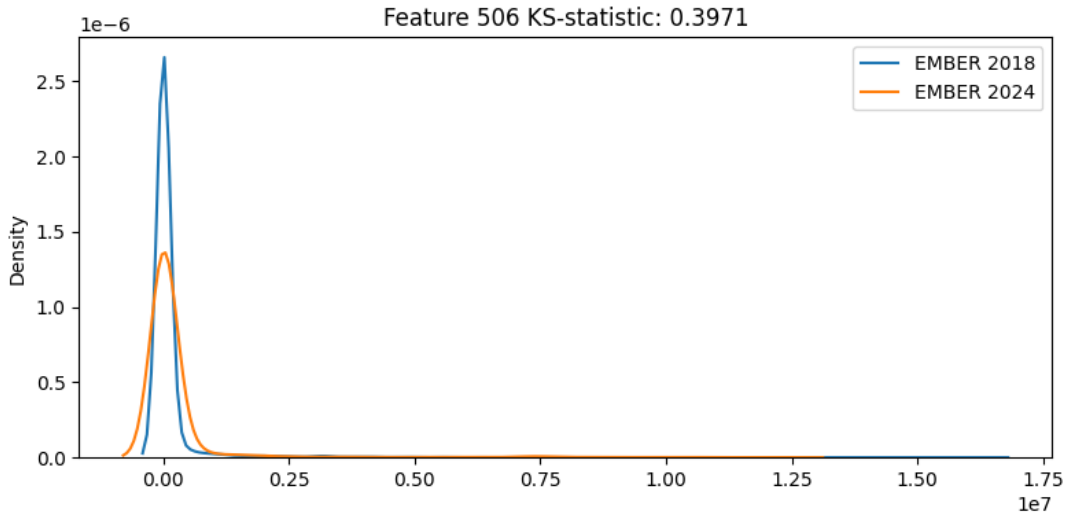
На фигурите от 2.9 до 2.13 са представени графики за оценка на плътността на ядрото (Kernel Density Estimation – KDE), подчертаващи 5-те най-отклонени характеристики.



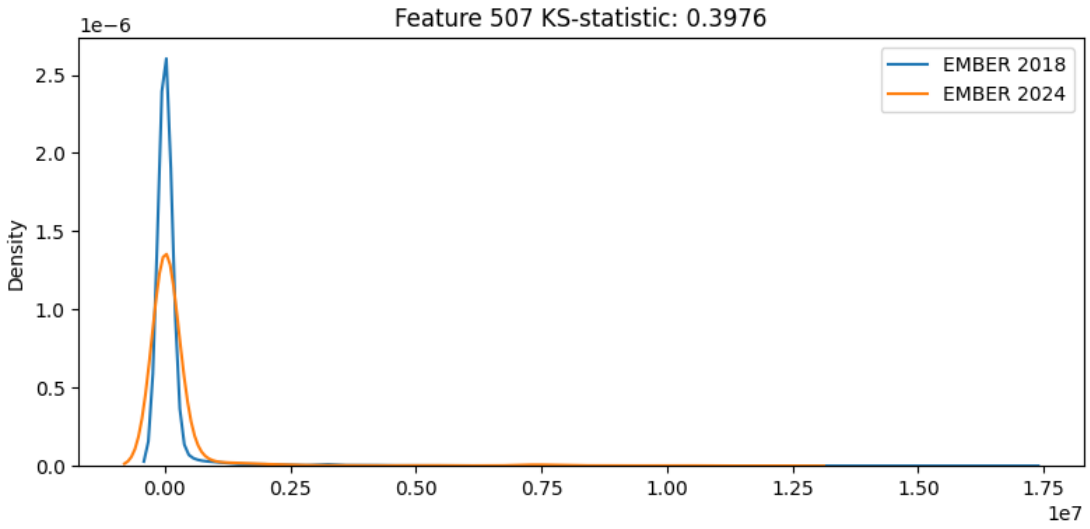

*Фиг 2.9. Feature 499 – KDE показва по-широко разпространение и изместване надясно в EMBER 2024.*



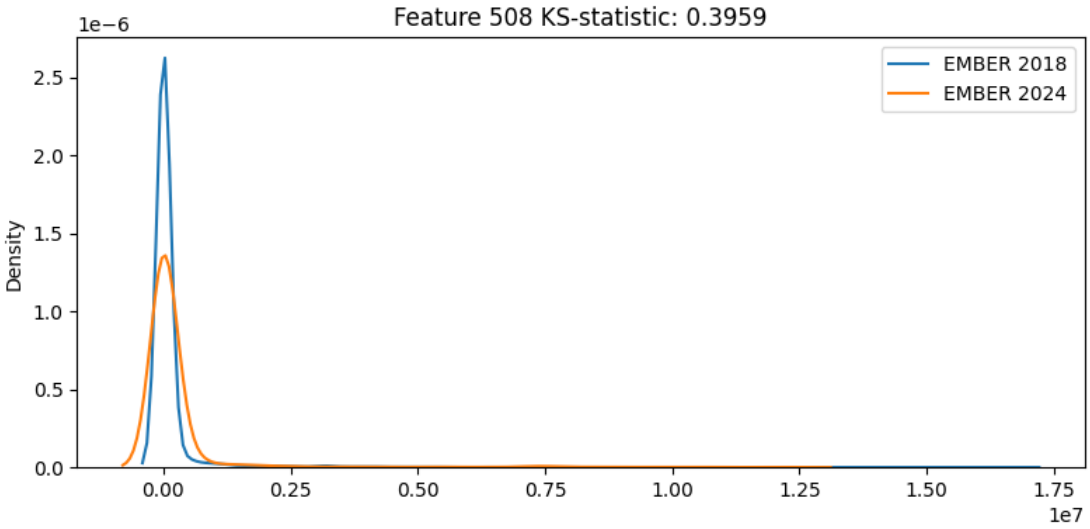

*Фиг. 2.10. Feature 501 – Подобна тенденция с увеличена средна стойност и дисперсия.*




*Фиг. 2.11. Feature 506 - EMBER 2018 показва по-висока концентрация близо до нулата.*




*Фиг. 2.12. Feature 507 – Забележимо отклонение във формата на пика.*




*Фиг. 2.13. Feature 508 – Изравнено разпределение в EMBER 2024.*

Всички тези графики визуално предоставят смислени доказателства за променените разпределения между EMBER 2018 и EMBER 2024.

**Въздействия**

Всички тези констатации потвърждават очакването, че изместването на концепциите и измъстването на характеристиките съществуват в наборите от данни за зловреден софтуер за определен период от време. Съответно, моделите на машинно обучение, които са били обучени само върху по-стари набори от данни, като EMBER 2018, са подложени на свръх увереност в тези предишни значения на характеристиките и несъвършено обобщение към варианти на съвременен зловреден софтуер. Следователно, тези констатации допълнително установяват необходимостта от архитектурна поддръжка за нашия самоосъзнат класификатор на модели (SAMC), който интелигентно би избирал между наследени и/или актуализирани модели в рамките на рамка за оперативна увереност.

В крайна сметка, анализът на изместването на характеристиките служи не само като аргумент за SAMC, но и като основа, с която да се изследват бъдещите насоки с онлайн обучение, постепенно преобучение и адаптивни системи за защита от зловреден софтуер.

**Обясним изкуствен интелект за анализ на приноса на характеристиките**

В киберсигурността е важно да се разбере вътрешната логика на решенията за моделите за машинно обучение, когато обикновено искаме да можем да се доверяваме и да одитираме прогнозите на модела. Прилагането на инструменти с обясним изкуствен интелект (XAI) позволява да разберем производителността на най-точния модел за откриване на зловреден софтуер, обучен върху набора от данни EMBER 2024, класификатора XGBoost (настроен с помощта на Optuna).

За да се анализира ефекта на всяка от входните характеристики върху изходното решение на модела, ние прилагаме SHAP (SHapley Additive exPlanations), подход, основан на теорията на игрите, който осигурява глобална и локална интерпретируемост. SHAP генерира оценки за принос за всяка характеристика (SHAP стойности) и можем да визуализираме средния принос и разпределението на ефектите на отделните характеристики върху прогнозите на модела за машинно обучение.

**Процедура за SHAP анализ**

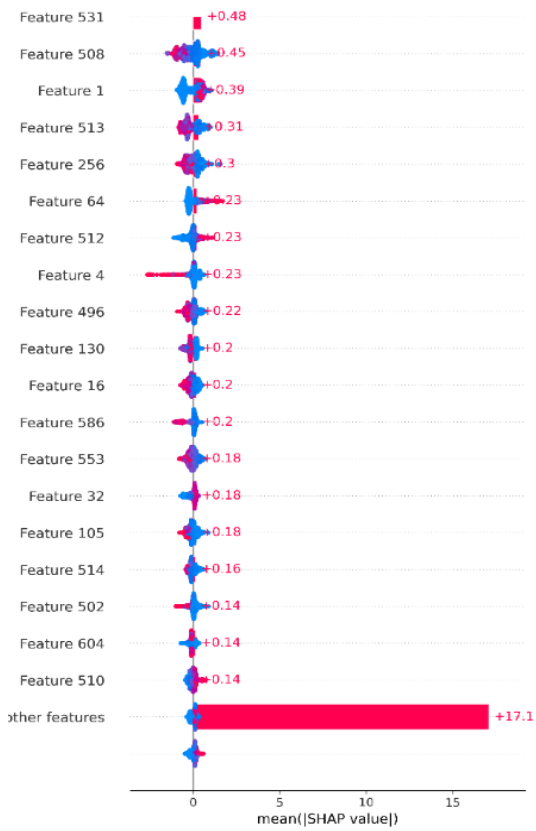
- Анализиран модел: XGBoost (Optuna), обучен със 100% от набора от данни EMBER 2024 Win32, съдържащ 616-мерни вектори
- Използван explainer: *shap.TreeExplainer(model)*
- Входни данни: пълни характеристики за обучение matrix *X_train_full.npy* (1.56 млн. проби)
- Инструмент за визуализация: *shap.summary_plot()* и *shap.plots.bar()*

**Забележителни резултати**

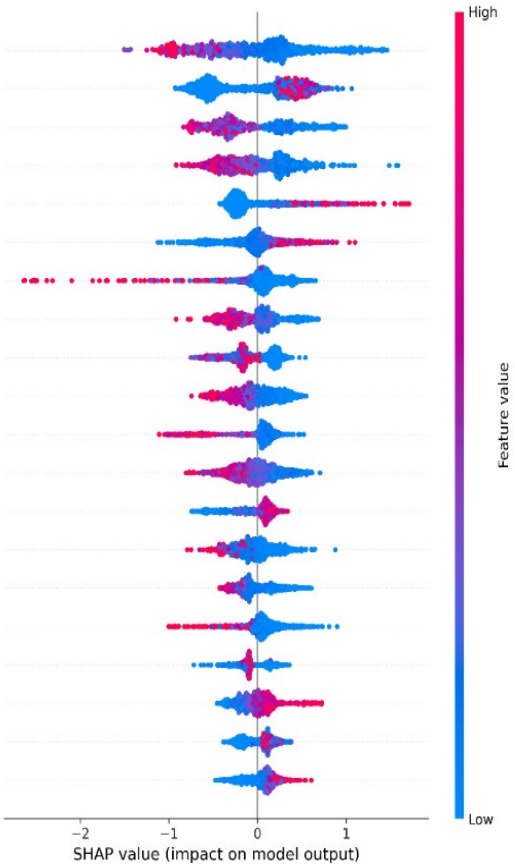
SHAP стълбовидната диаграма (Фиг. 2.14) показва приноса на характеристиките, класирани по средна абсолютна SHAP стойност, което показва ефективната сила на приноса. Резултатите показват, че първите 10 характеристики са предимно от секции на байтова хистограма и ентропийна хистограма. Това допълнително подчертава, че статистическите аспекти на двоичното съдържание продължават да бъдат силен дискриминатор за съвременната класификация на зловреден софтуер.

Обобщаващата SHAP диаграма, показана на Фиг. 2.15, предоставя допълнителни подробности за важността на характеристиките и разпределението на стойностите за всяка характеристика. Всяка точка представлява единична примерна SHAP стойност, чийто цвят съответства на действителната стойност на тази характеристика (червено = високо, синьо = ниско). Това показва дали големите или малките стойности на дадена характеристика тласкат прогнозата към злонамереност.

От Фиг. 2.14 и Фиг. 2.15 може да се установи, че Feature #507 и Feature #506, които преди това идентифицирахме в анализа на изместването на характеристиките като силно изместени, също са едни от най-влиятелните SHAP характеристики. Тази корелация предоставя допълнителни доказателства за забелязания дрейф на характеристиките и оправдава използването на адаптивни модели, като SAMC.




*Фиг. 2.14. SHAP стълбовидна диаграма на важността на характеристиките за целия обучителен набор.*




*Фиг. 2.15. SHAP обобщена диаграма на топ 20 характеристики за целия обучителен набор.*

### 2.4.3. Скенер за самоосъзната класификация на зловреден софтуер с графичен потребителски интерфейс

За да подобрим прозрачността, достъпността и обяснимостта на предложените модели за класификация на зловреден софтуер е необходимо да създадем графичен потребителски интепфейс. Той трябва да позволява на потребителя да сканира изпълним файл, използвайки три режима за избор:

- Manual Legacy Mode: използва оригиналния модел, обучен на EMBER 2018.
- Manual New Model Mode: използва оптимизирания модел, обучен на EMBER 2024.
- Auto Mode (SAMC): извършва интелигентен избор на модел въз основа на прагове на доверие.

**Автоматичен режим в SAMC (Auto Mode)**

Автоматичният режим реализира логиката на самоосъзнаващия се класификатор на модели (SAMC). В този режим, когато даден файл се сканира, системата трябва да изчислява вероятностите за класове въз основа както на наследения ансамбъл, така и на новите XGBoost модели. Системата оценява максималната надеждност на прогнозата от двата модела и избира модела, ако тя надвишава зададения праг (обикновено 0,85). Ако и двата модела не отговарят на този стандарт за надеждност, се активира резервен/непредвиден план с комбинация от претеглени вероятности (60% нов модел и 40% наследен модел). Крайната прогноза се прави от претегления ансамбъл, което позволява допълнителна увереност в случай на двусмислени случаи на надеждност на модела.

Подходът за рутиране на модел, съобразен с доверието, позволява на автоматичния режим да променя нивото въз основа на различните типове входни данни и да избира най-добрия модел-кандидат въз основа на доверието.

**SHAP обяснения**

Важен аспект на графичния потребителски интерфейс е възможността за създаване на локални SHAP (SHapley Additive exPlanations) обяснения на нашите прогнози за модела и показването им. Трябва да се отбележи, че поради ограниченията на SHAP визуализациите, само новият модел, базиран на XGBoost, ще позволи визуализации, базирани на възможностите на TreeExplainer.

Когато трябва да се активират SHAP обяснения и е избран несъвместим модел (например, стар модел или използване на резервен ансамбъл), графичният потребителски интерфейс автоматично ще използва новия модел, за да генерира SHAP обяснение и да маркира този избор. Този подход поддържа последователност, като същевременно минимизира всяка загуба на функционалност, доколкото е възможно.

На Фиг. 2.16 е представена илюстрация на интегрирания поток на рутиране, така както е планирано да се реализира чрез графичния потребителски интерфейс.


*Фиг. 2.16. Сканиране и рутиране с помощта на графичен потребителски интерфейс, включително логика за резервен подход, базирана на доверие.*

След като потребителят избере файл, той влиза в SAMC, след като статичните характеристики бъдат извлечени. SAMC определя кой модел (стар или нов) трябва да се използва въз основа на това колко уверена е била някоя от прогнозите или, ако някоя от прогнозите не е представила значимо ниво на увереност, каква тежест е била предоставена като резервен вариант. Системата ще информира за решението и ще предостави SHAP-базирано обяснение, ако е бил използван новият модел.

## 2.5. Адаптивна рамка, интегрираща доверие за класификация на зловреден софтуер чрез корекции за обратна връзка

Зловредният софтуер остава една от основните заплахи за информационните системи, като бързо се развива по сложност, вариации и неуловимост. Въпреки че детекторите за зловреден софтуер, базирани на машинно обучение, работят добре по време на първоначалната фаза на обучение, на практика те се борят със застоялост на модела. С трансформацията на семействата зловреден софтуер и появата на нови методи за обфускация, статичните модели стават неточни поради отклонение в концепцията и им липсва обратна връзка и адаптивност в реално време.

Бързата еволюция на семействата зловреден софтуер стимулира изследователските усилия към адаптивни и устойчиви системи за откриване (Alsubaei et al., 2025). Докато традиционните статични класификатори, моделирани върху хистограми на ниво байт, PE хедъри или импортирани функции, показват прилични резултати върху филтрирани набори от данни като EMBER, всички класификатори скоро ще претърпят концептуално отклонение в областта поради присъщите разлики в разпределението на пробите от зловреден софтуер.

Адаптивното обучение е предложено поради факта, че моделите остаряват. Онлайн и инкременталното обучение (често наричано непрекъснато обучение) позволяват класификаторите да бъдат непрекъснато актуализирани; въпреки че те обикновено изискват скъпоструващо преобучение (Zhang et al., 2021). По подобен начин са предложени федеративни рамки за обучение (например за класификатори), за да се даде възможност на страните да актуализират съвместно класификатори в разпределени крайни точки, без да се прехвърлят сурови данни (Mehdi et al., 2024). Тези рамки изглеждат достатъчно обещаващи със своята новост, но в крайна сметка биха се провалили и биха създали доста комуникационни разходи, когато бъдат внедрени в полеви условия.

В някои изследвания са обсъждани системи, основани на доверие, по отношение на киберсигурността от гледна точка на откриване на проникванията и откриване на аномалии (Pigola & de Souza Meirelles, 2024). По-специално, изследвано е използването на оценка на неопределеността (напр. Бейсово дълбоко обучение, конформно прогнозиране) за непълна оценка на доверието и намаляване на фалшивите аларми (Del Corso et al., 2025). Анализът и оценката са ограничени по отношение на класификацията на злонамерен софтуер, но по-голямата част от работата използва засилване на точността вместо калибрирано вземане на решения. Класификаторите, основани на доверие, включват концепции за калибриране, пренебрегване и въздържание, за да се заобиколят сценариите на неопределеност.

Обяснимият изкуствен интелект е от значение за приложенията за сигурност, тъй като осигурява интерпретируемост и прозрачност (Galli et al., 2024). Методите Local Interpretable Model-agnostic Explanations (LIME) и SHapley Addi-tive ExPlanations (SHAP) могат да се използват за подчертаване на характеристиките, които са най-влиятелни при определяне на класификацията за един случай (Salih et al., 2024). XAI е особено

важен при откриването на зловреден софтуер, защото позволява на анализатора по сигурността да провери дали автоматизираните решения са надеждни. Важно е да се обяснят процесите на класификация, като например дали става въпрос за правило, големина на зловреден софтуер или нещо по-стабилно. SHAP и LIME са способни да предоставят локално и глобално разбиране за класификаторите на зловреден софтуер (Roshinta & Gabor, 2024). SHAP предоставя атрибути на характеристики за дървовидни методи, които предоставят глобални прозрения, докато LIME предоставя локални обяснения, които са независими от модела. И двете помагат на анализаторите да валидират автоматизирани решения и са полезни, за да помогнат на агентите да изградят доверие в своите работни процеси за сигурност.

Съществуват някои трудности при прилагането на машинно обучение при откриване на зловреден софтуер, свързани с отклонението на характеристиките (Abusnaina et al., 2025; Garcia et al., 2023). Отклонението на характеристиките е опасно, тъй като промените в поведението на зловредния софтуер могат лесно да доведат до погрешни класификации. Изследването на отклонението на характеристиките е подходящо и за идентифициране на злонамерен трафик (Luo et al., 2024). Съобщава се за значителна загуба на производителност на статични модели при тестване върху времево различни набори от данни, което показва необходимост от онлайн или адаптивни и базирани на обратна връзка модели (Ali et al., 2024). За да се подобри статичният анализ, се предлага подход за откриване на зловреден софтуер чрез оптимизиране на извличането на характеристики, комбиниращ различни алгоритми за машинно обучение (Barzev & Borissova, 2025). Въпреки това все още няма интегрирана или ориентирана към потребителя система, включваща адаптация, обяснение и доверие като обща платформа за започване на изследователско проучване.

Традиционните модели за статично откриване използват ръчно извлечени характеристики, получени от PE - например байтови хистограми, ентропии на секции, таблици за импортиране и експортиране, низови статистики и т.н., както и класификатори като Random Forests, XGBoost и градиентно-усилено дърво (Imani et al., 2025). Тези методи демонстрират високо ниво на точност при бенчмаркове, но когато се въведе изместване на разпределението, резултатите варират.

Човешката обратна връзка (етикети, решения за триажи) е практичен и ефикасен начин за осигуряване на непрекъснат надзор. Корекциите от човешки анализатори

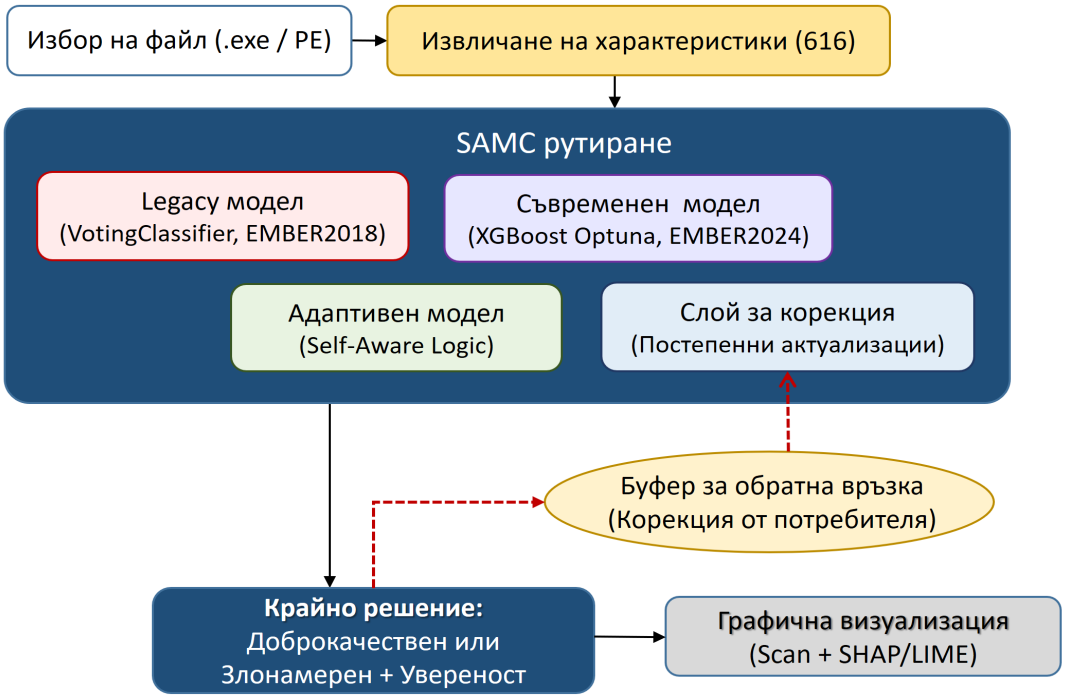
осигуряват подкрепа в обучителния процес, която може да бъде изрично настроена към често срещаните грешки, като същевременно намалява оперативните фалшиви положителни/отрицателни резултати и ефективно премахва незначителния шум от сигнала. Това, съчетано с периодично документирани одобрения от одити на моделиране, осигурява постепенно увеличение на ефикасността, без тежестта на необходимостта от пълно преобучение.

