Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“___” _____ 20__ р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти

на тему

“Дослідження та програмна реалізація системи кібербезпеки для кластеризації та аналізу даних з веб-ресурсів”

Виконав здобувач вищої освіти
ІІ курсу, групи _____
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Прокопов В.В.
«____» _____ 2021 р.

Керівник проекту
доктор технічних наук, доцент
_____ Єлизавета МЕЛЕШКО
«____» _____ 2021 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *магістр*

Галузь знань *12 "Інформаційні технології"*

Спеціальність *123 "Комп'ютерна інженерія"*

Освітньо-професійна (освітньо-наукова) програма *"Комп'ютерна інженерія"*

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
"____" _____20___року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Прокопов Володимир Вікторович
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи кібербезпеки кластеризації та аналізу даних з веб-ресурсів*

2. Керівник роботи *Мелешко Єлизавета Владиславівна, доктор техн. наук, доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу №42-13 від 02.08.2021 року

3. Строк подання роботи до захисту *20.12.2021 р.*

4. Мета та завдання випускної кваліфікаційної роботи: *Метою розробки є програмне забезпечення системи системи кібербезпеки кластеризації та аналізу даних з веб-ресурсів*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.
2.Перегляд аналогічних існуючих систем.
3. Опис і обґрунтування проектних рішень.
4. Етапи програмування системи.
5. Впровадження системи в промислову експлуатацію.
6. Наукова новизна
7. Економічна ефективність розробленої програми.
8. Заходи з охорони праці та техніки безпеки.
9. Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Діаграма процесів	*1 аркуш*
Показники економічної ефективності	*1 аркуш*

6. Консультанти по роботі, із зазначенням розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В., к.т.н., доцент	25.10.2021	12.11.2021
Охорона праці	Оришака О.В., к.т.н., доцент	04.11.2021	20.11.2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2021 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2021 р.	
3.	Розробка моделі компонента	20.10.2021 р.	
4.	Розробка структур даних	25.10.2021 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2021 р.	
6.	Програмування алгоритмів	10.11.2021 р.	
7.	Розрахунок економічної ефективності	13.11.2021 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2021 р.	
9.	Оформлення ПЗ	17.11.2021 р.	
10.	Попередній захист роботи	28.11.2021 р.	

Дата видачі завдання
«__»_____2021 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
«__»_____2021 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Прокопов В.В. Дослідження та програмна реалізація системи кібербезпеки для кластеризації та аналізу даних з веб-ресурсів. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2021.

В даній магістерській роботі розроблено програмне забезпечення, яке призначено для системи кібербезпеки для кластеризації та аналізу даних з веб-ресурсів.

Метою розробки є програмне забезпечення системи кібербезпеки для кластеризації та аналізу даних з веб-ресурсів.

Результат роботи – програмна реалізація системи кібербезпеки для кластеризації та аналізу даних з веб-ресурсів.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися в на будь-якому ПК з передвстановленим інтерпретатором Python версії 3 і вище та встановленими необхідними бібліотеками.

Програму розроблено на мові програмування Python.

Ключові слова: комп'ютерна інженерія, кібербезпека, аналіз даних, веб-ресурси.

ABSTRACT

Prokopov Vova Research and software implementation of cybersecurity system for clustering and analysis of data from web resources. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2021

In this master's thesis, software has been developed that is designed for a cybersecurity system for clustering and analyzing data from web resources.

The purpose of the d evelopment is cybersecurity system software for clustering and analysis of data from web resources.

The result is a software implementation of a cybersecurity system for clustering and analysis of data from web resources.

In the process of working on the s oftware model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

Instructions for working with software are given.

The program can be used on any PC with a pre -installed Python interpreter version 3 and higher and installed the necessary libraries.

The program is developed in the Python programming language.

Keywords: computer engineering, cybersecurity, data analysis, web resources.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

API – Application Programming Interface, інтерфейс прикладного програмування.

RDS – Relational Database Service, реляційна база даних

БД – база даних.

ІБ – інформаційна безпека.

МН – машинне навчання.

ПЗ – програмне забезпечення.

ПНЕ – помилка невідібраних елементів.