Алгоритмите за онлайн обучение и схемите за постепенно актуализиране се борят с отклонението, като актуализират параметрите на модела, когато станат достъпни нови етикетирани данни. При практическото внедряване е наложително да се балансира необходимостта от бърза адаптация с необходимостта от последователност във времето. Наивните инкрементални актуализации могат да доведат до катастрофално забравяне, докато преобучението на системата консумира ресурси. Хибрид от модели за задържане, вградени корекции и периодично актуализиране на системата, в крайна сметка отговаря на тази нужда.

Всичко това ни мотивира да разработим демонстративна версия на адаптивна рамка, базирана на доверие, за класификация на зловреден софтуер с обратна връзка и корекции. Тази рамка използва федеративна архитектура и хибриден статично-динамичен подход за анализ, наречен Shipka Guard. Рамката разчита на няколко модела на поколение (Legacy, Modern, Adaptive), слой с корекции, базиран на обратна връзка, и обясними AI компоненти, всички достъпни чрез потребителски интерфейс, базиран на Streamlit. Когато моделът не е уверен, вместо да се налага двоичен избор, предлагаме пробата да се маркира като несигурна и или да се накара човешки анализатор да прегледа несигурната проба, или да се използва резервен вариант. Прилагането на тези концепции намалява сериозните грешки, особено в области, свързани със сигурността.

### 2.5.1. Проектиране и внедряване на системата

Адаптивната рамка за класификация на зловреден софтуер с обратна връзка, Shipka Guard, е реализирана чрез интегриране на наследени и съвременни модели с адаптивен слой, базиран на обратна връзка. Системата е организирана в няколко взаимосвързани компонента, които са илюстрирани на Фиг. 2.17.




*Фиг. 2.17. Компоненти на Shipka Guard, показващи извличане на функции, SAMC рутиране, буфер за обратна връзка, слой за корекции и интеграция с графичен потребителски интерфейс.*

Рамката следва пайплайн дизайн, при който качените PE файлове се обработват чрез извличане на характеристики, избор на модел (SAMC рутиране), опционално пачване на корекции и вземане на решения. Потребителската обратна връзка се събира непрекъснато и интегрира, за да се подобри устойчивостта на модела с течение на времето. Конфигурационните параметри са събрани в един конфигурационен файл, *CONFIG.yaml*, а именно:

$$trust\_threshold: 0.85; patch\_trigger: 10; feedback\_limit: 10000;$$

$$models\_dir: "models"; feedback\_dir: "feedback"; logs\_dir: "logs".$$

Конфигурационният файл определя ограничението на прага на доверие на потребителя, броя на пробите в буфера за обратна връзка преди преобучението на съществуващите модели, броя на новите проби за пълно преобучение, модели, обратна връзка, логове и др. Тази част от функционалността създава отделяне на конфигурационните параметри от изходния код, което помага за преносимостта и повторяемостта в други системи.

Shipka Guard разчита на стандартизиран 616-измерен вектор от характеристики, който се извлича от всеки PE изпълним файл чрез PEFeatureExtractor. Екстракторът комбинира три компонента: (1) Байтови хистограми (разпределение на суровите

байтови стойности); (2) Байтови ентропийни хистограми; и (3) Метрики, базирани на низове (брой, средни дължини, съотношения на изведими символи и ентропия). 616-измерният статичен вектор на характеристиките е еднороден и максимизира сравнимостта между наборите от данни. Подравняването на характеристиките между EMBER 2018 и 2024 позволява моделите лесно да бъдат подравнени за сравнимост и ансамбълът да се използва без допълнително подравняване.

Всички потребителски корекции се записват в постоянен буфер feedback.json, където всеки запис съхранява примерния SHA-256 код, името на файла, прогнозата/вероятността на базовия модел, коригирания етикет, вектора на характеристиките и времевия печат. За да се предпазим от отравяне на обратната връзка и усилване на дублирането, ние дедублираме чрез SHA-256 преди записа. Буферът за обратна връзка поддържа експортиране/импортиране (JSON) за контролирано споделяне между инсталации и изграждане на групови набори от данни. Всеки запис също така маркира крайната вероятност на модела и последните три избора на класове (злонамерен, доброкачествен или несигурен), за да се съобрази със стратегията за въздържание на системата, основана на доверие, вместо да се налага двоичен избор въпреки ниското доверие.

Системата комбинира три вида модели:

(1) Legacy model (обучен върху EMBER 2018);

(2) Modern model (обучен върху EMBER 2024); и

(3) Adaptive model (периодично преобучаван чрез обратна връзка).

Прогнозите се рутират с помощта на SAMC като:

- Ако даден модел прогнозира 0.85 или по-висока стойност, тогава той се избира за прогнозиране.
- В противен случай се използва резервния ансамбъл с тегла: Adaptive model 0.5, Modern model 0.3, Legacy model 0.2.

В автоматичен режим, моделите-кандидати се оценяват по приоритетен ред (Adaptive > Modern > Legacy). Ако някой модел прогнозира 0.85 или по-висока стойност, тогава прогнозата от този модел се взема директно. В противен случай се използва резервен ансамбъл с тегла 0.5 Adaptive, 0.3 Modern, 0.2 Legacy, като най-адаптивният модел се запазва като доминиращ, а наследеният модел - с малка роля. За да се

осигури гъвкавост и сравнимост в различните среди, предложената системата трябва поддържа четири режима на сканиране:

(1) Автоматичен – Auto (SAMC): рутиране, съобразено с доверието, с праг на доверие и резервен ансамбъл.

(2) Форсиран Адаптивен – Force Adaptive: налага използването на най-новия адаптивен модел.

(3) Форсиран Модерен – Force Modern: налага използването на модерния модел.

(4) Форсиран Наследен – Force Legacy: Принудително наследяване: налага използването на наследения модел.

Режимите поддържат както автоматизирано поведение, съобразено с доверието, така и ръчно сравнение между поколения модели.

Потребителската обратна връзка се съхранява във *feedback.json* и се хешира чрез SHA-256, така че всеки запис да е уникален, предотвратявайки дублиране. Потребителят може също да коригира фалшиво отрицателни и фалшиво положителни резултати чрез графичния потребителски интерфейс. Слоят за корекция е лек SGDClassifier, който се обучава постепенно върху потребителската обратна връзка. Неговата важност се мащабира динамично като функция на уникалните записи за обратна връзка, което гарантира, че поведението му при корекция остава стабилно при малки размери на извадката и с много валидиращи корекции теглото му се увеличава с течение на времето. По-конкретно, корекцията расте по формулата:

$$w_{patch} = min\left(\alpha, \frac{n}{n+k}\alpha\right) \tag{2.12}$$

където $n$ е броят на уникалните проби за обратна връзка, $\alpha$ е максималният принос за избраната предварително зададена настройка, а $k$ е константа на изглаждане, контролираща скоростта на растеж.

Поддържат се три оперативни предварително зададени настройки:

(1) Консервативна: $\alpha = 0.12$, $k = 300$;

(2) Балансирана: $\alpha = 0.20$, $k = 100$; and

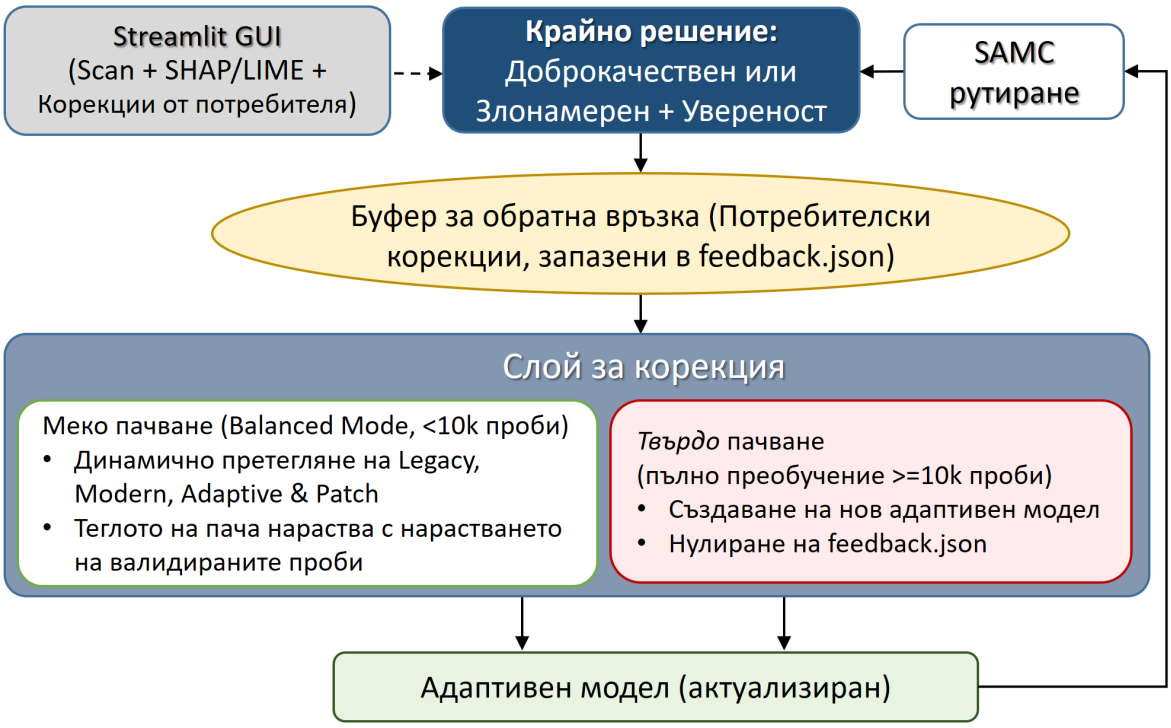
(3) Агресивна: $\alpha = 0.30$, $k = 50$.

Има и политика за повишаване на доверието – ако правилният модел е потвърден с ниско ниво (0.75), можем да увеличим теглото на patch-а с 25% (по

отношение на $\alpha$). Чрез този механизъм, пач слоят може да коригира систематични грешни класификации, без базовите класификатори да станат нестабилни.

За да се избегне нестабилност, пач слоят се преобучава само когато са предоставени достатъчен брой отделни проби за обратна връзка и когато се използва поне една проба за всеки клас. Това избягва възможността за дегенеративни модели, формирани при небалансирани корекции. Например, в случай на балансирана настройка, $\alpha = 0.20, k = 100$) при $n = 20$ пробите с обратна връзка имат незначително влияние ($w_{patch}(20) \approx 0.033$) и разчитат изцяло на базови модели. При $n = 1000$ проби с обратна връзка, пач слоят се доближава до максимално възможно влияние ($w_{patch}(1000) \approx 0.182$) като същевременно коригира систематичните грешни модели на прогнозиране.

Предвидена е и опция за синтетично FN/FP инжектиране, което помага при оценката на устойчивостта и стрес тестването на механизма на пача преди мащабно внедряване. Този дизайн позволява на пач слоя да има незначително влияние, когато са налични само няколко проби за обратна връзка, като същевременно бавно увеличава влиянието си с набирането на по-стабилни корекции (Фиг. 2.18).




*Фиг. 2.18. Адаптация на пач слоя: меко пачване (претегляне, управлявано от обратна връзка) и твърдо пачване (пълно преобучение с ≥10k проби).*

След като буферът за обратна връзка достигне до конфигурируемия праг (≥ 10 000 проби), системата инициира пълно преобучение на адаптивния модел в XGBoost. Наборът от данни за обратна връзка след това се експортира във формат JSONL за целите на възпроизводимостта и на моделът му се дава версия в директорията с модели.

# ГЛАВА 3. ЧИСЛЕНО ТЕСТВАНЕ НА ПРЕДЛОЖЕНИТЕ МОДЕЛИ ЗА ИЗБОР НА ВИРТУАЛНА МАШИНА, ХИБРИДНИ АЛГОРИТМИ И РАМКИ ЗА ОТКРИВАНЕ НА ЗЛОНАМЕРЕН СОФТУЕР

В тази глава са описани резултатите от:

- Числено тестване на предложените модели за групово вземане на решение при избор на софтуер за виртуална машина,
- Числено тестване на предложения подобрен подход на статичен анализ чрез оптимизиране извличането на характеристики, комбинирайки различни алгоритми за машинно обучение,
- Тестване на предложената рамка за статична класификация на зловреден софтуер, използваща оптимизация на функции и ансамблово обучение,
- Тестване на предложената самоосъзната класификация на зловреден софтуер чрез рутиране на модели на базата на система за доверие за избора и обяснимост на характеристиките,
- Резултати от разработеното и тествано приложение Shipka Guard на база на предложената адаптивна рамка, разчитаща на доверие за класификация на зловреден софтуер с корекции за обратна връзка.

## 3.1. Числено тестване на моделите за групово вземане на решение при избор на софтуер за виртуални машини

За целите на анализа и откриването на зловреден софтуер е необходимо да се инсталира десктоп софтуерът за виртуални машини на Windows. В тази връзка е необходимо да се направи избор между наличен софтуер за виртуални машини според някои критерии. За целите на численото тестване е използвана публикувана информация за вече оценени софтуерни продукти, които са показани в Таблица 3.1. Показаната извадка в Таблица 3.1 съдържа информация за първите 10 софтуерни продукта за виртуални машини за десктоп внедряване под Windows, класирани по най-много отзиви.

*Таблица 3.1. VM Software за Desktop Windows.*

| VM Software | Review Rating | | | |
|---|---|---|---|---|
| | ***Ease of Use*** | ***Customer Service*** | ***Features*** | ***Value for Money*** |
| **Microsoft Azure** | 4.1 | 4.2 | 4.6 | 4.2 |
| **VirtualBox** | 4.3 | 4.1 | 4.5 | 4.7 |
| **NAKIVO Backup & Replication** | 4.8 | 4.8 | 4.7 | 4.7 |
| **Altaro VM Backup** | 4.8 | 4.8 | 4.6 | 4.7 |
| **DiskStation** | 4.6 | 4.2 | 4.7 | 4.6 |
| **Ahsay Offsite Backup Server** | 3.8 | 3.4 | 4.0 | 3.7 |
| **Uranium Backup** | 4.7 | 4.7 | 4.5 | 4.7 |
| **Comet Backup** | 4.4 | 4.6 | 4.5 | 4.6 |
| **VMware Workstation Pro** | 4.6 | 4.6 | 4.7 | 4.2 |
| **Iperius Backup** | 4.1 | 4.1 | 4.1 | 4.3 |

*Note*: Data are taken from:
https://www.capterra.com/virtual-machine-software/?sortOrder=most_reviews&platform=%5B4%5D

В тази класация критериите за „ease of use“, „customer service“ и „value for money“ биха могли да бъдат лесно разбрани, но критерият, който се отнася до „features“, изисква повече внимание. Този обобщен критерий съдържа различни характеристики, които не се притежават от всички изброени алтернативи (виж Таблица 3.1) и са следните: *Access controls/Permissions; Backup and recovery; Compliance management; Configuration management; Desktop virtualization; Graphical user interface; Remote access/Control; Server monitoring; Virtual machine migration; Virtual machine monitoring; Virtual server; Virtualization*.

Въз основа на дадените ревюта е възможно да се направи подходящ избор. За да бъде изборът прозрачен и убедителен, могат да се използват предложените два оптимизационни модела, описани в раздел 2.1.1.

### 3.1.2. Резултати от числените тестване

За да се тества ефективността на предложените модели за избор на софтуер за виртуални машини, група от трима експерти, формиращи групата са представили своите мнения относно важността на четирите критерия: 1) лекота на използване; 2) обслужване на клиентите; 3) характеристики; и 4) съотношение цена-качество.

Оценките на тези критерии, показани в Таблица 3.1 са нормализирани в интервала от 0 до 1, за да се получи сравним диапазон с други коефициенти, използвани във формулираните модели. Коефициентите, които изразяват важността на критериите според мненията на водещите мениджъри, са показани в Таблица 3.2.

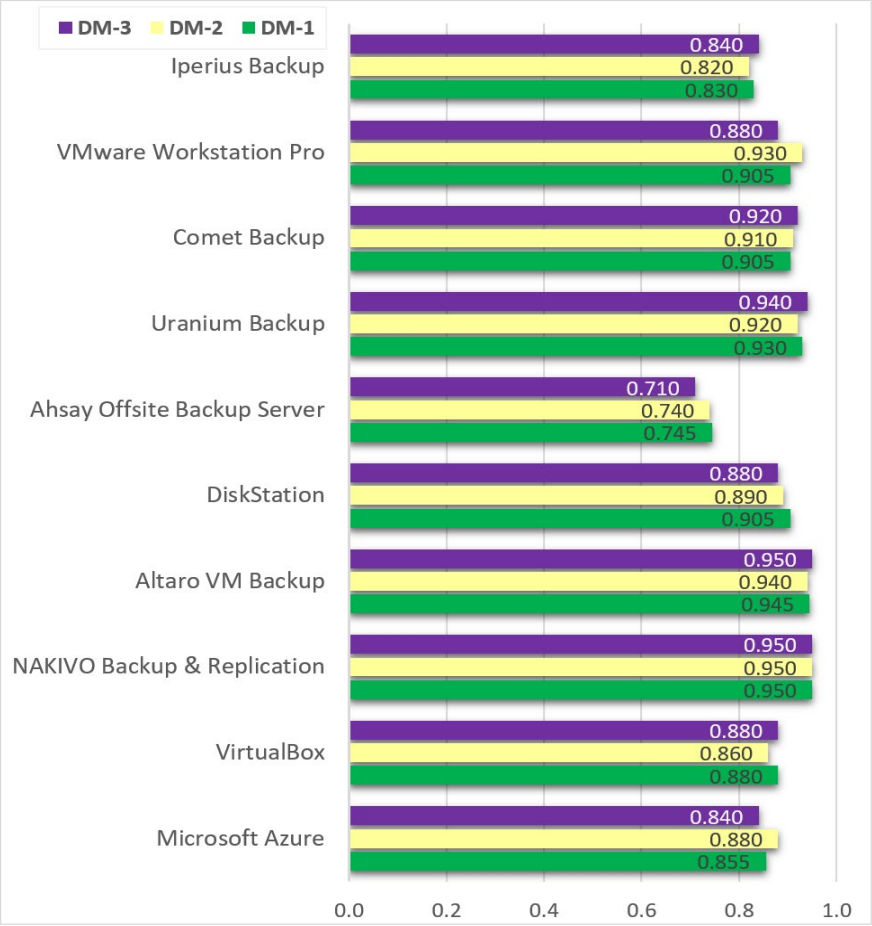
*Таблица 3.2. Коефициенти за важност на критериите според мненията на експерните.*

| Мениджъри | Коефициенти, изразяващи важността за критериите | | | |
|---|---|---|---|---|
| | ***Ease of Use*** | ***Customer Service*** | ***Features*** | ***Value for Money*** |
| **DM-1** | 0.25 | 0.25 | 0.25 | 0.25 |
| **DM-2** | 0 | 0.50 | 0.50 | 0 |
| **DM-3** | 0 | 0.50 | 0 | 0.50 |

Първият експерт, член на групата (DM-1), изразява гледната точка на главния дигитален директор, разглежда всички критерии с еднаква важност. Вторият експерт, също член на групата (DM-2) изразява гледната точка на потребителите и е фокусиран върху критериите за обслужване на клиентите и наличието на допълнителни характеристики. Третият член на групата (DM-3), изразява финансовата гледна точка и се интересува от критериите за обслужване на клиентите и съотношението цена-качество.

Използвайки данните от Таблица 3.2, формулирания оптимизационен модел (2.1) – (2.3) и нормализираните оценки в интервала от 0 до 1, може да се определи общото представяне на всички алтернативи в съответствие с различните гледни точки на членовете на групата, както е показано на Фиг. 3.1.

Използвайки само коефициенти за относителната важност на критериите за оценка според гледната точка на членовете на групата, може да се направи класиране на различните алтернативи, както е показано на Фиг. 3.1. Въпреки различната важност на критериите, решението и на тримата членове на групата определя като най-добър избор „NAKIVO Backup & Replication“. Втори в класацията е „Altaro VM Backup“. Тези резултати се дължат на по-високите оценки, дадени от членове на групата.



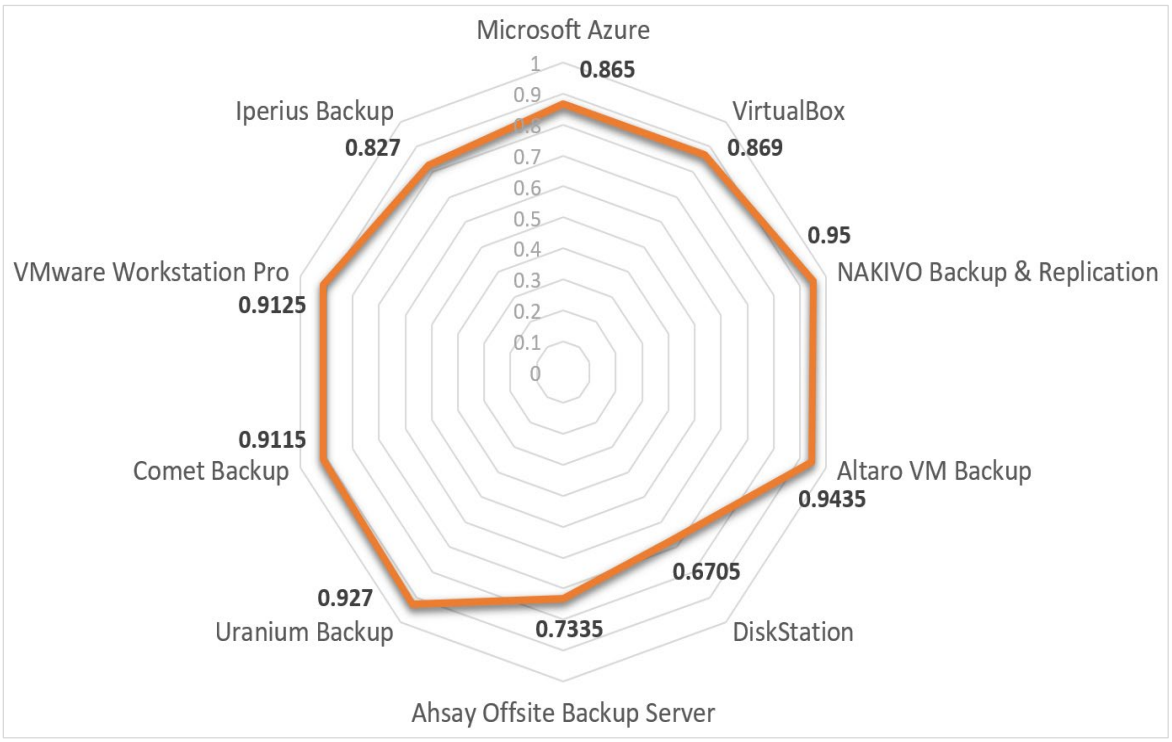

*Фиг. 3.1. Общо представяне на всички алтернативи.*

Интересно е да се покаже как крайното групово решение се влияе, когато се използват допълнителни коефициенти за мненията на членовете на групата, показани в Таблица 3.3.

*Таблица 3.3. Тегла за важност на мнението на експертите.*

| Тегла за експертите при формиране на окончателното групово решение | | |
|---|---|---|
| *DM-1* | ***DM-2*** | ***DM-3*** |
| **0.20** | 0.55 | 0.25 |

Използвайки данните от Таблица 3.1, Таблица 3.2 и Таблица 3.3, заедно с предложения модел за групово вземане на решения (2.1) – (2.3), се получава следното крайно класиране на алтернативите, както е показано на Фиг. 3.2.




*Фиг. 3.2. Класиране на алтернативите, отчитайки груповото решение.*

Формираното крайно групово решение за избора е съставено от предпочитанията на всичките членове на групата, заедно с допълнителните коефициенти от Таблица 3.3, определя „NAKIVO Backup & Replication“ като най-добър избор, тъй като неговото представяне има най-висока стойност, равна на 0.95. Това означава, че тази алтернатива е най-добрия избор съгласно използваните критерии за оценка. Втори в класацията е „Altaro VM Backup“ със стойност 0.9435, следван от „Uranium Backup“ със стойност 0.927. Възможно е този избор да се прецизира чрез намаляване на множеството от алтернативи, от които да се избира. Този процес може да се реализира чрез използване на предложения втори оптимизационен модел за групово вземане на решение (2.4) – (2.9). При този модел се формулиране оптимизационна задача, чието решение определя едновременно повече от една алтернатива, редуцирайки първоначалното множество от алтернативи. Едновременното определяне на 3, респективно на 5 алтернативи е показано в Таблица 3.4.

*Таблица. 3.4. Стойности на променливите при избор на множество алтернативи.*

| VM software | Избор на 3 алтернативи | Избор на 5 алтернативи |
|---|---|---|
| **Microsoft Azure** | 0 | 0 |
| **VirtualBox** | 0 | 0 |
| **NAKIVO Backup & Replication** | 1 | 1 |
| **Altaro VM Backup** | 1 | 1 |
| **DiskStation** | 0 | 0 |
| **Ahsay Offsite Backup Server** | 0 | 0 |
| **Uranium Backup** | 1 | 1 |
| **Comet Backup** | 0 | 1 |
| **VMware Workstation Pro** | 0 | 1 |
| **Iperius Backup** | 0 | 0 |

Изборът на 3 или 5 алтернативи се осъществява чрез присвояване на съответната стойност за константата C в (2.5). Първите 3 най-добри алтернативи, съобразно предпочитанията на членовете на групата са както следва: „NAKIVO Backup & Replication“, „Altaro VM Backup“ и „Uranium Backup“. Използвайки предложения оптимизационния модел (2.4) – (2.9), не е известно коя от тези 3 е най-добрата, но със сигурност може да се твърди, че това са първите 3 в класацията, съобразно предпочитенията на групата. За тези алтернативи решение на задачата определя следните стойности за променливите:

- $x_3 = x_4 = x_7 = 1$
- $x_1 = x_2 = x_5 = x_6 = x_8 = x_9 = x_{10} = 0.$

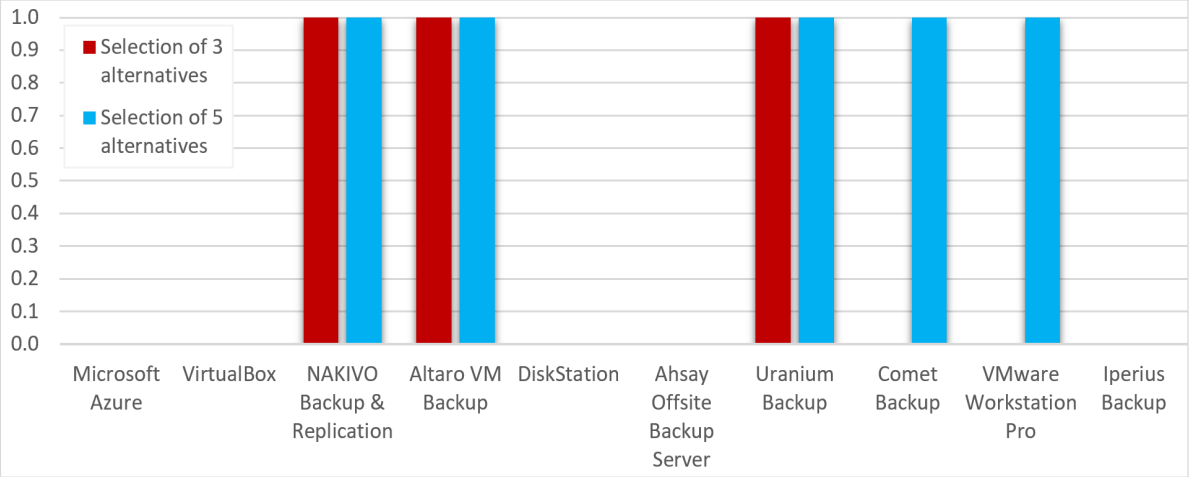
За втория случай, когато трябва да бъдат избрани 5 алтернативи, решение на задачата определя следните стойности за променливите:

- $x_3 = x_4 = x_7 = x_8 = x_9 = 1$
- $x_1 = x_2 = x_5 = x_6 = x_{10} = 0.$

При този втори случай, към вече определените 3 най-добри алтернативи се добавят още две допълнителни алтернативи, а именно „Comet Backup“ и „VMware Workstation Pro“.

Тези две ситуации, реализирани на база на предложения оптимизационния модел (2.4) – (2.9) водят до редуциране на оригиналния набор от 10

елемента/алтернативи до предварително определено подмножество от 3, респективно от 5 алтернативи, които са визуално представени на Фиг. 3.3.




*Фиг. 3.3. Общо представяне на всички алтернативи.*

Така полученото редуцирано подмножество може да се използва за по-прецизен избор на софтуер за виртуална машина, като се вземат предвид допълнителни критерии за оценка.

Предложените два математически модела реално показват, че могат да се използват успешно за избор на софтуер за виртуална машина. Чрез първия модел може да се определи най-добрата алтернатива, като вземе предвид оценките по критериите, важността на критериите и коефициенти за важност на мненията на членовете на групата. При втория модел могат да се определят повече от една алтернатива решавайки съответната оптимизационна задачата, благодарение на използването на двоични целочислени променливи. По този начин е възможно едновременно да се определят няколко алтернативи, за които със сигурност може да се твърди, че са сред най-добрите. Това е така, защото целевата функция търси максималната стойност, формирана от представянето на алтернативите. Резултатите от втория модел могат да се използват за по-прецизен последващ избор, като се вземат предвид допълнителни критерии за избор. Намаляването на алтернативите чрез използване на по-общи критерии води до намаляване на времето, необходимо на експертите да оценят допълнителни специфични характеристики.

Числените резултати демонстрират практическата приложимост и на двата предложени модела за групово вземане на решение. Изборът на подходящ софтуер за

виртуализация ще позволи провеждането на изолирани експерименти за откриване на зловреден софтуер, което ще доведе до повишена оперативна ефективност. Сложните елементи на предложентите модели са свързани с 1) определянето на експерти, които да формират групата за групово вземане на решение, и 2) определянето на коефициентите, изразяващи важността на мнението на експертите. Възможен подход за справяне с тези проблеми е да се възложи на главния технологичен директор или главния дигитален директор или главния информационен директор да оторизира подходящи експерти за конкретната задача.

Предложените модели за групово вземане на решение могат да се приложат към различни набори от алтернативи от различни области, където трябва да се направи подходящ избор. Благодарение на добре структурирания описан подход, използван и в двата модела, той може лесно да се внедри като онлайн инструмент за групово вземане на решение.

## 3.2. Тестване на предложения подобрен подход за статичен анализ за откриване на зловреден софтуер чрез оптимизиране на извличането на характеристики, комбинирайки различни алгоритми за машинно обучение

В този раздел е описано проведено тестване на подобрения подход за статичен анализ за откриване на зловреден софтуер чрез оптимизиране на извличането на характеристики, комбинирайки различни алгоритми за машинно обучение, съгласно описаното в т. 2.2 (Глава 2).

### 3.2.1. Изходни данни

Наборът от данни IoT-23, генериран от лабораторията Avast AIC с помощта на Zeek Network Security Monitor, съдържа прихванати записи с доброкачествен и злонамерен трафик от 2018 до 2019 г. Наборът от данни включва пълни и по-леки версии. Той обозначава различни типове атаки като „Доброкачествена“, „DDoS“, „Okiru“ и „Част от хоризонтално сканиране на портове“. Интересно е да се проучи дали предложените хибридни алгоритми биха могли да подобрят точността при откриване на злонамерен софтуер. Наборът от данни, описан в (Gotsev et al., 2021), изглежда е подходящ за изследване на възможностт за получаване на по-точни резултати при намиране на

злонамерена активност. За целта се използва подмножество от набора от данни IoT-23, състоящо се от 500 000 екземпляра, което включва както доброкачествени, така и злонамерени класове. Това подмножество е избрано, за да се осигури управляем размер за обработка, като същевременно се осигури представителна извадка от целия набор от данни.

**Работен процес на експеримента**

Експериментът се провежда, следвайки 5 основни етапа, както е показано на Фиг. 3.4:


*Фиг. 3.4. Работен процес на експеримента.*

Етап 1 се отнася до разбиране и четене на данни. Наборът от данни се зарежда и метаданните се изследват, за да се разберат неговата структура и съдържание. Първоначалното проучване включва идентифициране на ключови характеристики и етикети, необходими за задачата за класификация.

Следващият 2 етап се отнася до проучването на данни. Детайлна проверка на етикетите (етикет, подробен етикет) и характеристиките, за да се идентифицират симптоми на различни атаки. Анализ на корелациите на характеристиките, за да се разберат техните взаимовръзки и потенциални въздействия върху модела.

Етап 3 включва предварителна обработка на данни. Обединяване на съответните етикети в една целева колона (етикет), за да се опрости задачата за класификация. Кодиране на категорични характеристики и извършване на статистически корелационен анализ, за да се подготвят данните за обучение на модела.

След като данните бъдат предварително обработени, те трябва да преминат през етап 4 – подготовка на данни. Разделяне на набора от данни на обучителни (80%) и тестови (20%) набори, за да се гарантира, че производителността на модела може да бъде валидирана върху невидими данни. Присвояване на уникални идентификатори на всеки екземпляр за проследяване и управление на точките от данни по време на целия експеримент.

Последният етап 5 се отнася до напасването на модела. Тук е необходимо да се дефинират параметрите на модела и да се напаснат моделите към обучителните данни. Експериментиране с различни алгоритми за машинно обучение, като NB, KNN и RF.

Богатите и разнообразни данни на набора от данни IoT-23 го правят отличен ресурс за оценка на ефективността на хибридните алгоритми за машинно обучение при откриване на зловреден софтуер. Чрез използването на този набор от данни, се цели да предложи начин за подобряване точността и надеждността на системите за откриване на зловреден софтуер в IoT среди.

### 3.2.2. Анализ и обсъждане на резултати

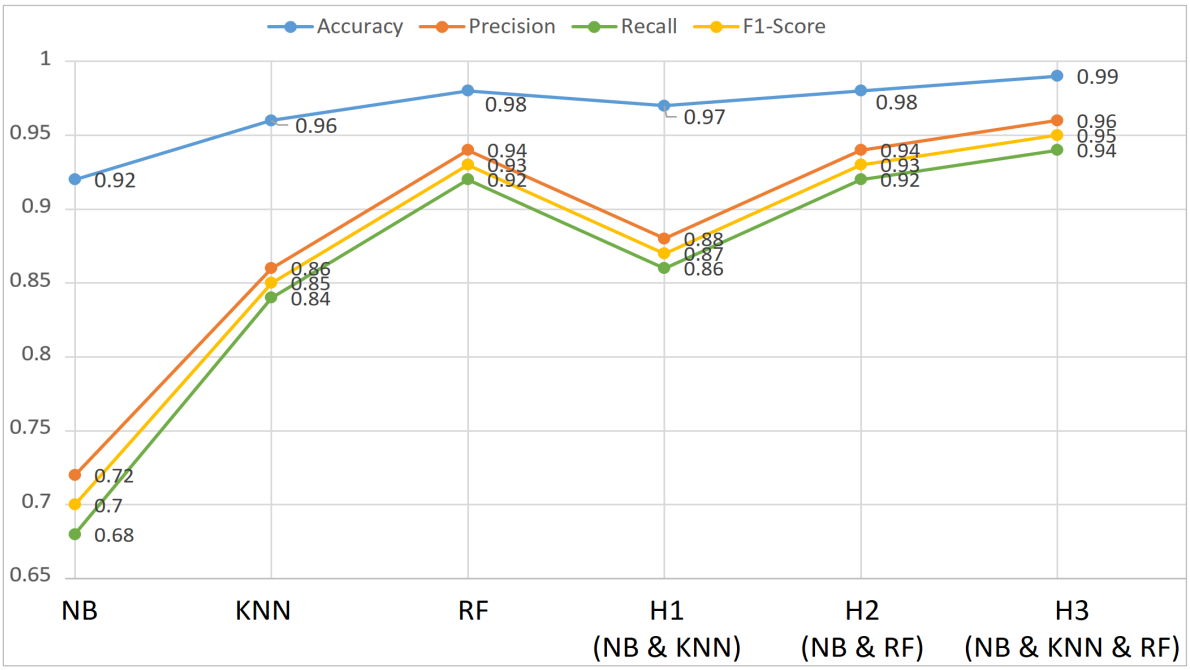
Резултатите от производителността, използващи единична имплементация на алгоритмите, и предложените хибридни алгоритми са показани в Таблица 3.5.

*Таблица 3.5. Резултати от производителността на алгоритмите.*

| Model | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| **NB** | 0.92 | 0.72 | 0.68 | 0.70 |
| **KNN** | 0.96 | 0.86 | 0.84 | 0.85 |
| **RF** | 0.98 | 0.94 | 0.92 | 0.93 |
| **H1 (NB & KNN)** | 0.97 | 0.88 | 0.86 | 0.87 |
| **H2 (NB & RF)** | 0.98 | 0.94 | 0.92 | 0.93 |
| **H3 (NB & KNN & RF)** | 0.99 | 0.96 | 0.94 | 0.95 |

Както се вижда от Таблица 3.5, някои от комбинираните алгоритми показват по-добра производителност. Сравнението между всички тези показатели и различните модели и комбинации от моделите е показано на Фиг. 3.5.

Предложените хибридни алгоритми допринасят значително за подобряване на производителността. Тази висока производителност на алгоритмите в нашето изследване за откриване на злонамерен мрежов трафик в IoT среди може да се обясни с няколко фактора. Чрез комбиниране на NB, KNN и RF, силните страни на всеки алгоритъм се използват, като същевременно се смекчават техните индивидуални слабости.




*Фиг. 3.5. Сравнение на показателите на различни алгоритми и техните комбинации.*

Например, комбинацията от NB & KNN & RF (H3) постигна най-високите показатели с точност от 0,99, прецизност от 0,96, попълнота от 0,94 и F1-оценка от 0,95. Тези комбинирани модели са по-добри от отделните модели, тъй като те обработват ефективно само различни аспекти на данните.

Получените резултати в следствие на проведените експерименти подчертават значението на оптимизирането на извличането на характеристики, което е от решаващо значение за повишаване на точността на моделите за машинно обучение. Ефективното извличане на характеристики трансформира суровите данни във формат, който е по-информативен и дискриминативен за задачата, подобрявайки процеса на обучение. Техники като намаляване на размерността се използват за намаляване на шума и подобряване на качеството на сигнала за класификаторите.

Правилните стъпки за предварителна обработка на данните, като например обединяване на етикети, кодиране и статистическа корелация, гарантират, че данните, въведени в моделите, са чисти и добре структурирани. Тази подготовка е от решаващо значение за ефективното учене на алгоритмите и правенето на точни прогнози.

Въпреки че някои комбинации не се представиха по-добре от отделните алгоритми поради небалансирани набори от данни, използването на балансирани набори от данни, където е възможно, помогна за постигане на по-добра точност и

надеждност на резултатите. Гарантирането, че данните за обучение представляват адекватно всички класове, предотвратява пристрастия и подобрява обобщението на модела.

Цялостната оценка, използваща множество показатели – точност, прецизност, попълнота и F1-оценка – предостави цялостна оценка на производителността на моделите. Тази многомерна оценка гарантира, че моделите са не само точни, но и прецизни и надеждни при откриване на истински положителни резултати, като същевременно минимизират фалшиво положителните и отрицателните резултати.
Чрез комбиниране на тези стратегии, е демонстриран надежден подход за подобряване на откриването на злонамерен трафик в IoT среди, което води до висока производителност на използваните алгоритми за машинно обучение.

Голямо внимание се обръща на откриването на зловреден софтуер, тъй като този софтуер използва различни техники, за да се скрие. Предложен е подобрен подход за статичен анализ за откриване на зловреден софтуер, базиран на оптимизиране на извличането на характеристики чрез комбиниране на някои добре познати статични алгоритми за откриване на зловреден софтуер. Резултатите показват, че е възможно да се постигне по-добра точност при комбиниране на класификатори поради използването на различни набори от характеристики. Представен е подробен анализ на производителността на алгоритмите за машинно обучение NB, KNN и RF и техните комбинации за откриване на зловреден софтуер върху набора от данни IoT-23. Предложените хибридни алгоритми показват обещаващи резултати за откриване на зловреден софтуер. Например, точността на алгоритъма H3 (NB, KNN, RF) за откриване на зловреден софтуер е 0.99, а прецизността е 0.96. Някои комбинации не се представиха по-добре от алгоритмите поотделно, тъй като наборите от данни са небалансирани.

## 3.3. Числено тестване на предложената рамка за статична класификация на зловреден софтуер, използваща оптимизация на функции и ансамблово обучение

Извличането на характеристики се извършва чрез модифициране на съществуващия Python код за характеристики от проекта EMBER, за да работи с lief 0.16.5 и Python 3.8. Стойността на ентропията, имената на секциите, елементите за импортиране/експортиране и отчетените статистически данни за метаданни, низове и др. Всеки PE файл е представен като вектор от 616 променливи.

Средата lief 0.16.5 и Python 3.8 са избрани след основно усъвършенстване, за да се отстранят проблеми със съвместимостта, за да се опита да се дублира среда, тясно свързана с времевата рамка EMBER 2018. Всяка по-нова версия от тази на LIEF или Python или не е извличала, или вече не е била съвместима с някои двоични обекти, които са били често използвани за зловреден софтуер по това време. При кодирането трябваше да променим кода си при преминаване през секции, обработка на импортиране и кодиране на планове за действие в извънредни ситуации за справяне с крайни случаи от образци на по-стария зловреден софтуер.

При подготовката на данните са включени следните стъпки:

(1) Прочитане на всички файлове train_features_*.jsonl и test_features.jsonl;

(2) Съпоставяне на хеша sha256 със съответния етикет;

(3) Изтриване на полето sha256;

(4) Създаване на масиви X_train, y_train, X_test, y_test.

Тази подготовка полага основите за всички по-нататъшни експерименти, докато изпробваме различни модели, размери на данните и нюанси на оптимизация.

### 3.3.1. CLI скенер: Практически случай на употреба

CLI скенерът е създаден специално за използване в същата среда (Python 3.8 и lief 0.16.5), за да се гарантира пълна съвместимост с извличането на характеристики на EMBER 2018 и за да се предотвратят проблеми с по-нови PE файлови структури. CLI скенерът предоставя среда от команден ред, която използва входен аргумент, предоставен от потребителя (пътят до PE файла), за да извлече характеристики (с помощта на вграден механизъм), да зареди крайния модел VotingClassifier и да получи ниво на риск. Разработеният CLI скенер е напълно офлайн, така че може да работи в

изолирани мрежи и среди (т.е. виртуални). Той е създаден в Python среда и е пакетирано .exe приложение, използващо PyInstaller. Характеристиките се извличат с помощта на lief и са точно 616. CLI скенерът е способен да генерира и поддържа лог файл, показващ подробно състояние на сканирането и вероятностни оценки.

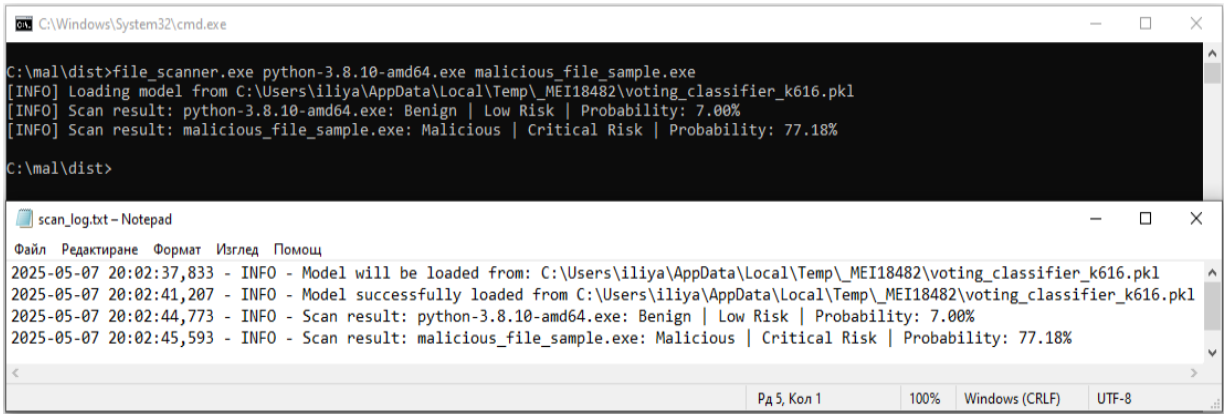
Производителността на CLI скенера е следната: Средното време за сканиране на един PE файл е ~1,0 секунди. Всеки конзолен изход се записва и регистрира, което включва всички подробности, както бихте видели в конзолния изход: дата и час, път на файла, прогнозирана вероятност, рискова оценка и краен клас. Скенерът може да обработва един файл или няколко файла едновременно (с посочени пътища), но сканирането на директории ще бъде разработено в бъдеще. Примерна команда на CLI скенера е:

```
python scanner.py --file suspicious.exe
```

Примерен изход от CLI скенера:

```
[INFO] Scan result: suspicious.exe: Malicious | Ciritcal Risk |
Probability: 95%
```

Действителен конзолен изход от CLI скенер, в който е показан пример за сканиране на 2 PE файла, един безобиден PE файл (python-3.8.10-amd64.exe) и един злонамерен PE файл (malicious_file_sample.exe). В изхода са предоставени категория риск, вероятностен резултат и запис в лога. Целият изход се записва в scan_log.txt, както е показано на Фиг. 3.6.