ВСТУП

Актуальність теми. Стрімке розвинення інформаційних технологій, впровадження досягнень науково-технічного прогресу у сучасний побут та масштабна цифровізація багатьох сфер життя, дозволили розглядати сучасний світ у формі формалізованих сукупностей представлень інформації, що піддаються інтерпретації, – даних. Таке представлення навколишнього світу у вигляді абстрактних сутностей дало можливість, використовуючи математичні алгоритми та статистичні методи, досліджувати їх закономірності, взаємозв'язки, процеси та створювати прогнози про явища та події, що відбуваються в ньому. Інакше кажучи, тісне переплетіння обчислювальних потужностей сучасної комп'ютерної техніки та математичного апарату дало людині змогу розпочати спроби навчання не тільки собі подібних та тілесних істот але й машин (комп'ютерів). Таку науку, яка дозволяє машинам навчатися за рахунок обробки величезної кількості інформації, отримувати із даних знання, робити на основі отриманих відомостей прогнози, назвали машинним навчанням. Навчену для виконання якогось певного завдання сутність називають моделлю машинного навчання. У сучасному суспільстві важко уявити сферу людського життя у якій тим чи інакшим чином не були б задіяні навчені та готові до виконання покладених на них завдань такі моделі. Прогнозування акцій, автоматизація виробництва, керування транспортом, фільтри спаму, рекомендація фільмів, музики, літератури, навіть залучення у творчу діяльність (створення музики, картин і т.д.) – все це лише вельми помірний перелік областей у які було впроваджене машинне навчання.

Зважаючи на широке розповсюдження машинного навчання, цілком природно, що цю технологію почали оцінювати з точки зору можливостей впровадження її у сферах пов'язаних зі забезпеченням як інформаційної так і комп'ютерної безпек. Забезпечення комп'ютерної безпеки має всі підстави

розглядатися, як одна із найбільш важливих проблем сучасного суспільства. Так як сучасне суспільство стає все більш залежними від комп'ютерів у роботі, проведенні дозвілля, розвагах, в звичайному житті, в рівних пропорціях зростає і значимість наявності вразливостей і лазівок в комп'ютерних системах, що привертають зовсім недоречну увагу кола вкрай не доброзичливих особистостей, які сподіваються такими способами отримати гроші або просто заподіяти збитків. Можна зауважити навіть більше того, оскільки системи стають все більш складними та взаємопов'язаними, все важче забезпечити відсутність у них помилок та непередбачених лазівок, які відкривають доступ атакуючим.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки кластеризації та аналізу даних з веб-ресурсів.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– огляд існуючих систем кібербезпеки кластеризації та аналізу даних з веб-ресурсів;

– дослідження системи кібербезпеки кластеризації та аналізу даних з веб-ресурсів;

– програмна реалізація системи кібербезпеки кластеризації та аналізу даних з веб-ресурсів.

Об'єктом дослідження є процес аналізу даних з веб-ресурсів у системах кібербезпеки.

Предметом дослідження є методи та алгоритми аналізу даних з веб-ресурсів засновані на лінійних моделях та ансамблевих рішеннях.

Методи дослідження базуються на методах розробки програмного забезпечення, функціональній парадигмі програмування, теорії ймовірності та теорії статистики.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

- удосконалено метод кластеризації та аналізу даних з веб-ресурсів, що

був розроблений на основі алгоритмів машинного навчання, серед яких було визначено найбільш ефективні для виявлення мережевих атак та проведено їх подальшу оптимізацію;

- створено вітчизняний аналог реалізації програм аналізу даних, який невибагливіший до ресурсів комп'ютера та більш ефективно їх використовує і легший у застосуванні.

Практична цінність отриманих результатів полягає в тому, що розроблено алгоритми кластеризації та аналізу даних з веб-ресурсів, які дозволяють успішно вирішувати задачі виявлення мережевих атак у комп'ютерних системах та мережах.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, відповідністю отриманих результатів окремими результатами, наведеними у науковій літературі.

Робота апробована на IV Міжнародній науково-практичній конференції "Інформаційна безпека та комп'ютерні технології" (15-16 квітня 2021 р., м. Кропивницький), за результатами виступу на якій було опубліковано тези доповідей «Дослідження способу приведення текстових даних до зручної для обробки алгоритмами кластеризації форми для аналізу даних з веб-ресурсів».

Зважуючи усе вищезазначене, дослідження та програмна реалізація системи кібербезпеки кластеризації та аналізу даних з веб-ресурсів є актуальною задачею, яка потребує вирішення у магістерській роботі.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Програмне забезпечення призначене для виконання обробки та аналізу інформації має володіти таким функціональними можливостями для роботи з даними: проведення статистичного аналізу даних, систематизації, виявлення закономірностей та тенденцій, графічне (візуальне) представлення.

Загалом людині притаманне чудове розпізнавання шаблонів і закономірностей у графічних зображеннях тож важливим етапом ознайомлення із даними є їх візуалізація, тобто це представлення даних у вигляді, який забезпечує найбільш ефективну роботу людини по їх вивченню. Візуалізація даних знаходить широке застосування в наукових і статистичних дослідженнях (зокрема, в прогнозуванні, інтелектуальному аналізі даних, бізнес-аналізі), в педагогічному дизайні для навчання і тестування, в новинних зведеннях і аналітичних оглядах. Візуалізація даних пов'язана з візуалізацією інформації, інфографікою, візуалізацією наукових даних, розвідувальним аналізом даних і статистичної графікою.