*Фиг. 3.6. Действителен изход от конзолата от CLI скенер.*

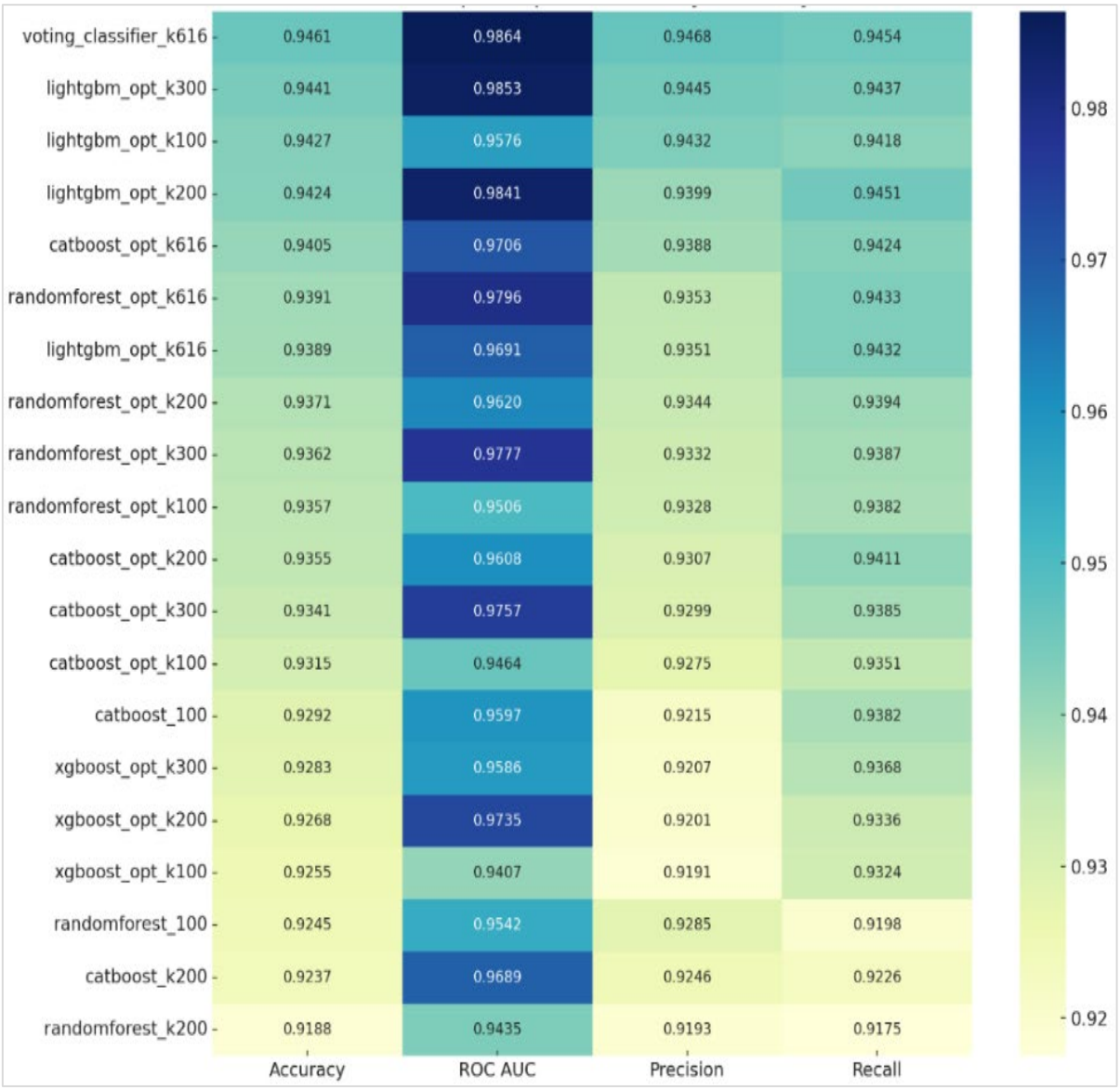
CLI скенерът идентифицира 5 нива на риск: 0: Доброкачествен; 1: Нисък риск; 2: Среден риск; 3: Висок риск; и 4: Критичен риск. Предложеният CLI скенер може да се използва от ИТ администратори на правителствени или корпоративни системи, които нямат достъп до интернет. CLI скенерът е съвместим с остарял хардуер със стари операционни системи. Този скенер може да се използва като част от автоматизация на скриптове или вътрешни защитни системи.

### 3.3.2. Визуализации и показатели

Този раздел представя колекция от визуализации, които улесняват интерпретацията и анализа на поведението на модела, изцяло от тестовия набор (20% от EMBER 2018), който не е включен в обучението или оптимизацията по никакъв начин. Тези фигури служат като допълнение към числените резултати, обсъдени в предишния раздел, и позволяват визуален сравнителен анализ между различни архитектури, нива на оптимизация и количества данни за обучение. По отношение на визуализациите, 20-те най-добри модела за визуализиране се определят въз основа на различни показатели. Визуализациите, които ще бъдат представени, включват класиране на модели, топлинни карти, сравнения по ключови показатели, фалшиво положителни/фалшиво отрицателни оценки на модели, класификации по важност на характеристиките и ROC криви. Всяка фигура ще включва кратко описание и интерпретация.

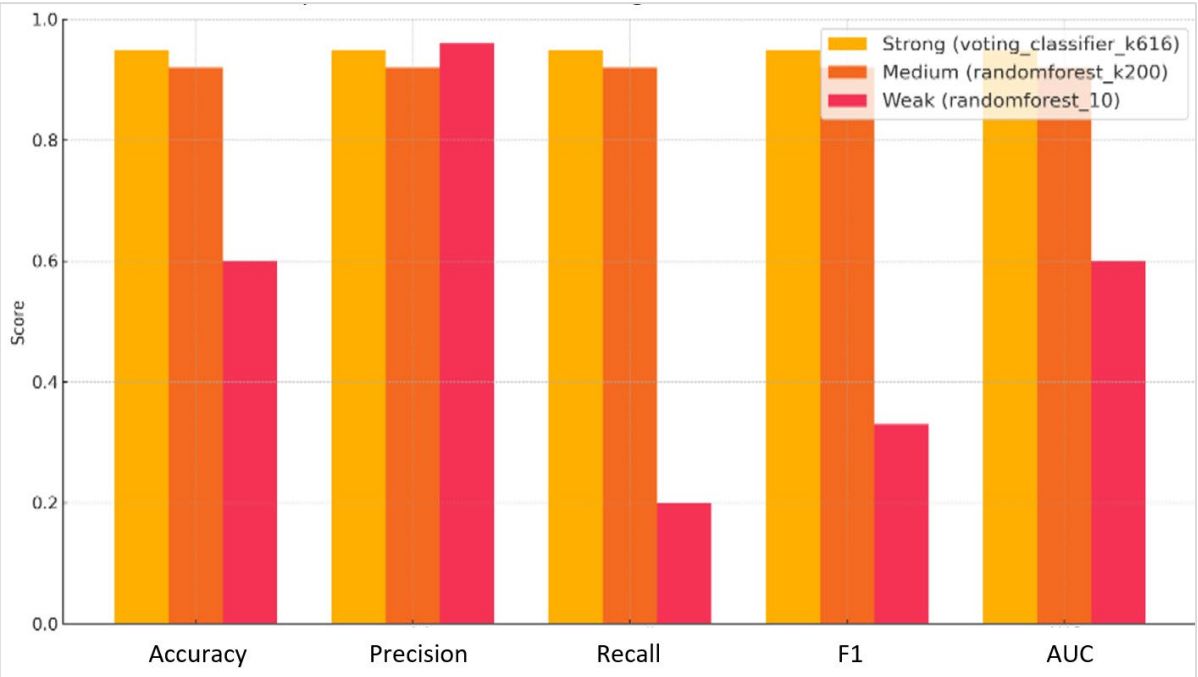
Топлинната карта на топ 20 модела е показана на Фиг. 3.7.

Тя съдържа всички основни показатели за топ 20 модела, които имат стойности за (точност, ROC-AUC, прецизност и попълнота) за 5 избрани показателя. Цветовият градиент е особено полезен, за да се види как моделите се представят с по-високи стойности спрямо моделите във всеки от 5-те избрани показателя. От тази топлинна карта е видно, че VotingClassifier е по-добър от другите модели в почти всички ситуации, така че можем да потвърдим, използвайки този показател, че моделът е надежден.



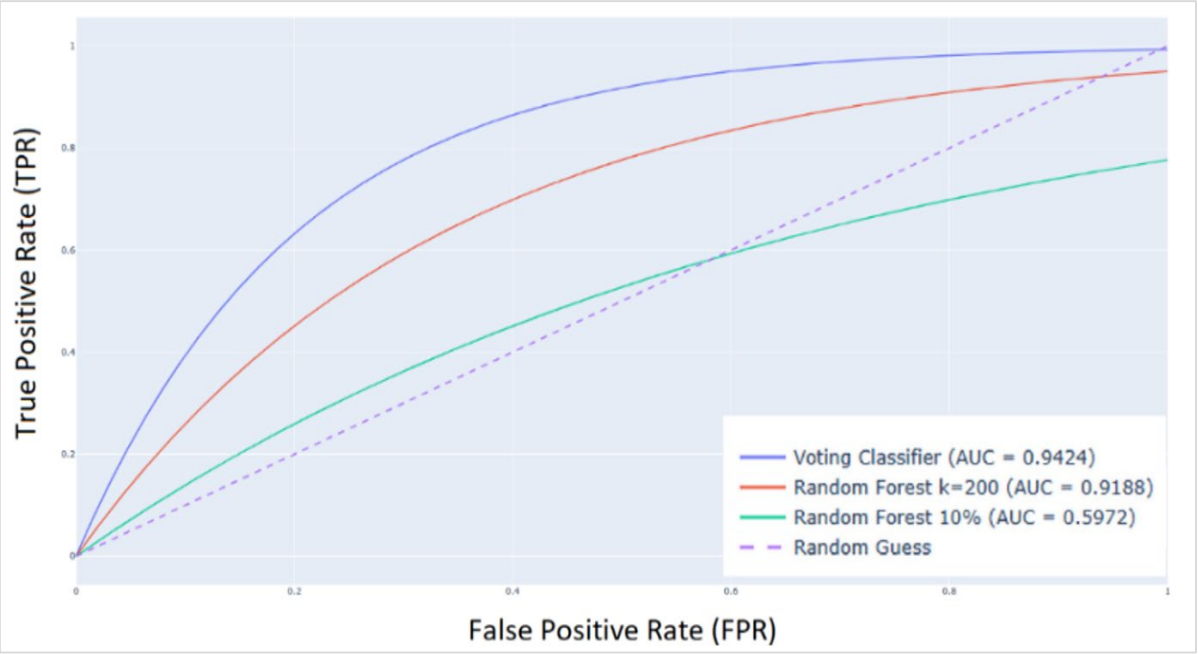

*Фиг. 3.7. Топлинна карта на точност, ROC-AUC, прецизност и попълнота за топ 20 модела.*

На Фиг. 3.8 е показана групирана стълбовидна диаграма, показваща три различни представителни модела от три различни класа на сила – силен (VotingClassifier), среден (Random Forest с k=200) и слаб (Random Forest с 10% обучаващ набор) – по отношение на пет показатели за оценка.




*Фиг. 3.8. Сравнение на силни, средни и слаби модели по accuracy, precision, recall, F1 и AUC.*

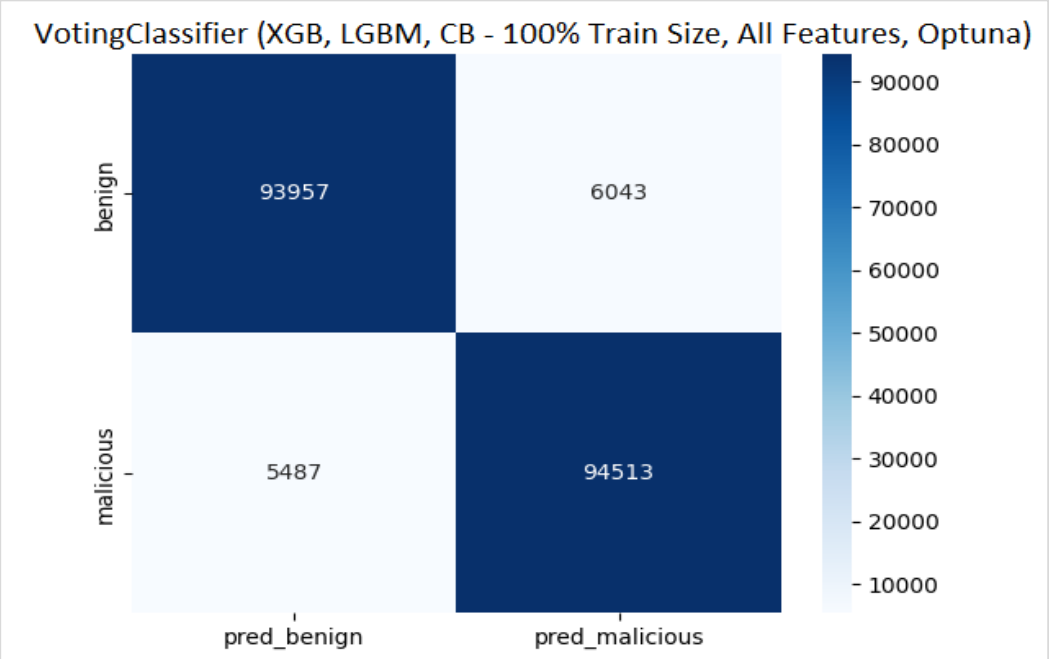
ROC кривата, показана на Фиг. 3.9, е реализирана, за да се видят резултатите от силен модел (VotingClassifier), среден модел (Random Forest с k=200) и слаб модел (размер за обучение на Random Forest, 10%). AUC на VotingClassifier е най-висок (0.9424); слабият модел е малко по-добър от случайното предположение (0.5972).




*Фиг. 3.9. Сравнителни ROC криви.*

Разликата в AUC като цяло показва, че размерът за обучение е по-малко мощен от оптимизацията върху способността за класификация.

Матрицата на объркването, показана на Фиг. 3.10, илюстрира истинските и фалшивите положителни, както и истинските и фалшивите отрицателни резултати от VotingClassifier, изпълнени общо върху тестовия набор.

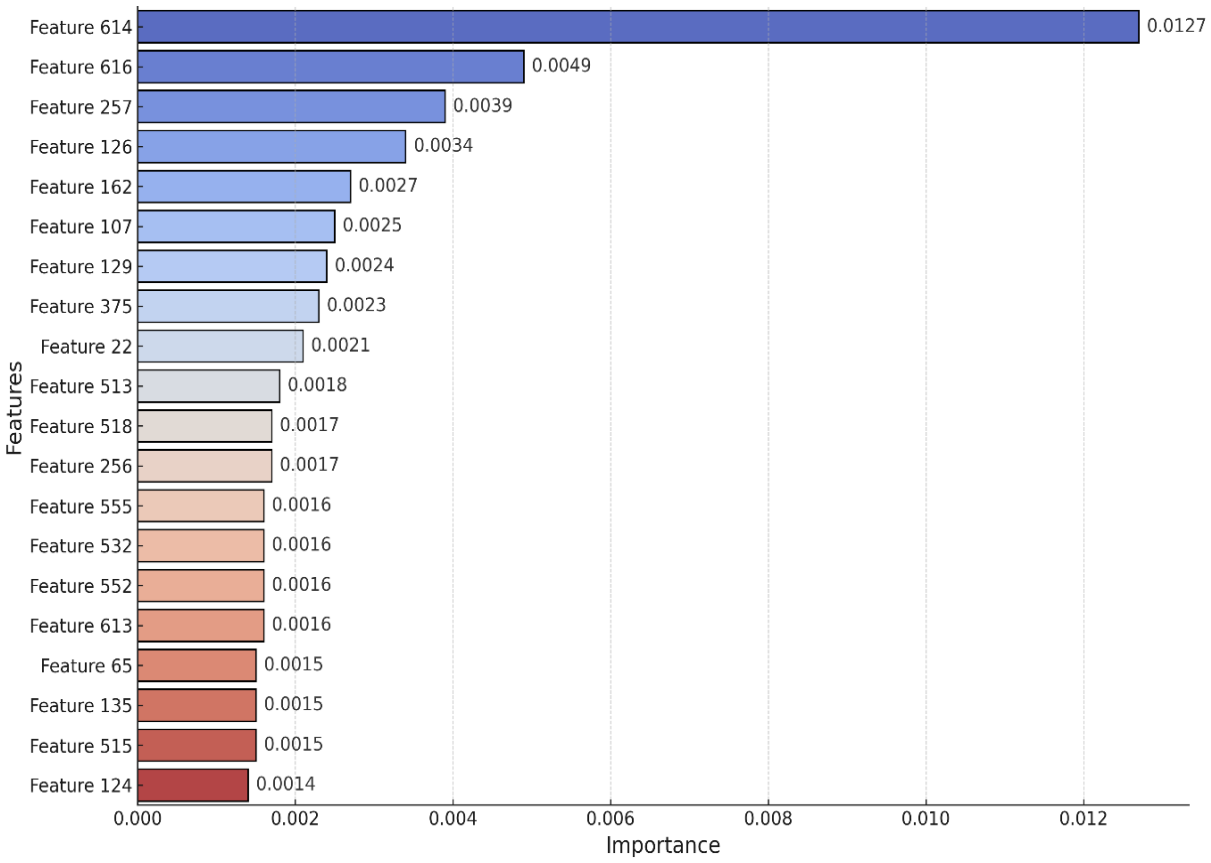



*Фиг. 3.10. Матрица на объркване за модела VotingClassifier.*

Общата производителност при прогнозирането започва да показва балансирано представяне, по-малко грешки и матриците на объркването са полезни за разбиране на поведението на една система в реалния свят.

На Фиг. 3.11 са илюстрирани 20-те най-влиятелни характеристики, базирани на VotingClassifier.

Характеристиките са подредени от най-голямо към най-малко влияние като идентификатор на характеристика (базирано на низходящи стойности), където характеристика 614 е най-влиятелната. Диаграмата е за обяснимост на модела и показва кои характеристики са отговорни за най-голямо влияние върху решенията.




*Фиг. 3.11. Най-влиятелните 20 характеристики, базирани на VotingClassifier.*

Поради естеството на набора от данни EMBER 2018, имената на характеристиките са представени само като числови (Характеристика 614 и т.н.), тъй като семантичните етикети за характеристиките не са публично достъпни. Емпиричната стойност на важността обаче дава важно разбиране за това как работи моделът.

### 3.3.3. Дискусия и резултати

Експериментите с по-малки обучителни набори (10% и 20%) наблюдавахме много по-голяма вариабилност, както и свръхнапасване на модела в случаите, когато се базираме по-специално на CatBoost и XGBoost. Това също илюстрира необходимостта от регуларизация или поне по-пълно подмножество, за да се получат приемливи състояния на прогнозиране. Тестването на SelectKBest (k=100/200/300) показа, че някои модели са особено чувствителни към намаляване на характеристиките; XGBoost е особено засегнат от прекомерно намаляване.

Оптимизацията на Optuna показа, че по-малките размери на обучение често водят до хиперпараметри, които са агресивно добре обучени спрямо базата, но лошо

обобщаващи за тестовия набор, като е доказано свръхнапасване. Важно е да се отбележи, че производителността на VotingClassifier се оценява само с помощта на данни от предварително разделения тестов набор (20% от EMBER), които не се използват за обучение и следователно не са виждани преди. Това гарантира безпристрастност на измерването. Резултатите ясно показват предимствата на използването на VotingClassifier пред единичните модели. Всички показатели (точност, F1, AUC, прецизност, попълнота) постигнаха постоянно висока производителност, където VotingClassifier постигна точност от 94,61%.

CatBoost и Random Forest независимо постигнаха добра производителност без избор на характеристики и се подобриха допълнително след използване на Optuna за оптимизация. LightGBM, като най-бързият модел за обучение, има уникално предимство в ситуации, където изчислителните възможности са ограничени. Моделите могат да бъдат ускорени чрез намаляване на характеристиките със SelectKBest, без да се ограничава точността – особено k=200/300. VotingClassifier, комбиниращ модели с различен брой на характеристиките, е непрактичен. Поради това се използва ансамбъл с унифициран вход от 616 характеристики. Оценява се само един набор от данни (EMBER 2018), което ограничава възможността за обобщаване на тези резултати. Скенерът извършва само статичен анализ; динамичното поведение не се взема предвид. Многокласовата класификация (напр. троянски кон, червей) все още не е внедрена.

FP/FN анализът на фалшиво положителните резултати за ансамбъла е значително по-нисък от този на отделните модели. Визуализацията на важността на характеристиките показва, че относително малко характеристики имат съществена роля за точното прогнозиране. Въпреки че EMBER 2018 предоставя надежден бенчмарк, възрастта му може да доведе до пристрастия в набора от данни. Използването на по-нови набори от данни в бъдещи експерименти ще помогне за оценка на обобщаемостта на предложената система.

Предложената система за автоматизирана класификация на PE файлове се счита за високоефективна, особено благодарение на ансамбъла (VotingClassifier), който комбинира четири оптимизирани алгоритъма за машинно обучение. Като цяло, експерименталните резултати потвърждават, че висока точност и устойчивост при всяка задача за откриване на зловреден софтуер може да се постигне чрез комбиниране на

човешкия опит в статичния анализ с обективно настроени класификатори за машинно обучение.

Ключово откритие в това изследване е използването на исторически данни - по-специално наборът от данни EMBER 2018. Въпреки че този набор от данни служи като добър, стандартен основен набор от данни, структурите и техниките, които той включва, може вече да не са истинско представяне на текущия зловреден софтуер. Това води до концепцията за отклонение на характеристиките, което предполага, че статистическите свойства и характеристики на зловредния софтуер могат да се променят с течение на времето. Това също означава, че производителността на всеки модел вероятно ще се влоши, особено в производствени ситуации, където непрекъснато се появяват нови семейства зловреден софтуер и техники за избягване. Така че, за да се запази ефективността на откриването постоянно във времето, в това изследване трябва да бъдат включени механизми за идентифициране и облекчаване на отклонението. Възможните решения могат да включват проследяване на разпределението на характеристиките, периодично актуализиране на обучителния набор или канали за преобучение, съобразени с отклонението.

Най-значимият принос на цялата система е практическият офлайн, CLI-базиран скенер, който може да се внедри в изолирани и наследени среди и осигурява по-бърз и интерпретируем резултат, отколкото анализът на офлайн PE, използвайки цял обучен ансамблов модел. Всеки компонент е оптимизиран за възпроизводимост, модулност и разширяемост.

## 3.4. Резултати от тестването на предложената самоосъзната класификация на зловреден софтуер чрез рутиране на модели на базата на система за доверие за избора и обяснимост на характеристиките

Този раздел орисвава проведените тестове чрез предложената рамка за откриване на зловреден софтуер, базирана на SAMC, измерваща производителността и оперативното поведение, съгласно описанието в Глава 2, раздел 2.4. Тук се представят не само проведените числени резултати, но и се анализира гъвкавостта на рамката, както и как разсъждения са обясними и по какъв начин рамката осигурява доверие и адаптивна промяна.

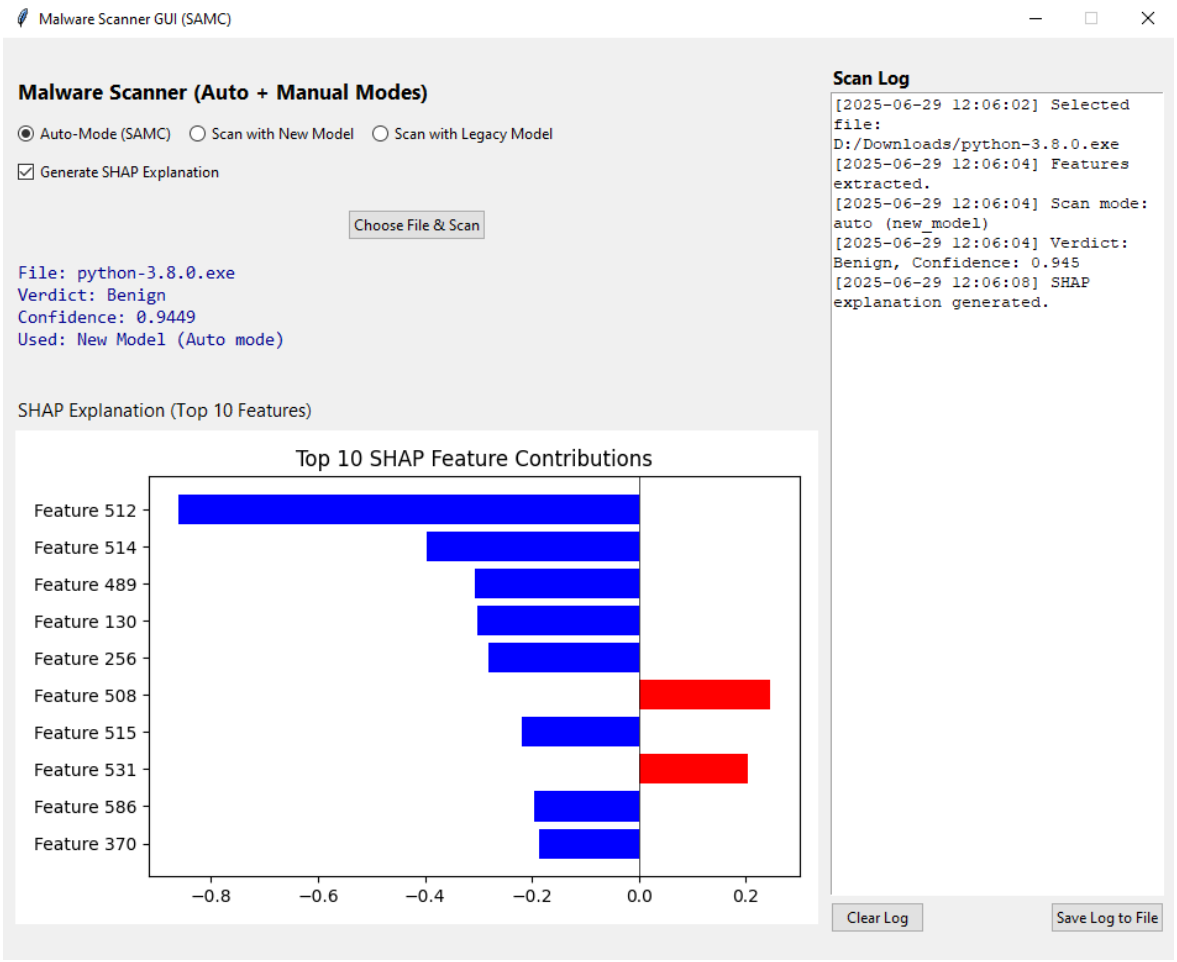
Първо се проучват доказателства, свързани с латентността и поведението при рутиране на решения, докато изпълняваме 100 изпълними примера. Всеки файл е обхванат от край до край, включително извличане на характеристики, рутиране на тези характеристики през SAMC и директно регистриране на тези прогнози като част от единния процес. Не само ни интересуваше времето за изпълнение, но и искахме да разберем колко пъти сме използвали резервен метод и дали доверието в модела означава смислено надежден модел в контекста на реалния свят.

### 3.4.1. Дизайн и функции на графичния потребителски интерфейс

Дзайнът на потребителския интерфейс е разделен на 3 основни панела:

- Левият панел ще позволява на потребителя да избере режима за сканиране, да включва/изключва SHAP обясненията, да избере файл за сканиране и да види резултата.
- Десният панел ще позволява да се види лог на сканирането в реално време с времеви запис на съобщенията.
- Разделът с обяснения на SHAP представя хоризонтална стълбовидна диаграма, изобразяваща 10-те най-влиятелни характеристики, допринасящи за прогнозите, оцветени по знак (червено = положителен клас, синьо = отрицателен клас).

Експериментален резултат от проведено сканиране на безвреден файл със SHAP обяснение е представен на Фиг. 3.12:




*Фиг. 3.12. Примерен GUI интерфейс със SHAP обяснение за безвреден файл, сканиран чрез автоматичен режим, използвайки новия модел.*

От Фиг. 3.12 може да се види кои са 10-те най-важни характеристики, допринасящи за класификацията. Червените ленти показват характеристики, които насочват прогнозата към злонамерен клас; сините ленти показват характеристики, допринасящи за доброкачествена заключение.

### 3.4.2. Време за изпълнение и латентност на рутирането

За да се определи възприеманата производителност в реално време на рамката SAMC, е направен бенчмарк на латентността на всички .exe примери, разположени в тестовата директория. Измерено е времето за изпълнение от точката, в която е започнато извличането на характеристики, до момента, в който е достигната прогнозата, идентифицирана от терминала, използвайки логиката за рутиране на SAMC (legacy model, new model или fallback).

Всяка проба преминава през:

- Извличане на вектор от характеристики (616 характеристики)
- Рутиране въз основа на достоверност чрез SAMC
- Извеждане на модела
- Регстриране, класифициране

Сканирането на тези 100 файла може да се обобщи, както е показано в Таблица 3.6:

*Таблица 3.6. Метрики и стойнсти от сканирането на 100 файла.*

| Метрики | Стойност (sec.) |
|---|---|
| **Fastest** | 0.0569 |
| **Slowest** | 0.5129 |
| **Average** | 0.2334 |

Тези доказателства показват, че дори използването на логиката, базирана на доверието в прогнозата, е позволило класификация в почти реално време. Резервната логика е била използвана понякога за случаи, за които е установено, че няма несигурност относно резултата от прогнозата (напр. под прага), тъй като това е самоосъзнат дизайн, който отчита неяснотата/неизвестността или входните данни или моделите, които са предоставили оценки за доверие, оценени като ниски.

Съществува възможност в бъдеще да бъдат намерени подобрения, които биха могли да съкратят времето за оптимизация на предварителната обработка и традиционното зареждане на характеристики.

### 3.4.3. Сравнение на производителността: Рутиране срещу нерутиране

За да се оцени практическата ефективност на механизмите за рутиране, базирани на доверие, и резервните механизми, проведохме експеримент, в който оценихме четири различни стратегии за извод:

1. **Само Legacy Model** – използва VotingClassifier, обучен на EMBER 2018 и разчита единствено на исторически модели.

2. **Само New Model –** използва настроения и оптимизиран от Optuna класификатор XGBoost, обучен на EMBER 2024, който използва по-новото разпределение на данните.

3. **Комбинирано гласуване (без рутиране)** – просто наивен ансамбъл от двата модела с меко гласуване без никакви прагове за доверие.

4. **SAMC рутиране + резервен механизъм** – предложения интелигентен модел, който динамично избира модел въз основа на достоверността на прогнозата и който използва претеглена резервна логика, ако никой модел не премине прага.

За този експеримент бе създаден балансиран тестов набор от 50 известни проби от зловреден софтуер и 50 доброкачествени изпълними файла, като всеки файл премина през всичките четири канала за извод, след като беше статично анализиран в 616-измерен PE вектор на характеристиките. Резултатите са обобщени в Таблица 3.7:

*Таблица 3.7. Сравнение на ефективността на различни стратегии за извод върху балансиран тестов набор (50 злонамерени и 50 доброкачествени файла).*

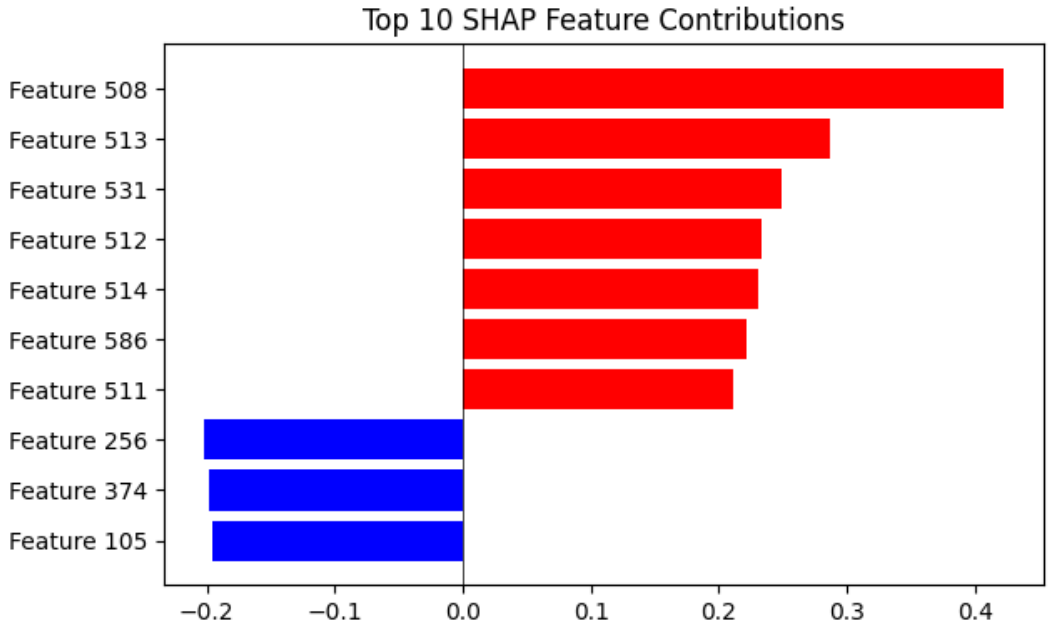
| Mode | Accuracy | F1-Score | ROC-AUC | False Positives | False Negatives |
|---|---|---|---|---|---|
| **Legacy Only (EMBER 2018)** | 0.6740 | 0.5000 | 0.9425 | 1 | 30 |
| **New Model Only (EMBER 2024)** | 0.8260 | 0.8421 | 0.8820 | 9 | 5 |
| **Combined Voting (No Routing)** | 0.8680 | 0.8608 | 0.9683 | 3 | 7 |
| SAMC Routing + Fallback | 0.9140 | 0.9036 | 0.9981 | 1 | 6 |

Тези резултати показват, че стратегията за рутиране SAMC е превъзхождала всички останали режими по всички показатели за оценка, постигайки F1-оценка над 0.9 и перфектна прецизност, но също така много висока попълнота и ROC-AUC. За разлика от традиционния модел, който показваше прекалено консервативно поведение при отчитане (без фалшиви положителни резултати, но с много ниска отзивчивост), SAMC постига среден вариант, тъй като или селективно се доверява на по-уверения модел, или прилага претеглен резервен метод. Това подкрепя заключението, че стратегията за избор на модел, съобразен с доверието, не само подобрява устойчивостта, но и помага за намаляване на фалшивите положителни и фалшиви отрицателни резултати при сравняване на производителността.

При изследване на ефективността на класификацията спрямо наивно ансамблово гласуване, подходът на SAMC беше по-добър, тъй като избягваше сляпото осредняване и вместо това се адаптираше, за да използва по-достоверния модел селективно за всяка извадка.

### 3.4.4. SHAP обяснение за зловреден файл

За да дадем още един пример за обяснимост и доверие, подготвихме локално SHAP обяснение за един от резултатите от злокачествени проби от новия модел. Фиг. 3.13 показва 10-те най-важни характеристики, които са допринесли за заключението „злонамерен“.




*Фиг. 3.13. SHAP обяснение за злонамерен файл. Червените ленти показват положителен принос към злонамереното прогнозиране, докато сините ленти показват доброкачествени характеристики.*

SHAP визуално показва, че много от сегментите на ентропията и байтовата хистограма имат доминиращо значение за решението, което означава, че моделът силно обръща внимание на аномалиите в статистическите характеристики. Някои от същите характеристики също показват голямо отклонение в KS теста (напр. Характеристика 507), което подкрепя ротацията на адаптивното и обяснимото откриване.

Като цяло, нашите открития показват, че SAMC предоставя силни агрегирани показатели, но е значително подобрен в калибрирането на доверието, резервните методи и локалната интерпретируемост. Логиката на рутиране увеличава мощността на ансамбъла с решения, базирани на доверие, и предоставя гъвкав механизъм, полезен както за минали, така и за съвременни характеристики на зловредния софтуер.

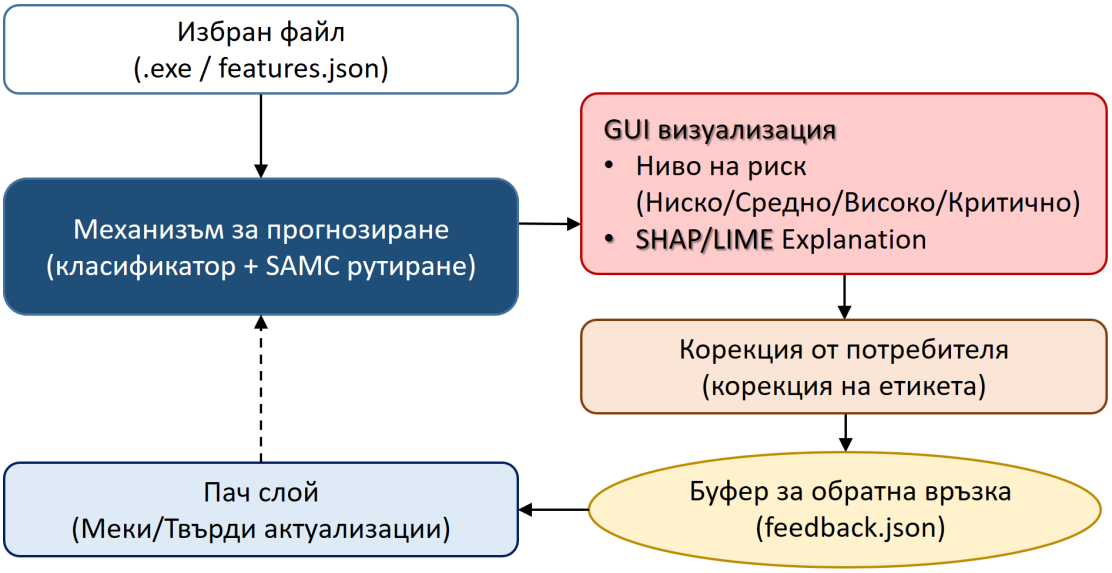
## 3.5. Резултати от разработеното и тествано приложение Shipka Guard на база на предложената адаптивна рамка, разчитаща на доверие за класификация на зловреден софтуер с корекции за обратна връзка

Графичният потребителски интерфейс (GUI) на разработеното приложение под името „Shipka Guard“ е изграден с помощта на рамката Streamlit. Това решение позволява бързо внедряване и предоставя достъп на анализаторите без инсталация или сложни конфигурации. В рамките на графичния потребителски интерфейс потребителите могат да качват PE файлове, да правят прогнози в реално време със свързани стойности на достоверност и да получават обясними (XAI) визуализации. Включени са както SHAP, така и LIME обяснения, с предоставен допълващ изглед; SHAP показва глобални приноси на характеристиките въз основа на архитектурата на модела, докато LIME показва локално интерпретируеми приближения въз основа на локалното решение.

Освен това преложението има опция, която може да се използва за ръчен избор на адаптивни версии в падащо меню, за да се сравняват и противопоставят паралелни адаптивни модели. Вграденият импорт/експорт на обратна връзка също позволява запазване или споделяне на обратната връзка от анализаторите, което допринася за възпроизводимостта на всеки експеримент и постига съвместен работен процес. Комбинацията от подробни регистрационни файлове и регулируема опция за прагове, с възможностите, прави Shipka Guard интерактивна изследователска и оперативна платформа.

В разработената версия на приложението „Shipka Guard“ са интегрирани два XAI метода: (1) LIME локално обяснява всички модели и (2) SHAP глобално обяснява приноса на вградените характеристики в моделите, базирани на дървета (Adaptive XGBoost). Вградените визуализации се представят в графичния потребителски интерфейс, за да помогнат за разбирането и валидирането на решенията. Тези решения

са интегрирана в предложената и реализирана архитектура на системата за машинно обучение с обратна връзка за класификация и обяснимост, паказана на Фиг. 3.14.




*Фиг. 3.14. Архитектура на системата за машинно обучение с обратна връзка за класификация и обяснимост (XAI).*

Всички системни събития (сканирания, записи за обратна връзка, преобучение) се регистрират в logs/scan.log. Освен това, рамката проследява времето за изпълнение (в ms) за всяка прогноза, за да гарантира, че системата ще работи в реално време на потребителския хардуер. Тази функция предоставя перспектива за осъществимостта и внедряването на проекта в корпоративни среди, включително резултати от сканиране, записи за обратна връзка, актуализации на корекции и събития за преобучение.

### 3.5.1. Оценка на разработеното приложение Shipka Guard, на база на предложената рамка

Оценката на приложимостта на разработеното приложение Shipka Guard, на база на предложената рамка, използва два публично достъпни набора от данни за бенчмаркове: EMBER 2018, съдържащ над 1 милион PE проби, характеризиращи се с 616 статични характеристики (т.е. хистограми, ентропия, низови статистики) и използван като базов „наследен“ модел за изследването; и EMBER 2024 с приблизително 960 000 проби. EMBER 2024 съдържа съвременни семейства злонамерен софтуер и е хармонизиран в същото 616-измерно пространство от характеристики за сравнение с набора от данни EMBER 2018. Освен това се използва

трети, генериран от потребителя набор от данни. Рамката е структурирана така, че да интегрира генерирани от потребителя данни по лесен начин, за да се хармонизира в същото 616-измерно пространство от характеристики. Следователно е важно Shipka Guard да остане гъвкав за бъдещи източници на данни и почти реални приложения извън публичните бенчмаркове. Хармонизирането позволява на модели, обучени върху различни поколения набори от данни, все още да работят с рамката SAMC.

Експериментите бяха проведени на потребителска машина със следната конфигурация: CPU: 16-ядрен Intel i7 (10-то поколение); Памет: 32 GB RAM; GPU: NVIDIA RTX 2060 (6 GB VRAM); Софтуер: Python 3.8 (старият екстрактор е само за Python 3.8, но всички други софтуерни подходи могат да използват по-късни версии, експериментите се извършват последователно под 3.8.), Streamlit, scikit-learn, XGBoost, LIME, SHAP; Среда: Всички модели бяха обучени офлайн; Операциите с инференция и графичен потребителски интерфейс бяха изпълнени в реално време. Тази конфигурация отразява типичния високопроизводителен хардуер от потребителски клас, оценяването показва, че тя по подходящ начин отразява достоверни сценарии за внедряване.

За целите на честност и възпроизводимост на експериментите, от EMBER2018 и EMBER2024 беше създаден балансиран набор от 100 000 проби (50 000 доброкачествени, 50 000 злонамерени). Първоначално базовите модели и методът за автоматично рутиране SAMC бяха сравнени при балансираната оценка, както е показано в Таблица 3.8.

*Таблица 3.8. Производителност на Legacy, Modern и SAMC Auto (T=0.85).*

| Режим | Accuracy | Precision | Recall | F1 | ROC-AUC | FN | FP |
|---|---|---|---|---|---|---|---|
| Force Legacy | 0.92409 | 0.9540 | 0.8906 | 0.9213 | 0.9621 | 5472 | 2148 |
| Force Modern | 0.96641 | 0.9793 | 0.9529 | 0.9660 | 0.9955 | 2353 | 1006 |
| Auto (SAMC) [T=0.85] | 0.95917 | 0.9852 | 0.9324 | 0.9580 | 0.9942 | 3381 | 702 |

Моделът Legacy се представя слабо, като по-ниското качество на данните за обучение определя по-висока FN (попълнота). Моделът Modern постига висока точност и попълнота с повече фалшиви положителни резултати. SAMC Auto е някъде по средата: по-малко FP (702) от Modern (1006), за сметка на повече FN. Това дава добра

илюстрация на полезността на динамичното рутиране като метод за балансиране на риска.

Ефектът от леката адаптация на Patch спрямо пълното преобучение е показан в Таблица 3.9.

*Таблица 3.9. Аблационно проучване, сравняващо изходните данни Modern, Patch с 500 проби за обратна връзка и пълно преобучение с 10 000 проби за обратна връзка.*

| Режим | Accuracy | Precision | Recall | F1 | ROC-AUC | FN | FP |
|---|---|---|---|---|---|---|---|
| Base (Modern) | 0.96641 | 0.9793 | 0.9529 | 0.9660 | 0.9955 | 2353 | 1006 |
| Patch (500 fb) | 0.96727 | 0.9799 | 0.9541 | 0.9668 | 0.9918 | 2294 | 979 |
| Full Retrain (10k) | 0.94719 | 0.9672 | 0.9257 | 0.9460 | 0.9906 | 3713 | 1568 |

Слоят Patch, обучен с 500 проби за обратна връзка, показва забележими подобрения: както FN (-59), така и FP (-27) намаляват, а резултатът F1 леко се увеличава. По този начин, дори малък и евентуално небалансиран буфер за обратна връзка може да се използва за прилагане на последователни, бързи корекции. За разлика от това, пълното преобучение с 10 000 проби за обратна връзка дава по-лош резултат. Въпреки че изглежда нелогично, обяснението се корени в пристрастия и дисбаланс в пула за обратна връзка: ние умножаваме проблемните проби и увеличаваме грешките, вместо да ги поправяме. Това наистина предоставя критичен урок: качеството и разнообразието са също толкова важни, колкото количеството на обратната връзка.

Двете се допълват – Patch е изключително полезен в случаи с малко или шумни опашки за обратна връзка, осигурявайки бърза и лека адаптация, а пълното преобучение трябва да се извършва само когато има достатъчно примери, които демонстрират баланс и покритие на променливостта между класовете. Вместо да се счита за ограничение, това също така демонстрира важността на канала за обратна връзка, основан на доверието, за да не доведе до наивно преобучение, което да дестабилизира системата.

Изпълняваме SAMC Auto, използвайки различни прагове на доверие, както е показано в Таблица 3.10.

*Таблица 3.10. Метрики при различните прагове на доверие.*

| T | Precision | Recall | %Uncertain |
|---|---|---|---|
| 0.80 | 0.9854 | 0.9310 | 6.5% |
| 0.85 | 0.9852 | 0.9324 | 8.2% |
| 0.90 | 0.9851 | 0.9341 | 10.6% |

По-ниските прагове (T = 0.80) водят до по-малко въздържания, но по-голям риск от погрешни класификации. Използването на по-високи прагове (T = 0.90) увеличава броя на въздържанията, което също подобрява надеждността, но е съпроводено с намаляване на пропускателната способност. Това допълнително подкрепя идеята, че T е регулируем бутон за безопасност.

Времето за изпълнение на файл е измерено директно от системните лог файлове за всеки модел и резултатите са дадени в Таблица 3.11.

*Таблица 3.11. Латентност на изпълнението, нормализирана до 100 проби на модел*

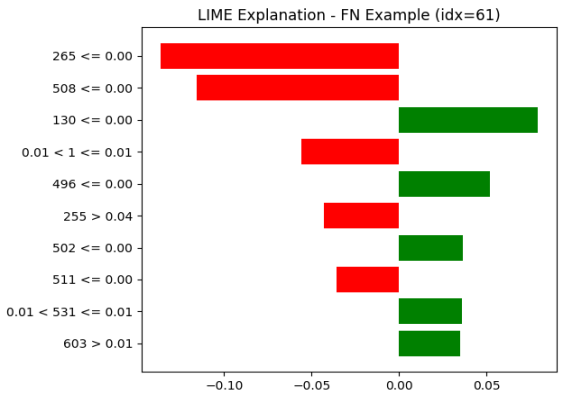
| Режим | n_samples | Mean (ms) | Std (ms) | p50 | p90 | p99 |
|---|---|---|---|---|---|---|
| Modern | 100 | 47.6 | 24.5 | 33.1 | 83.6 | 91.3 |
| Legacy | 100 | 131.4 | 0.0 | 131.4 | 131.4 | 131.4 |
| Adaptive | 100 | 109.7 | 29.5 | 109.7 | 133.3 | 138.6 |

Моделът Modern постига средно време за извод под 50 ms, което е доста под целевата граница от 90 ms/проба. Legacy и Adaptive са по-бавни (~110–130 ms), но все още в рамките на реалното време. Тези резултати потвърждават възможността за внедряване на системата дори с активиран SHAP/LIME.

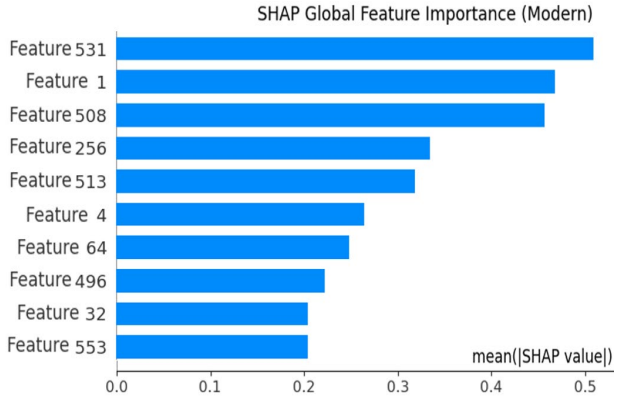
Три примерни казуса, илюстриращи динамиката на обратната връзка и обясненията:

- **Фалшиво отрицателен резултат, коригиран чрез Patch.** На Фиг. 3.15 (LIME) е показаня злонамерена проба, погрешно класифицирана като доброкачествена. След корекция с 500 семпли за обратна връзка, решението беше коригирано, намалявайки FN.
- **Фалшиво положителен резултат, коригиран чрез обратна връзка.** Доброкачествена проба, погрешно класифицирана като злонамерена, беше добавена към буфера за обратна връзка. Моделът Patch беше коригиран и фалшиво положителният резултат беше елиминиран.

- **Несигурен случай, обяснен чрез LIME и SHAP.** При T=0.90, SAMC се въздържа. LIME показа противоречиви локални доказателства, докато глобалното значение на SHAP (Фиг. 3.16) обясни влиянията на доминиращите характеристики.

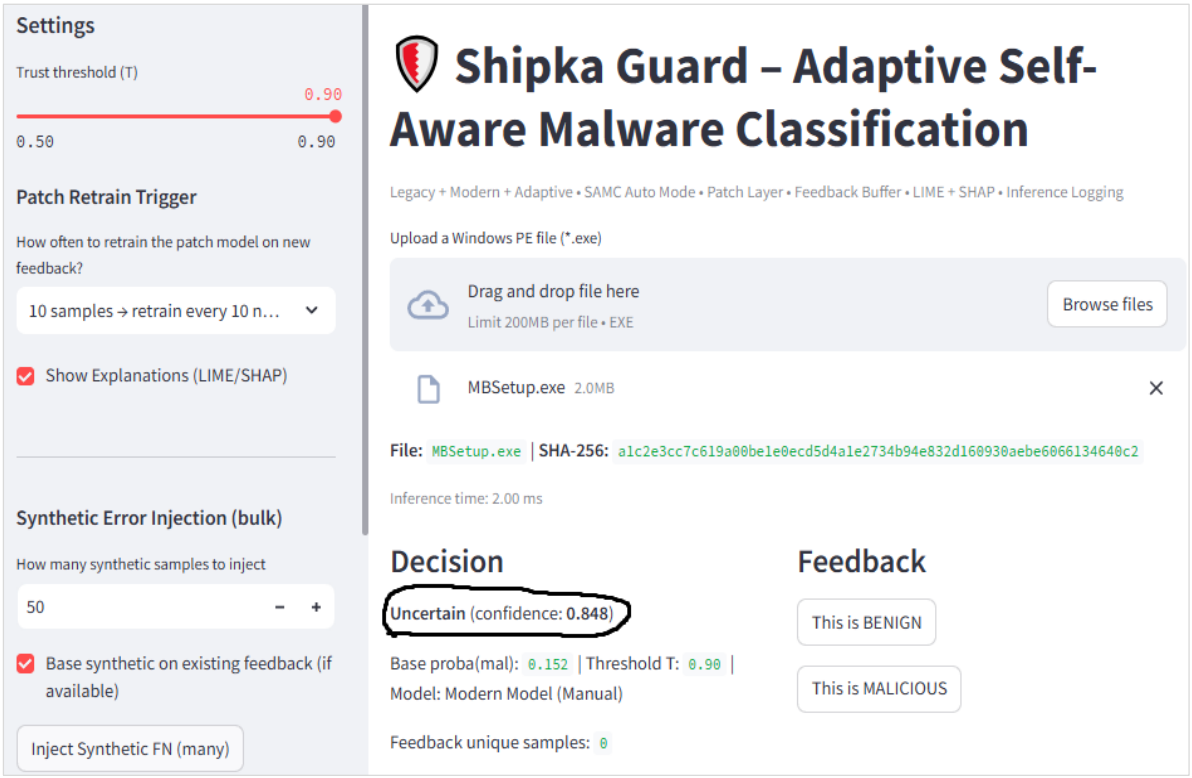



*Фиг. 3.15. Обяснението на LIME за FN е коригирано от Patch.*




*Фиг. 3.16. Важност на глобалните характеристики на SHAP за Modern модела.*

GUI, показан на Фиг. 3.17, показа въздържанието на анализатора, който може да предостави коригираща обратна връзка.




*Фиг. 3.17. Екранна снимка на потребителския интерфейс с „Несигурно решение“ (T=0.90).*

Оценката потвърждава ефективността и ограниченията на предложената рамка. Старите модели са остарели, докато съвременните модели се представят добре, но произвеждат повече FP. SAMC Auto балансира рисковете чрез динамично рутиране. Слоят Patch осигурява лесно и леко почистване на грешки с малки обработени пулове, а пълното преобучение не дава превантивно предимство, освен ако няма подходящо количество обратна връзка. Прагът на доверие предлага настройваем баланс между попълнота и въздържания. Латентността остава в рамките на реалното време, докато казусите показват как модулът за обяснение (LIME/SHAP) и несигурните въздържания впоследствие са увеличили доверието на анализаторите или поне са валидирали резултатите. Следователно, ние обосноваваме процеса SAMC като ефективен и надежден процес за адаптиране на класификацията на зловредния софтуер.

### 3.5.2. Дискусии и изводи

Оценката показва, че Shipka Guard е в състояние да включи наследени, модерни и адаптивни модели в интегрирана SAMC рамка и предоставя обратна връзка, управлявана от потребителя, за да продължи адаптацията. Добавянето на преобучение на корекции, изводи за обратна връзка и регистриране на времето за извод прави системата по-устойчива и лесна за внедряване на практика. Shipka Guard има ясен слой на доверие: ако нито един модел не премине прага на доверие, зададен от потребителя, системата избира резервен ансамбъл или връща флаг „Несигурно“ за човешка проверка. Това осигурява безопасно въздържание: проби с отрицателно или несигурно доверие се изпращат за проверка, вместо да се налага автоматизирано решение. По този начин слоят на доверие намалява грешките с големи последици и позволява да се даде приоритет на човешкия анализ на случаи с ниско доверие.

Shipka Guard интегрира 4 режима на сканиране, инжектиране на синтетични грешки, експорт/импорт на обратна връзка, базиран на JSON, и мащабируем анализ. Това прави рамката Shipka Guard не само по-адаптивна, но и по-подходяща за реални условия на внедряване.

Рамката Shipka Guard е базирана на интерактивен графичен потребителски интерфейс, който намалява бариерата за неспециализирани потребители да експериментират с класификацията на зловреден софтуер. Буферът за обратна връзка позволява на потребителя съвместно да коригира модела, което му позволява да

експортира/импортира файлове с обратна връзка. Интеграцията на обяснимостта допринася за доверието в решенията, което се е увеличило чрез повече информация за функциите от локална и глобална гледна точка. Регистрирането на мащабируемостта отразява адаптивността на системата, което не възпрепятства работните процеси в реално време. Всички тези аспекти показват, че адаптивното откриване на зловреден софтуер може да бъде практично и прозрачно.

Въпреки това, все още оставаме с ограничения, свързани със статичния анализ. Системата използва само характеристики от PE файлове. При разширено пакетиране и злонамерен софтуер, работещ само по време на изпълнение, системата ще се затруднява. Синтетично инжектиране срещу реални враждебни проби – инжектирането на грешки е чудесно за тестване на адаптацията, но ще трябва да се докаже устойчивостта спрямо враждебно размити (или обфускирани) PE файлове. Адаптацията за обратна връзка за обхвата на отклонението на характеристиките отчита само инкременталното отклонение, а по-големите разпределителни промени все още биха изисквали преобучение върху нови широкомащабни набори от данни. Мащабируемостта е отвъд потребителския хардуер, докато резултатът е бърз с една машина, внедряването на корпоративно ниво със значителна пропускателна способност може да изисква разпределена или облачна имплементация (напр. Kubernetes технология). Горните последици и ограничения осигуряват както практическата жизнеспособност, така и текущите ограничения за Shipka Guard и подготвят почвата за бъдеща работа.

Рамката предлага няколко нови приноса. Първият от тях е механизъм за рутиране, съобразен с доверието, който използва множество поколения модели, включително резервен механизъм. Следващият е буфер за обратна връзка и слой за корекции за лека адаптация от потребителски корекции и синтетични инжекции. Реализацията му се осъществява чрез интерактивен графичен потребителски интерфейс Streamlit, който предоставя четири режима на сканиране, споделяне на обратна връзка на базата на JSON и локални/глобални обяснения (LIME+SHAP). Той също така осигурява мащабируемо регистриране, което представя изводи в реално време (< 90 ms) на потребителски хардуер.

Комбинацията от адаптивност, интерпретируемост и мащабируемост води до справяне с предизвикателствата на непрекъснатото разработване на практични

системи за откриване на зловреден софтуер от следващо поколение, които постоянно остават „верни“ за дадените модели на заплахи, вместо да се променят постоянно. Включването на избор на версия и интерфейс за обратна връзка като част от приложението позволява възпроизводимост и съвместна използваемост, което прави Shipka Guard не само практичен инструмент за защита от зловреден софтуер, но и инструмент за съвместни изследвания или оценки.

# ЗАКЛЮЧЕНИЕ – РЕЗЮМЕ НА ПОЛУЧЕНИТЕ РЕЗУЛТАТИ

Зловреднният софтуер (malware) може да приеме много и различни форми, като вируси, червеи, троянски коне и ransomware, и може да причини значителни вреди на хора, организации и дори на цели държави. Зловреднният софтуер остава една от най-сериозните заплахи за информационната сигурност. С еволюцията на дигиталните технологии се променят и методите за извършване на атаки, което налага разработване на нови алгоритми за ефективно и навременно откриване на зловредно поведение. Традиционните решения, базирани предимно на сигнатури, вече не са достатъчни за справяне със съвременните заплахи.

Отчитайки необходимостта от разработването на ефективни средства за противодействие на зловредния софтуер, определи и основната цел на настоящия дисертационен труд, свързана с Изследване и анализ на възможностите за откриване на злонамерен софтуер чрез средствата на машинно обучение. За де се реализират експериментите безопасно в контролирана среда е предложен математически модел за избор на подходяща виртуална машина. При наличието на безопасна в контролирана среда са проведени експерименти за установяване ефективността на различни алгоритми на машинното обучение и е направен анализ на тяхното представяне за целите на откриване на зловреден софтуер. На база на установените констатации е предложен подобрен подход за статичен анализ за откриване на зловреден софтуер чрез оптимизиране извличането на характеристики и комбинирайки различни алгоритми за машинно обучение. В допълнение на тези усилия е предложена рамка за статична класификация на зловреден софтуер, използваща оптимизация на функции и ансамблово обучение. Резултатите от проведеното тестване показват, че анализът на фалшиво положителните резултати за ансамбъла е значително по-нисък от този на отделните модели. За подобряване на класификация на зловреден софтуер е предложена самоосъзната рамка, интегрираща рутиране на модели на базата на система за доверие за избора и обяснимост на характеристиките. Логиката на рутиране увеличава мощността на ансамбъла с решения, базирани на доверие, и предоставя гъвкав механизъм, полезен както за

минали, така и за съвременни характеристики на зловредния софтуер. За прецизиране на класификация на зловредния софтуер е предложена адаптивна рамка, съобразена с доверието, позволяваща класификация на зловреден софтуер с възможност за корекции чрез обратна връзка. Буферът за обратна връзка позволява на потребителя съвместно да коригира модела, което му позволява да експортира/импортира файлове с обратна връзка. Интеграцията на обяснимостта допринася за доверието в решенията, което се е увеличило чрез повече информация за функциите от локална и глобална гледна точка. Тази адаптивна рамка е реализирана в разработена демо версия на софтуерно приложение под името „Shipka Guard“. Тя интегрира механизъм за рутиране, съобразен с доверието, който използва множество поколения модели, включително резервен механизъм. В нея се използва буфер за обратна връзка и слой за корекции за лека адаптация от потребителски корекции и синтетични инжекции. Благодарение на комбинацията от адаптивност, интерпретируемост и мащабируемост се постига решават предизвикателства, свързани с непрекъснатото разработване на практични системи за откриване на зловреден софтуер от следващо поколение, които постоянно остават „верни“ за дадените модели на заплахи, вместо да се променят постоянно.

Чрез проведените многобройни експерименти с различни видове данни е установена практическата приложимост както на предложените хибридни алгоритми така и на разработеното приложение под името „Shipka Guard“.

Като бъдещо развитие на изследванията в дисертационния труд се планира изследвания, свързани с едновременно и многокласово идентифициране на зловреден софтуер.

Получените резултати по темата на дисертационното изследване са докладвани на 3 международни конференции. Представените резултати са отразени в общо 4 научни публикации в издания, които са реферирани и индексирани в световноизвестни бази данни с научна информация – Scopus.

# ПРИНОСИ

Получените резултати, описани в настоящия дисертационен труд, могат да се обобщят в следните научни и научно-приложни приноси:

1. Предложени са два матаматически модела, чрез които може да се направи избор на софтуер за подходяща виртуална машина за целите на експерименталното тестване за откриване на зловреден софтуер.
2. Предложен е подобрен подход за статичен анализ за откриване на зловреден софтуер чрез оптимизиране извличането на характеристики чрез комбиниране на различни алгоритми за машинно обучение. Проведените тестове с предложените хибридни алгоритми показват по-добра производителност.
3. Предложена е рамка за статична класификация на зловреден софтуер, която използва оптимизация на функции и ансамблово обучение. Резултатите показват, че анализът на фалшиво положителните резултати за ансамбъла е значително по-нисък от този на отделните модели.
4. Предложена е самоосъзната класификация на зловреден софтуер чрез рутиране на модели на базата на система за доверие за избора и обяснимост на характеристиките. Логиката на рутиране увеличава мощността на ансамбъла с решения, базирани на доверие, и предоставя гъвкав механизъм, полезен както за минали, така и за съвременни характеристики на зловредния софтуер.
5. Предложена е адаптивна рамка, съобразена с доверието, за класификация на зловреден софтуер с корекции за обратна връзка. Тази рамка е едновременно адаптивна и устойчива, тъй като включва самоосъзнат класификатор на модели, който използва адаптивна логика за автоматичен избор между традиционни и съвременни слоеве на моделите чрез измерване на надеждността на прогнозирането. Интеграцията на обяснимостта допринася за доверието в решенията, което се е увеличило чрез повече информация за функциите от локална и глобална гледна точка.

# СПИСЪК НА ПУБЛИКАЦИИТЕ ПО ДИСЕРТАЦИОННИЯ ТРУД

1. **Barzev**, I., Borissova, D.: An improved static analysis approach for malware detection by optimizing feature extraction combining different ML algorithms. In: Bennour, A., Bouridane, A., Almaadeed, S., Bouaziz, B., Edirisinghe, E. (eds) Intelligent Systems and Pattern Recognition. ISPR 2024. Communications in Computer and Information Science, vol. 2304, pp. 102–115, 2025, Springer, Cham. https://doi.org/10.1007/978-3-031-82153-0_8. Print ISBN 978-3-031-82152-3, **SJR Q4=0.182**
2. **Barzev**, I., Borissova, D.: Performance analysis of LSTM, SVM, CNN and CNN-LSTM algorithms for malware detection in IoT dataset. WSEAS TRANSACTIONS on COMPUTER RESEARCH, vol. 13, 288-296, 2025, http://dx.doi.org/10.37394/232018.2025.13.27. E-ISSN: 2415-1521, **SJR Q4=0.140**
3. **Barzev, I.**, Borissova, D., Buhtiyarov, N.: Comparison of different binary classification algorithms for malware detection. In: Rocha, Á., Ferrás, C., Hochstetter Diez, J., Diéguez Rebolledo, M. (eds) Information Technology and Systems. ICITS 2024. Lecture Notes in Networks and Systems, vol. 932, pp. 369–378, 2024, Springer, Cham. https://doi.org/10.1007/978-3-031-54235-0_33. Print ISBN 978-3-031-54234-3, Online ISBN 978-3-031-54235-0, **SJR Q4=0.17.**
4. Borissova, D., **Barzev, I.**, Yoshinov, R., Kotseva, M.: Group decision-making models for selection of virtual machine software for malware detection purposes. In: Proc. of 12th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 2023, https://doi.org/10.1109/MECO58584.2023.10155084. ISBN 979-8-3503-2291-0. **Scopus**

Публикации, приети за петат:

- Barzev, I., Borissova, D.: Trust-Aware Adaptive Framework for Malware Classification with Feedback Patching. 9th International Conference on Information Technology & Systems, 16-18 February 2026
- Barzev, I., Borissova, D.: A scalable Python Framework for static malware classification using feature optimization and ensemble Learning. International Conference on Advanced Research in Technologies, Information, Innovation, and Sustainability (ARTIIS 2025). 21-23 October 2025.
- Barzev, I., Borissova, D.: Self-aware malware classification via confidence-guided model routing and explainable feature attribution. 5th International Conference on Intelligent Systems and Pattern Recognition, 25-27 September 2025

# СПИСЪК НА ЗАБЕЛЯЗАНИТЕ ЦИТИРАНИЯ НА ПУБЛИКАЦИИТЕ

- Borissova, D., Barzev, I., Yoshinov, R., Kotseva, M.: Group decision-making models for selection of virtual machine software for malware detection purposes. In: Proc. of 12th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 2023. https://doi.org/10.1109/MECO58584.2023.10155084

  1. Guliashki, V., Marinova G.: Impact of cyberattacks on human's health, In: 2024 IEEE International Workshop on Technologies for Defense and Security (TechDefense), Naples, Italy, 2024, pp. 31-35, https://doi.org/10.1109/TechDefense63521.2024.10863627.
  2. Ramezanip A.: Fusion Models for Cyber Threat Defense: Integrating Clustering with Kmeans, Random Forests, and SVM against Windows Malware. In: 2024 IEEE World AI IoT Congress (AIIoT), Seattle, WA, USA, 2024, pp. 465-470, https://doi.org/10.1109/AIIoT61789.2024.10578947
  3. Gospodinov, M.: The Role of The Concept of Trust in Trust Services in Cyberspace to Enhance Cyber Security. Problems of Engineering Cybernetics and Robotics, Vol. 81, pp. 23-28, 2024, https://doi.org/10.7546/PECR.81.24.03.
  4. Liu, S., Chen, X.: Mitigating data exfiltration ransomware through advanced decoy file strategies. 2023, https://doi.org/10.21203/rs.3.rs-3750416/v1.
  5. Tomov, P., Mateeva, G., Parvanov, D.: Entropy test degradation after random numbers scaling. Problems of Engineering Cybernetics and Robotics, Vol. 80, pp. 3-12, 2023, https://doi.org/10.7546/PECR.80.23.01.

- Barzev, I., Borissova, D.: Performance analysis of LSTM, SVM, CNN and CNN-LSTM algorithms for malware detection in IoT dataset. WSEAS TRANSACTIONS on COMPUTER RESEARCH, Print ISSN: 1991-8755, E-ISSN: 2415-1521, vol. 13, 288-296, 2025, http://dx.doi.org/10.37394/232018.2025.13.27.

  6. Muça, F., Ahmad, W.: Explainable AI for Malware Classification: Bridging the Gap Between Accuracy and Transparency. In: Dhoska, K., Spaho, E. (eds) AI and Digital Transformation: Opportunities, Challenges, and Emerging Threats in Technology, Business, and Security. ICITTBT 2025. Communications in Computer and Information Science, vol 2669, pp 79–99, 2026, Springer, Cham. https://doi.org/10.1007/978-3-032-07373-0_6.

- Barzev, I., Borissova, D., Buhtiyarov, N.: Comparison of different binary classification algorithms for malware detection. In: Rocha, Á., Ferrás, C., Hochstetter Diez, J., Diéguez Rebolledo, M. (eds) Information Technology and Systems. ICITS 2024. Lecture Notes in Networks and Systems, vol. 932, pp. 369–378, 2024, https://doi.org/10.1007/978-3-031-54235-0_33.

  7. Abd El-Latif, A.A., ElAffendi, M.A., AlShara, M.A., Maleh, Y. (Eds.).: Cybersecurity, Cybercrimes, and Smart Emerging Technologies: Proceedings of the Second International Conference on Cybersecurity, Cybercrimes, and Smart Emerging Technologies (CCSET 2023), Riyadh, Saudi Arabia, 5-7 December, 2023 (1st ed.). CRC Press, 2025, https://doi.org/10.1201/9781003614197.
  8. Guliashki, V., Marinova G.: Impact of cyberattacks on human's health. In: 2024 IEEE International Workshop on Technologies for Defense and Security (TechDefense), Naples, Italy, 2024, pp. 31-35, https://doi.org/10.1109/TechDefense63521.2024.10863627.

# ДЕКЛАРАЦИЯ ЗА ОРИГИНАЛНОСТ НА РЕЗУЛТАТИТЕ

Декларирам, че настоящата дисертация съдържа оригинални резултати, получени при проведени от мен научни изследвания с подкрепата и съдействието на научния ми ръководител. Резултатите, които са получени, описани и/или публикувани от други учени, са надлежно и подробно цитирани в библиографията.

Настоящата дисертация не е прилагана за придобиване на научна степен в друго висше училище, университет или научен институт.

Подпис: .....

/Илиян Магдаленов Барзев/

# БИБЛИОГРАФИЯ


1. Aboaoja, F.A., A. Zainal, F.A. Ghaleb, B.A.S. Al-rimy, T.A.E. Eisa, A.A.H. Elnour: Malware detection issues, challenges, and future directions: A survey. Applied Sciences, vol. 12(17), 8482, 2022, https://doi.org/10.3390/app12178482.
2. About malware and PUA, https://portal.av-atlas.org/malware, last access 2024/04/03.
3. Abusnaina, A., Anwar, A., Saad, M., Alabduljabbar, A., Jang, R., Salem, S., Mohaisen, D.: One step forward, two steps back: ML-based malware detection under concept drift. Computing, vol. 107, 207 2025, https://doi.org/10.1007/s00607-025-01543-7.
4. Agache, A., M. Brooker, A. Florescu, A. Iordache, A. Liguori, R. Neugebauer, P. Piwonka, D.-M. Popa: Firecracker: Lightweight virtualization for serverless applications. In: 17th USENIX Symposium on Networked Systems Design and Implementation, pp. 419–434, 2020.
5. Ahmed, I.T., Hammad, B.T., Jamil, N.: A comparative performance analysis of malware detection algorithms based on various texture features and classifiers. IEEE Access, vol. 12, pp. 11500-11519, 2024, https://doi.org/10.1109/ACCESS.2024.3354959.
6. Al Mamun, H., Keikhosrokiani, P.: Chapter 23 – Predicting onset (type-2) of diabetes from medical records using binary class classification. Ed: P. Keikhosrokiani, Big Data Analytics for Healthcare, pp. 301-312, 2022, https://doi.org/10.1016/B978-0-323-91907-4.00012-1.
7. Alahi, M.E.E, A. Sukkuea, F.W. Tina, A. Nag, W. Kurdthongmee, K. Suwannarat, S.C. Mukhopadhyay, Integration of IoT-enabled technologies and artificial intelligence (AI) for smart city scenario: Recent advancements and future trends, Sensors. vol. 23(11), 5206, 2023, https://doi.org/10.3390/s23115206.
8. Al-Andoli, M.N., Sh. Ch.Tan, K.S. Sim, Ch. P. Lim, and P.Y. Goh, “Parallel deep learning with a hybrid BP-PSO framework for feature extraction and malware classification”. Applied Soft Computing, vol. 131, 109756, 2022, https://doi.org/10.1016/j.asoc.2022.109756.
9. Ali, E., Batool, N., Rizwan, M., Sarwar, S.: Assessing concept drift in malware: A comprehensive review and analysis. In: 21st International Bhurban Conference on Applied Sciences and Technology (IBCAST), Murree, Pakistan, pp. 564-569, 2024, https://doi.org/10.1109/IBCAST61650.2024.10876901.
10. Almotairi, K.H., Application of internet of things in healthcare domain, Journal of Umm Al-Qura University for Engineering and Architecture, vol. 14, pp. 1-12, 2023, https://doi.org/10.1007/s43995-022-00008-8.
11. Alsmadi, T., Alqudah, N.: A survey on malware detection techniques. In: International Conference on Information Technology (ICIT), Amman, Jordan, 2021, pp. 371-376, https://doi.org/10.1109/ICIT52682.2021.9491765.
12. Alsubaei, F.S., Almazroi, A.A., Atwa, W.S., Almazroi, A.A., Ayub, N., Jhanjhi, N.Z.: Adaptive malware identification via integrated SimCLR and GRU networks. Sci Rep, vol. 15, 25309, 2025, https://doi.org/10.1038/s41598-025-08556-4.
13. Amin, M., Al-Obeidat, F., Tubaishat, A., Shah, B., Anwar, S., Ali Tanveer, T.: Cyber security and beyond: Detecting malware and concept drift in AI-based sensor data streams using statistical techniques. Computers and Electrical Engineering, vol. 108, 108702, 2023, https://doi.org/10.1016/j.compeleceng.2023.108702.

14. Anderson, H.S., Roth, P.: EMBER: An open dataset for training static PE malware machine learning models. 2018, https://doi.org/10.48550/arXiv.1804.04637.
15. Anowar, F., Sadaoui, S., Selim, B.: Conceptual and empirical comparison of dimensionality reduction algorithms (PCA, KPCA, LDA, MDS, SVD, LLE, ISOMAP, LE, ICA, t-SNE), Computer Science Review, vol. 40, 100378, 2021, https://doi.org/10.1016/j.cosrev.2021.100378.
16. Arora, P., Gupta, R., Malik, N., Kumar, A.: Malware analysis types & techniques: A survey. In Proc. of 5th International Conference on Information Management & Machine Intelligence (ICIMMI '23). Association for Computing Machinery, New York, NY, USA, 135, pp.1-6, 2024, https://doi.org/10.1145/3647444.3652439.
17. Asghar, H.J., Zhao, B.Z.H., Ikram, M., Nguyen, G., Kaafar, D., Lamont, S., Coscia, D.: Use of cryptography in malware obfuscation. Journal of Computer Virology and Hacking Techniques, 2023, https://doi.org/10.1007/s11416-023-00504-y.
18. Aslam, S., H. Herodotou; E. Garro, Á. Martínez-Romero, M.A. Burgos, A. Cassera, G. Papas, P. Dias, M.P. Michaelides, IoT for the maritime industry: Challenges and emerging applications, In: 2023 18th Conference on Computer Science and Intelligence Systems (FedCSIS), Warsaw, Poland, 2023, pp. 855-858, https://doi.org/10.15439/2023F3625.
19. Aslan, Ö., Yilmaz, A.A.: A new malware classification framework based on deep learning algorithms. IEEE Access, vol. 9, pp. 87936–87951, 2021, https://doi.org/10.1109/ACCESS.2021.3089586.
20. Aslan, Ö.A., Samet, R.: A comprehensive review on malware detection approaches, In: IEEE Access, Vol. 8, pp. 6249-6271, 2020, https://doi.org/10.1109/ACCESS.2019.2963724.
21. Augello, A., De Paola, A., Lo Re, G.: M2FD: Mobile malware federated detection under concept drift. Computers & Security, 152, 104361, 2025, https://doi.org/10.1016/j.cose.2025.104361.
22. Awad, M., Khanna, R.: Support vector machines for classification. In: Efficient Learning Machines. Apress, Berkeley, CA. (2015). https://doi.org/10.1007/978-1-4302-5990-9_3;
23. Ayeni, R.K., Adebiyi, A.A., Okesola, J.O., Igbekele, E.: Phishing attacks and detection techniques: A systematic review. In: Proc. of Int. Conf. on Science, Engineering and Business for Driving Sustainable Development Goals (SEB4SDG), 2024, pp. 1-17, https://doi.org/10.1109/SEB4SDG60871.2024.10630203.
24. Baghirov, E., A comprehensive investigation into robust malware detection with explainable AI, Cyber Security and Applications, vol. 3, 100072, 2025, https://doi.org/10.1016/j.csa.2024.100072.
25. Bakır, H.: A new method for tuning the CNN pre-trained models as a feature extractor for malware detection. Pattern Analysis and Applications, vol. 28, 26, 2025, https://doi.org/10.1007/s10044-024-01381-x.
26. Balram, N., Hsieh, G., McFall, C.: Static malware analysis using machine learning al-gorithms on APT1 dataset with string and PE header features. In: International Confer-ence on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2019, pp. 90-95, https://doi.org/10.1109/CSCI49370.2019.00022.
27. Baptista, I., S. Shiaeles, N. Kolokotronis, A novel malware detection system based on machine learning and binary visualization, 2019 IEEE International Conference on Communications Workshops (ICC Workshops), Shanghai, China, 2019, pp. 1-6, https://doi.org/10.1109/ICCW.2019.8757060 .
28. Barzev, I., Borissova, D., Buhtiyarov, N.: Comparison of different binary classification algorithms for malware detection. In: Rocha, Á., Ferrás, C., Hochstetter Diez, J., Diéguez Rebolledo, M. (eds)

Information Technology and Systems. ICITS 2024. Lecture Notes in Networks and Systems, vol. 932, pp. 369-378, 2024, https://doi.org/10.1007/978-3-031-54235-0_33.

29. Barzev, I., Borissova, D.: An improved static analysis approach for malware detection by optimizing feature extraction combining different ML algorithms. In: Intelligent Systems and Pattern Recognition. ISPR 2024. Communications in Computer and In-formation Science, vol. 2304, pp. 102-115, 2025, https://doi.org/10.1007/978-3-031-82153-0_8.
30. Berikol, G.B., Berikol, G.: Chapter 7 – Predictive models in precision medicine. Ed: D. Barh, Artificial Intelligence in Precision Health, pp. 177-188, 2020, https://doi.org/10.1016/B978-0-12-817133-2.00007-0.
31. Blagoev, I.: Cyber security threats in the public hosting services. In: Tagarev, T., Stoianov, N. (eds) Digital Transformation, Cyber Security and Resilience. DIGILIENCE 2020. Communications in Computer and Information Science, vol. 1790, 2024, https://doi.org/10.1007/978-3-031-44440-1_10.
32. Blagoev, I.I, Shalamanov, V.: Development of cyber ranges as a reference environment for digital transformation. In: 2023 4th International Conference on Communications, Information, Electronic and Energy Systems (CIEES), Plovdiv, Bulgaria, 2023, pp. 1-5, https://doi.org/10.1109/CIEES58940.2023.10378806.
33. Borissova, D., Barzev, I., Yoshinov, R., Kotseva, M.: Group decision-making models for selection of virtual machine software for malware detection purposes. In: 12th Mediterranean Conference on Embedded Computing (MECO), Budva, Montenegro, 2023, https://doi.org/10.1109/MECO58584.2023.10155084.
34. Borissova, D., Dimitrova, Z.: An integrated group decision-making approach considering uncertainty conditions. Business Information Systems, vol. 1, pp. 307-316, 2021, https://doi.org/10.52825/bis.v1i.52.
35. Borissova, D., P. Cvetkova, I. Garvanov, and M. Garvanova, "A framework of business intelligence system for decision making in efficiency management". Lecture Notes in Computer Science, vol. 12133, pp. 111–121, 2020, https://doi.org/10.1007/978-3-030-47679-3_10.
36. Brown, A., Gupta, M., Abdelsalam, M.: Automated machine learning for deep learning based malware detection. Computers & Security, vol. 137, 103582, 2024, https://doi.org/10.1016/j.cose.2023.103582.
37. Chaganti, R., Ravi, V., Pham, T.D.: Deep learning based cross architecture internet of things malware detection and classification. Computers & Security, vol. 120, 102779, 2022, https://doi.org/10.1016/j.cose.2022.102779.
38. Chanajitt, R., Pfahringer, B., Gomes, H.M.: Combining static and dynamic analysis to improve machine learning-based malware classification. In: IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA), Porto, Portugal, 2021, pp. 1-10, https://doi.org/10.1109/DSAA53316.2021.9564144.
39. Chandrashekar, G., Sahin, F.: A survey on feature selection methods. Computers & electrical engineering, vol. 40(1), pp. 16-28, 2014.
40. Christmann, A., Steinwart, I.: Support vector machines. Springer New York, NY, 2008, https://doi.org/10.1007/978-0-387-77242-4.
41. Čisar, P., Maravić Čisar, S., Pásztor, A.: Application of heuristic scanning in malware detection. In: Kovács, T.A., Fürstner, I. (eds) Critical Infrastructure Protection: Ad-vanced Technologies for Crisis

Prevention and Response. NATO ATC 2024. NATO Science for Peace and Security Series C: Environmental Security, pp. 83-97, 2025, https://doi.org/10.1007/978-94-024-2308-2_6.

42. Cvitić, I., D. Peraković, M. Periša, A. Jevremović, A. Shalaginov, An overview of smart home IoT trends and related cybersecurity challenges, Mobile Networks and Applications, vol. 28, pp. 1334-1348, 2023, https://doi.org/10.1007/s11036-022-02055-w.
43. Del Corso, G., Colantonio, S., Caudai, C.: Shedding light on uncertainties in machine learning: formal derivation and optimal model selection. Journal of the Franklin Institute, vol. 362(3), 107548, 2025, https://doi.org/10.1016/j.jfranklin.2025.107548.
44. Dhanya, L., Chitra, R., Anusha Bamini, A.M.: Performance evaluation of various ensemble classifiers for malware detection. Materials Today: Proceedings, vol. 62(7), pp. 4973-4979, 2022, https://doi.org/10.1016/j.matpr.2022.03.696.
45. Dimitrova, Z., D. Borissova, R. Mikhov, and V. Dimitrov, "Group decision-making involving competence of experts in relation to evaluation criteria: Case study for e-commerce platform selection". Communications in Computer and Information Science, vol. 1761, pp. 42-53, 2023, https://doi.org/10.1007/978-3-031-27034-5_3.
46. Dutta, N., Jadav, N., Tanwar, S., Sarma, H.K.D., Pricop, E. Introduction to Malware Analysis. In: Cyber Security: Issues and Current Trends. Studies in Computational Intelligence, vol. 995, pp. 129-141, 2022, https://doi.org/10.1007/978-981-16-6597-4_7 .
47. Farfoura, M.E., Mashal, I., Alkhatib, A., Batyha, R.M.: A lightweight machine learning methods for malware classification. Cluster Computing, vol. 28, 1, 2025, https://doi.org/10.1007/s10586-024-04755-2.
48. Galli, A., La Gatta, V., Moscato, V., Postiglione, M., Sperli, G.: Explainability in AI-based behavioral malware detection systems. Computers & Security, vol. 141, 103842, 2024, https://doi.org/10.1016/j.cose.2024.103842.
49. Garcia, D.E., DeCastro-Garcia, N., Munoz Castaneda, A.L.: An effectiveness analysis of transfer learning for the concept drift problem in malware detection. Expert Systems with Applications, vol. 212, 118724, 2023, https://doi.org/10.1016/j.eswa.2022.118724.
50. Garvanov, I., Garvanova, M.: New approach for smart cities transport development based on the Internet of Things concept. In: 2021 17th Conference on Electrical Machines, Drives and Power Systems (ELMA), Sofia, Bulgaria, pp. 1-6, 2021, https://doi.org/10.1109/ELMA52514.2021.9503084.
51. Gaurav, A., Gupta, B.B., Panigrahi, P.K.: A comprehensive survey on machine learning approaches for malware detection in IoT-based enterprise information system. Enterprise Information Systems, vol. 17(3), 2023764, 2023, https://doi.org/10.1080/17517575.2021.2023764.
52. Gibert, D., Mateu, C., Planes, J.: The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. Journal of Network and Computer Applications, vol. 153, 102526, 2020, https://doi.org/10.1016/j.jnca.2019.102526.
53. Gopinath, M., Sethuraman, S.Ch.: A comprehensive survey on deep learning based malware detection techniques. Computer Science Review, vol. 47, 100529, 2023, https://doi.org/10.1016/j.cosrev.2022.100529.
54. Gorment, N. Z., Selamat, A., Cheng, L. K., Krejcar, O.: Machine learning algorithm for malware detection: Taxonomy, current challenges, and future directions. IEEE Access, vol. 11, pp. 141045-141089 2023, https://doi.org/10.1109/ACCESS.2023.3256979.

55. Gotsev, L., Jekov, B., Kovatcheva, E., Nikolov, R., Barzev, I., Shoikova, E.: A case study: Integrating big data technologies for IoT security into experimental lab session. In: Proc. of INTED2021, pp. 5028-5037 2021, https://doi.org/10.21125/inted.2021.1036.
56. Gu, J., Z. Wang, J. Kuen, L. Ma, A. Shahroudy, B. Shuai, T. Liu, X. Wang, G. Wang, J. Cai, T. Chen, Recent advances in convolutional neural networks, Pattern Recognition, vol. 77, pp. 354-377, 2018, https://doi.org/10.1016/j.patcog.2017.10.013.
57. Guliashki, V., Mankolli, E., Bushati, S.: A machine learning approach improving uni-versity campus security. In: IEEE International Workshop on Technologies for Defense and Security (TechDefense), Rome, Italy, 2023, pp. 341-345, https://doi.org/10.1109/TechDefense59795.2023.10380911.
58. Hancock, J.T., Khoshgoftaar, T.M. CatBoost for big data: an interdisciplinary review. Journal of Big Data, vol. 7, 94, 2020, https://doi.org/10.1186/s40537-020-00369-8.
59. Hasan, R., Biswas, B., Samiun, M., Abu Saleh, M., Prabha, M., Akter, J., Joya, F. H., Abdullah, M.: Enhancing malware detection with feature selection and scaling tech-niques using machine learning models. Scientific Reports, vol. 15, 9122 , 2025, https://doi.org/10.1038/s41598-025-93447-x .
60. Hermawan, D. R., Ghanial Fatihah. M.F., Kurniawati, L., Helen, A.: Comparative study of J48 decision tree classification algorithm, random tree, and random forest on in-vehicle CouponRecommendation Data. In: Proc. of International Conference on Artificial Intelligence and Big Data Analytics, Bandung, Indonesia, pp. 1-6, 2021, https://doi.org/10.1109/ICAIBDA53487.2021.9689701.
61. Ho, T.K., Hull, J.J., Srihari, S.N.: Decision combination in multiple classifier systems. IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 16(1), pp. 66-75, 1994, https://doi.org/10.1109/34.273716.
62. Hochreiter, S., J. Schmidhuber, Long short-term memory, Neural Computation, vol. 9(8), pp. 1735-1780, 1997, https://doi.org/10.1162/neco.1997.9.8.1735.
63. Hou, T., Sun, W., Chen, C., Yang, G., Meng, X., Peng, J.: Marine floating raft aquaculture extraction of hyperspectral remote sensing images based decision tree algorithm. International Journal of Applied Earth Observation and Geoinformation, vol. 111, 102846, 2022, https://doi.org/10.1016/j.jag.2022.102846.
64. Imani, M., Beikmohammadi, A., Arabnia, H.R.: Comprehensive analysis of random forest and XGBoost performance with SMOTE, ADASYN, and GNUS under varying imbalance levels. Technologies, vol. 13, 88, 2025, https://doi.org/10.3390/technologies13030088.
65. Inayat, U., Zia, M.F., Ali, F., Ali, S.M., Khan, H.M.A., Noor, W.: Comprehensive re-view of malware detection techniques. In: International Conference on Innovative Computing (ICIC), Lahore, Pakistan, 2021, pp. 1-6, https://doi.org/10.1109/ICIC53490.2021.9693072.
66. Iranzad, R., Liu, X.: A review of random forest-based feature selection methods for data science education and applications. International Journal of Data Science and Analytics, vol. 20, pp. 197-211, 2025, https://doi.org/10.1007/s41060-024-00509-w
67. Ismael, M.F., Thanoon, K.H.: Investigation malware analysis depend on reverse engineering. In: 2022 International Conference on Data Science and Intelligent Computing (ICDSIC), Karbala, Iraq, pp. 251-256, 2022, https://doi.org/10.1109/ICDSIC56987.2022.10076144.
68. James, G., Witten, D., Hastie, T., Tibshirani, R., Taylor, J.: Linear Regression. In: An Introduction to Statistical Learning. Springer Texts in Statistics. 2023, https://doi.org/10.1007/978-3-031-38747-0_3
69. Jerbi, M., Z.Ch. Dagdia, S. Bechikh, and L.B. Said: Android malware detection as a Bi-level problem. Computers & Security, vol. 121, 102825, 2022, https://doi.org/10.1016/j.cose.2022.102825.

70. Jia, L., Yang, Y., Tang, B., Jiang, Z.: ERMDS: A obfuscation dataset for evaluating ro-bustness of learning-based malware detection system. BenchCouncil Transactions on Benchmarks, Standards and Evaluations, vol. 3(1), 100106, 2023, https://doi.org/10.1016/j.tbench.2023.100106.
71. Jiang, T., Gradus, J.L., Rosellini, A.J.: Supervised Machine Learning: A Brief Primer, Behavior Therapy, vol. 51(5), pp. 675-687, 2020, https://doi.org/10.1016/j.beth.2020.05.002.
72. Kalla, D., Mohammed, A.S., Boddapati, V.N., Jiwani, N., Kiruthiga, T.: Investigating the impact of heuristic algorithms on cyberthreat detection. In: 2nd International Con-ference on Advances in Computation, Communication and Information Technology (ICAICCIT), Faridabad, India, 2024, pp. 450-455, https://doi.org/10.1109/ICAICCIT64383.2024.10912106.
73. Kecman, V.: Support vector machines – An introduction. In: Wang, L. (eds) Support Vector Machines: Theory and Applications. Studies in Fuzziness and Soft Computing, vol. 177, pp. 1-47, 2005, https://doi.org/10.1007/10984697_1.
74. Khalid, O., S. Ullah, T. Ahmad, S. Saeed, D. A. Alabbad, M. Aslam, A. Buriro, and R. Ahmad, "An insight into the machine-learning-based fileless malware detection". Sensors, vol. 23, 612, 2023, https://doi.org/10.3390/s23020612.
75. Khan, K., Khurshid, A., Cifuentes-Faura, J.: Is artificial intelligence a new battleground for cybersecurity? Internet of Things, vol. 28, 101428, 2024, https://doi.org/10.1016/j.iot.2024.101428.
76. Khang, A., V. Abdullayev, V. Hahanov, V. Shah, (Eds.). Advanced IoT Technologies and Applications in the Industry 4.0 Digital Economy, CRC Press. 2024, https://doi.org/10.1201/9781003434269.
77. Kimmell, J.C., Abdelsalam, M., Gupta, M.: Analyzing machine learning approaches for online malware detection in cloud. In: IEEE International Conference on Smart Computing, pp. 189-196, 2021, https://doi.org/10.1109/SMARTCOMP52413.2021.00046.
78. Kittler, J., Hatef, M., Duin, R.P.W., Matas, J.: On combining classifiers. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20(3), pp. 226-239, 1998, https://doi.org/10.1109/34.667881.
79. Kotu, V., Deshpande, B.: Chapter 4 – Classification. Editor(s): Vijay Kotu, Bala Deshpande, Data Science (Second Edition), Morgan Kaufmann, pp. 65-163, 2019, https://doi.org/10.1016/B978-0-12-814761-0.00004-6.
80. Kumar, A., Raj, A., Kumar, R., Ranjan, N.: Malware Detection Using Machine Learning. In: Jain, A., Polkowski, Z., Babo, R., Sharma, N. (eds) Proc. of International Conference on AI Systems and Sustainable Technologies. ASSET 2025. Lecture Notes in Networks and Systems, vol. 1580, 2026, pp 181-191, https://doi.org/10.1007/978-981-95-0629-3_14.
81. Larose, D.T.; Larose, C.D.: k-Nearest Neighbor Algorithm. In Discovering Knowledge in Data: An Introduction to Data Mining, Wiley, pp. 149-164, 2014, https://doi.org/10.1002/9781118874059.ch7.
82. Lebbie, M., Prabhu, S.R., Agrawal, A.K.: Comparative analysis of dynamic malware analysis tools. In: International Conference on Paradigms of Communication, Compu-ting and Data Sciences. Algorithms for Intelligent Systems. pp. 359-3689, 2022, https://doi.org/10.1007/978-981-16-5747-4_31.
83. Lee, J.H., Shi, Z., Gao, Z.: On LASSO for predictive regression. Journal of Econometrics, vol. 229(2), pp. 322-349, 2022, https://doi.org/10.1016/j.jeconom.2021.02.002.
84. Li, X., Yi, S., Cundy, A.B., Chen, W.: Sustainable decision-making for contaminated site risk management: A decision tree model using machine learning algorithms, Journal of Cleaner Production, vol. 371, 133612, 2022, https://doi.org/10.1016/j.jclepro.2022.133612.

85. Ling, X., L. Wu, J. Zhang, Z. Qu, W. Deng, X. Chen, Y. Qian, C. Wu, S. Ji, T. Luo, J. Wu, Y. Wu, Adversarial attacks against Windows PE malware detection: A survey of the state-of-the-art, Computers & Security, vol. 128, 103134, 2023, https://doi.org/10.1016/j.cose.2023.103134.
86. Liu, K., S. Xu, G. Xu, M. Zhang, D. Sun and H. Liu, "A review of Android malware detection approaches based on machine learning". In IEEE Access, vol. 8, pp. 124579-124607, 2020, https://doi.org/10.1109/ACCESS.2020.3006143.
87. Lujan-Moreno, G.A., Howard, P.R., Rojas, O.G., Montgomery, D.C.: Design of experiments and response surface methodology to tune machine learning hyperparameters, with a random forest case-study. Expert Systems with Applications, vol. 109, pp. 195-205, 2018, https://doi.org/10.1016/j.eswa.2018.05.024.
88. Luo, X., Liu, C., Gou, G., Xiong, G., Li, Z., Fang, B.: Identifying malicious traffic under concept drift based on intraclass consistency enhanced variational autoencoder. Science China Information Sciences, vol. 67, 182302, 2024, https://doi.org/10.1007/s11432-023-4010-4.
89. Maniriho, P., Mahmood, A. N., Chowdhury, M.J.M.: API-MalDetect: Automated malware detection framework for windows based on API calls and deep learning techniques, Journal of Network and Computer Applications, vol. 218, 103704, 2023, https://doi.org/10.1016/j.jnca.2023.103704.
90. Maniriho, P., Mahmood, A.N., Chowdhury, M.J.M.: A systematic literature review on Windows malware detection: Techniques, research issues, and future directions. Journal of Systems and Software, vol. 209, 111921, 2024, https://doi.org/10.1016/j.jss.2023.111921.
91. Manthena, H., Kimmel, J.C., Abdelsalam, M., Gupta, M.: Analyzing and explaining black-box models for online malware detection. IEEE Access, vol. 11, pp. 25237-25252, 2023, https://doi.org/10.1109/ACCESS.2023.3255176.
92. Manthena, H., Shajarian, S., Kimmell, J.C., Abdelsalam, M., Khorsandroo, S., Gupta, M.: Explainable artificial intelligence (XAI) for malware analysis: A survey of tech-niques, applications, and open challenges. IEEE Access, vol. 13, pp. 61611-61640, 2025, https://doi.org/10.1109/ACCESS.2025.3555926.
93. Mathews, S.M.: Explainable Artificial Intelligence Applications in NLP, Biomedical, and Malware Classification: A Literature Review. In: Arai, K., Bhatia, R., Kapoor, S. (eds) Intelligent Computing. CompCom 2019. Advances in Intelligent Systems and Computing, vol. 998, pp. 1269-1292, 2019, https://doi.org/10.1007/978-3-030-22868-2_90.
94. Mazurczyk, W., Caviglione, L.: Information hiding as a challenge for malware detection. IEEE Security & Privacy, vol. 13(2), 89–93 (2015), https://doi.org/10.1109/MSP.2015.33.
95. Mehdi, M., Makkar, A., Conway, M.: A comprehensive review of open-source federated learning frameworks. Procedia Computer Science, vol. 260, pp. 540-551, 2024, https://doi.org/10.1016/j.procs.2025.03.232.
96. Melvin, A.A.R., Kathrine, G.J.W.: A quest for best: A detailed comparison between drakvuf-VMI-based and cuckoo sandbox-based technique for dynamic malware analysis. Advances in Intelligent Systems and Computing, vol. 1167, 2021, https://doi.org/10.1007/978-981-15-5285-4_27.
97. Mienye, I. D., Jere, N.: A survey of decision trees: Concepts, algorithms, and applications. IEEE Access, vol. 12, pp. 86716-86727, 2024, https://doi.org/10.1109/ACCESS.2024.3416838
98. Mishra, P., A. Gupta, P. Aggarwal, E.S. Pilli,: vServiceInspector: Introspection-assisted evolutionary bag-of-ngram approach to detect malware in cloud servers. Ad Hoc Networks, vol. 131, 102836, 2022, https://doi.org/10.1016/j.adhoc.2022.102836.

99. Mohamed, N.: Artificial intelligence and machine learning in cybersecurity: a deep dive into state-of-the-art techniques and future paradigms. Knowledge and Information Systems, vol. 67, 2025, pp. 6969-7055, https://doi.org/10.1007/s10115-025-02429-y.
100. Mucherino, A., Papajorgji, P.J., Pardalos, P.M.: k-Nearest Neighbor Classification. In: Data Mining in Agriculture. Springer Optimization and Its Applications, vol. 34, pp. 83-106, 2009, https://doi.org/10.1007/978-0-387-88615-2_4.
101. Mumuni, A., Mumuni, F.: Automated data processing and feature engineering for deep learning and big data applications: A survey. Journal of Information and Intelligence, 2024, https://doi.org/10.1016/j.jiixd.2024.01.002.
102. Niu, Z., Xue, J., Qu, D., Wang, Y., Zheng, J., Zhu, H.: A novel approach based on adap-tive online analysis of encrypted traffic for identifying Malware in IIoT. Information Sciences, vol. 601, pp.162-174, 2022, https://doi.org/10.1016/j.ins.2022.04.018.
103. Nobakht, M., R. Javidan, A. Pourebrahimi, SIM-FED: Secure IoT malware detection model with federated learning, Computers and Electrical Engineering, vol. 116, 109139, 2024, https://doi.org/10.1016/j.compeleceng.2024.109139.
104. Omar, M., Gouveia, L. B., Al-Karaki, J., Mohammed, D.: Reverse-engineering Malware. In M. Dawson, O. Tabona, & T. Maupong (Eds.), Cybersecurity Capabilities in Developing Nations and Its Impact on Global Security, pp. 194-217, 2022, https://doi.org/10.4018/978-1-7998-8693-8.ch010.
105. Or-Meir, O., Nissim, N., Elovici, Y., Rokach, L.: Dynamic malware analysis in the modern rra – A state of the art survey. ACM Computing Surveys, vol. 52(5), 88, 2019, https://doi.org/10.1145/3329786.
106. Ozkan-Okay, M., Akin, E., Aslan, O., Kosunalp, S., Iliev, T., Stoyanov, I., Beloev, I.: A comprehensive survey: Evaluating the efficiency of artificial intelligence and machine learning techniques on cyber security solutions. IEEE Access, vol. 12, pp. 12229-12256, 2024, https://doi.org/10.1109/ACCESS.2024.3355547.
107. Palsa, J., Hurtuk, J., Chovanec, M., Chovancova, E.: Using machine learning algo-rithms to detect malware by applying static and dynamic analysis methods. Acta Poly-technica Hungarica vol. 19(7), pp. 177-196, 2022, https://doi.org/10.12700/APH.19.7.2022.7.10.
108. Pardhi, P.R., Rout, J.K., Ray, N.K.: Implementation of a malware scanner using signa-ture-based approach for android applications. In: 19th OITS International Conference on Information Technology (OCIT), Bhubaneswar, India, 2021, pp. 14-19, https://doi.org/10.1109/OCIT53463.2021.00015.
109. Parmar, A., Brahmbhatt, K. An Optimized Intelligent Malware Detection Framework for Securing Digital Data. Wireless Pers Commun, vol. 133, pp. 351-371, 2023, https://doi.org/10.1007/s11277-023-10771-z.
110. Patil, R., Dudeja, H., Modi, C.: Designing in-VM-assisted lightweight agent-based malware detection framework for securing virtual machines in cloud computing. International Journal of Information Security, vol. 19, pp. 147-162, 2020, https://doi.org/10.1007/s10207-019-00447-w.
111. Pigola, A., de Souza Meirelles, F.: Unraveling trust management in cybersecurity: insights from a systematic literature review. Inf Technol Manag, 2024, https://doi.org/10.1007/s10799-024-00438-x.
112. Pisner, D.A., Schnyer, D.M.: Chapter 6 – Support vector machine. Eds: A. Mechelli, S. Vieira, Machine Learning, pp. 101-121, 2020, https://doi.org/10.1016/B978-0-12-815739-8.00006-7.
113. Praveen, E. Kumar, S. Priyanka, A comprehensive survey on hardware-assisted malware analysis and primitive techniques, Computer Networks, vol. 235, 109967, 2023, https://doi.org/10.1016/j.comnet.2023.109967.

114. Project description – featurewiz. https://pypi.org/project/featurewiz/
115. Puga, J., Krzywinski, M., Altman, N.: Bayes' theorem. Nature Methods, vol. 12, pp. 277-278, 2015, https://doi.org/10.1038/nmeth.3335.
116. Raghuraman, C., S. Suresh, S. Shivshankar, R. Chapaneri, Static and dynamic malware analysis using machine learning, In: Luhach, A., Kosa, J., Poonia, R., Gao, XZ., Singh, D. (eds) First International Conference on Sustainable Technologies for Computational Intelligence. Advances in Intelligent Systems and Computing, vol. 1045, 2020, https://doi.org/10.1007/978-981-15-0029-9_62.
117. Rajamaki, J., Lahdenpera, J., Shalamanov, V.: Design science research towards ECHO governance and management information dystem. In: 12th International Conference on Dependable Systems, Services and Technologies (DESSERT), Athens, Greece, pp. 1-7, 2022, https://doi.org/10.1109/DESSERT58054.2022.10018802.
118. Rani, V., Nabi, S.T., Kumar, M., Mittal, A., Kumar, K.: Self-supervised Learning: A Succinct Review. Archives of Computational Methods in Engineering, vol. 30, pp. 2761-2775, 2023, https://doi.org/10.1007/s11831-023-09884-2
119. Razaque, A., Bektemyssova, G., Yoo, J., S. Hariri, Khan, M.J., Nalgozhina, N., Hwang, J., Khan, M.A." Review of malicious code detection in data mining applications: challenges, algorithms, and future direction. Cluster Computing, vol. 28, 206, 2025, https://doi.org/10.1007/s10586-024-05017-x.
120. Razgallah, A., Khoury, R., Halle, S., Khanmohammadi, K.: A survey of malware detection in Android apps: Recommendations and perspectives for future research. Computer Science Review, vol. 39, 100358, 2021, https://doi.org/10.1016/j.cosrev.2020.100358.
121. Rida, I., Al-Maadeed, N., Al-Maadeed, S., & Bakshi, S.: A comprehensive overview of feature representation for biometric recognition. Multimedia Tools and Applications, vol. 79, pp. 4867-4890, 2020, https://doi.org/10.1007/s11042-018-6808-5.
122. Rizkallah, L.W.: Enhancing the performance of gradient boosting trees on regression problems. Journal of Big Data, vol. 12, 35, 2025, https://doi.org/10.1186/s40537-025-01071-3.
123. Roshinta, T.A., Gabor, S.: A comparative study of LIME and SHAP for enhancing trustworthiness and efficiency in explainable AI systems. In: IEEE International Conference on Computing (ICOCO), Kuala Lumpur, Malaysia, 2024, pp. 134-139, https://doi.org/10.1109/ICOCO62848.2024.10928183.
124. Roy, A., Chakraborty, S.: Support vector machine in structural reliability analysis: A review, Reliability Engineering & System Safety, vol. 233, 109126, 2023, https://doi.org/10.1016/j.ress.2023.109126.
125. Sadhu, P.K., V.P. Yanambaka, and A. Abdelgawad, "Internet of Things: Security and Solutions Survey". Sensors, vol. 22(19), 7433, 2022, https://doi.org/10.3390/s22197433.
126. Salih, A. M., Raisi-Estabragh, Z., Galazzo, I. B., Radeva, P., Petersen, S. E., Lekadir, K., Menegaz, G.: A perspective on explainable artificial intelligence methods: SHAP and LIME. Advanced Intelligent Systems, vol. 7(1), 2400304, 2024, https://doi.org/10.1002/aisy.202400304.
127. Salman, A.M.., Al-Nuaimi, B.T., Ali Subhi, A., Alkattan, H., Alfilh, R.H.C.M: Enhancing Cybersecurity with Machine Learning: A Hybrid Approach for Anomaly Detection and Threat Prediction. Mesopotamian Journal of CyberSecurity, vol. 5(1), pp. 202-215, 2025, https://doi.org/10.58496/MJCS/2025/014.
128. Saravanan, N., Gayathri, V.: Performance and classification evaluation of J48 algorithm and Kendall's based J48 algorithm (KNJ48). International Journal of Computational Intelligence and Informatics, vol. 7(4), 2018.

129. Shakya, A. K., Pillai, G., Chakrabarty, S.: Reinforcement learning algorithms: A brief survey. Expert Systems with Applications, vol. 231, 120495, 2023, https://doi.org/10.1016/j.eswa.2023.120495.
130. Shalaginov, A., Banin, S., Dehghantanha, A., Franke, K.: Machine learning aided static malware analysis: A survey and tutorial. In: Dehghantanha, A., Conti, M., Dargahi, T. (eds) Cyber Threat Intelligence. Advances in Information Security, vol. 70, pp. 7-45, 2018, https://doi.org/10.1007/978-3-319-73951-9_2.
131. Shalamanov, V., Monov, V., Blagoev, I. Z., Matern, S., Vassileva, G., Blagoev, I.I.: A model of ICT competence development for digital transformation. Information & Security, vol. 46(3), pp. 269-284, 2020, https://doi.org/10.11610/isij.4619.
132. Shaukat, K., S. Luo, V. Varadharajan, A novel deep learning-based approach for malware detection, Engineering Applications of Artificial Intelligence, vol. 122, 106030, 2023, https://doi.org/10.1016/j.engappai.2023.106030.
133. Shoorkand, H.D., M. Nourelfath, A. Hajji, A hybrid CNN-LSTM model for joint optimization of production and imperfect predictive maintenance planning, Reliability Engineering & System Safety, vol. 241, 109707, 2024, https://doi.org/10.1016/j.ress.2023.109707.
134. Sihwail, R., K. Omar, K.A.Z. Ariffin, A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis, International Journal on Advanced Science, Engineering and Information Technology, vol. 8(4-2), pp. 1662-1671, 2018, https://doi.org/10.18517/ijaseit.8.4-2.6827.
135. Singh, B., Indu, S., Majumdar, S.: Comparison of machine learning algorithms for classification of Big Data sets, Theoretical Computer Science, vol. 1024, 114938, 2025, https://doi.org/10.1016/j.tcs.2024.114938.
136. Singh, J., Singh, J.: A survey on machine learning-based malware detection in execuТаблица files. Journal of Systems Architecture, vol. 112, 101861, 2021, https://doi.org/10.1016/j.sysarc.2020.101861.
137. Smmarwar, S. K., Gupta, G.P., Kumar, S.: Deep malware detection framework for IoT-based smart agriculture, Computers and Electrical Engineering, vol. 104, Part A, 108410, 2022, https://doi.org/10.1016/j.compeleceng.2022.108410.
138. Smmarwar, S.K., Gupta, G.P., Kumar, S., Kumar, P.: An optimized and efficient android malware detection framework for future sustainable computing. Sustainable Energy Technologies and Assessments, vol. 54, 102852, 2022, https://doi.org/10.1016/j.seta.2022.102852.
139. Smmarwar, S.K., Gupta, G.P., Kumar, S.: Android malware detection and identification frameworks by leveraging the machine and deep learning techniques: A comprehensive review. Telematics and Informatics Reports, vol. 14, 100130, 2024, https://doi.org/10.1016/j.teler.2024.100130.
140. Soja Rani, S., Reeja, S.R.: A survey on different approaches for malware detection using machine learning techniques. In: Karrupusamy, P., Chen, J., Shi, Y. (eds) Sustainable Communication Networks and Application. ICSCN 2019. Lecture Notes on Data Engineering and Communications Technologies vol. 39, pp. 389-398, 2020, https://doi.org/10.1007/978-3-030-34515-0_42.
141. Stankov, I., Tsochev, G.: Vulnerability and protection of business management systems: Threats and challenges. Problems of Engineering Cybernetics and Robotics, vol. 72, pp. 29-40, 2020, https://doi.org/10.7546/PECR.72.20.04.
142. Suthaharan, S.: Support Vector Machine. In: Machine Learning Models and Algorithms for Big Data Classification, Integrated Series in Information Systems, vol. 36, pp. 207-235, 2016, https://doi.org/10.1007/978-1-4899-7641-3_9.

143. Swascan: Ransomware Landscape in Europe. https://www.digitalsme.eu/digital/uploads/Report-H1-Europa-ENG.pdf.
144. Syrris, V., Geneiatakis, D.: On machine learning effectiveness for malware detection in Android OS using static analysis data. Journal of Information Security and Applications, vol. 59, 102794, 2021, https://doi.org/10.1016/j.jisa.2021.102794.
145. Thai, H.-T.: Machine learning for structural engineering: A state-of-the-art review. Structures, vol. 38, pp. 448-491, 2022, https://doi.org/10.1016/j.istruc.2022.02.003.
146. Thakur, P., Kansal, V., Rishiwal, V.: Hybrid deep learning approach based on LSTM and CNN for malware detection. Wireless Personal Communications, vol. 136, 1879-1901, 2024, https://doi.org/10.1007/s11277-024-11366-y.
147. Tian, D., Q. Ying, X. Jia, R. Ma, Ch. Hu, and W. Liu: MDCHD: A novel malware detection method in cloud using hardware trace and deep learning. Computer Networks, vol. 198, 108394, 2021, https://doi.org/10.1016/j.comnet.2021.108394.
148. Tsochev, G., Yoshinov, R.: Research on the security of cyber-physical systems. Education and Knowledge Publishing House, 2020, 258 pages (in Bulgarian).
149. Ucci, D., L. Aniello, and R. Baldoni, "Survey of machine learning techniques for malware analysis". Computers & Security, vol. 81, pp. 123-147, 2019, https://doi.org/10.1016/j.cose.2018.11.001.
150. Urooj, B., Shah, M.A., Maple, C., Abbasi, M.K., Riasat, S.: Malware detection: A framework for reverse engineered Android applications through machine learning algorithms. IEEE Access, vol. 10, pp. 89031-89050, 2022, https://doi.org/10.1109/ACCESS.2022.3149053.
151. Usmani, U.A., Happonen, A., Watada, J.: A review of unsupervised machine learning frameworks for anomaly detection in industrial applications. In: Arai, K. (eds) Intelligent Computing. SAI 2022. Lecture Notes in Networks and Systems, vol.507, pp. 158-189, 2022, https://doi.org/10.1007/978-3-031-10464-0_11
152. Vaza, R.N., R. Prajapati, D. Rathod, and D. Vaghela, "Developing a novel methodology for virtual machine introspection to classify unknown malware functions". Peer-to-Peer Networking and Applications, vol. 15, pp. 793-810, 2022, https://doi.org/10.1007/s12083-021-01281-5.
153. Vivekanandam, B.: Design an adaptive hybrid approach for genetic algorithm to detect effective malware detection in Android division. Journal of Ubiquitous Computing and Communication Technologies, vol. 3(2), pp. 135-149, 2021, https://doi.org/10.36548/jucct.2021.2.006.
154. Wressnegger, C., Freeman, K., Yamaguchi, F., Rieck, K.: Automatically inferring malware signatures for anti-virus assisted attacks. In: Proc. of the 2017 ACM on Asia Conference on Computer and Communications Security, ASIA CCS '17, ACM, New York, NY, USA pp. 587-598, 2017, https://doi.org/10.1145/3052973.3053002.
155. Zhang, S., Liu, J., Zuo, X.: Adaptive online incremental learning for evolving data streams. Applied Soft Computing, vol. 105, 107255, 2021, https://doi.org/10.1016/j.asoc.2021.107255.