Засоби візуалізації забезпечують:

- стисліть – здатність одночасного відображення великої кількості різнотипних даних;

- відносність і близькість – здатність демонструвати в результатах запиту кластери, відносні розміри груп, схожість і відмінність груп, значення даних, що відрізняються від більшості;

- концентрацію і контекст – взаємодія з деяким обраним об'єктом з можливістю перегляду його положення і зв'язків з контекстом;

- масштабованість – здатність легко і швидко переміщатися між мікро- і макропредставленням;

- орієнтацію на «праву півкулю» – надання користувачеві не тільки заздалегідь встановлених методів роботи з даними (забезпечують його навмисні і сплановані підходи до пошуку потрібної інформації), але і підтримка його інтуїтивних, імпровізаційних когнітивних процесів ідентифікації закономірностей. [1]

Під поняттям інтелектуальний аналіз даних розуміють особливий підхід до опрацювання даних, головним фокус якого націлений на моделюванні і відкритті даних, але не на описані їх. Бізнес-аналітика охоплює аналіз даних, який покладається на агрегацію. Із статистичної точки зору можна поділяти аналіз даних на описову статистику,перевірку статистичних гіпотез і дослідницький аналіз даних. Дослідницький аналіз даних займається відкриттям нових характеристик даних, а перевірка статистичних гіпотез орієнтується на спростування чи підтвердження існуючих гіпотез. Прогнозовий аналіз зосереджується на застосуванні структурних чи статистичних моделей за для класифікації чи передбачення, а аналіз тексту застосовує структурні, статистичні та лінгвістичні методи для вилучення та систематизації інформації з текстових джерел, які відносять до неструктурованих даних. Все це різні прояви інтелектуального аналізу даних.

Попередником аналізу даних є інтеграція даних, під яким слід розуміти сам аналіз даних в якому тісно переплітаються візуалізація даних із поширенням даних. Термін «аналіз даних» іноді використовується як синонім до моделювання даних. [2]

Процесом збору інформації та вимірювання цільових показників в сформованій системі, який згодом дозволяє відповісти на актуальні питання і оцінити отримані результати називають збором даних. Він є частиною досліджень у великій кількості областей пізнання, до яких входять бізнес, громадські науки, гуманітарні науки та фізика. Хоча для різних дисциплін застосовують різні методи, акцент на забезпечення правдивої та точної інформації залишається тим же самим. Метою всього збору даних є отримання

доказ того, що отримані дані є якісними, що дозволить при подальшому їх аналізі дати переконливі і надійні відповіді на поставлені запитання.

Незалежно від галузі вивчення або переваг при визначенні даних (якісних або кількісних), ретельний збір даних є суттєвою складовою для цілісності дослідження. Вибір відповідних інструментів збору даних (існуючі, модифіковані або спеціально розроблені), а також ясно визначені інструкції щодо правильного застосування інструментів скорочують можливість виникнення помилок. Наслідком неправильно зібраних даних може бути: неможливість точно відповісти на питання дослідження; неможливість повторити і перевірити дослідження. [3]

Основним призначенням розроблюваного програмного забезпечення буде використання потужностей сучасного математичного апарату створення алгоритмів машинного навчання (МН) задля проведення аналізу мережевого трафіку з метою виявлення кіберзагроз, відділення зловмисного потоку даних від нормального.

1.2 Область застосування

Підвищений інтерес суспільства до використання машинного навчання призів до його впровадження у повсякденне життя людей, що не оминуло стороною й сферу кібер- та інформаційної безпек.

Поняття захисту інформації існує так само давно як і виникнення самої інформації та осіб для яких вона призначена та тих, які не мають її отримати . З розвитком суспільства, культури, науково-технічного прогресу змінювалися й представлення щодо визначення інформації, способів її зберігання, обробки, використання, способи її представлення та захисту; змінювалися носії, призначенні для зберігання інформації та способи її передачі; формувалися різні інформаційні середовища в контексті яких виникали різні точки зору щодо інформації, її захисту та забезпечення безпеки. У сучасному світі під поняттям

інформаційна безпека розуміють практику із запобігання несанкціонованому доступу, розкриттю, спотворенню, дослідженням, змінам, використанню, запису чи знищенню інформації. Інформаційна безпека є універсальним поняттям, яке застосовується незалежно від форми представлення даних (електронна або фізична). Основним завданням інформаційної безпеки є збалансований захист конфіденційності, цілісності та доступності даних, з урахуванням доцільності застосування та без будь-якої шкоди для продуктивності організації. Це досягається, в основному, за допомогою багатоетапного процесу управління ризиками, що дозволяє ідентифікувати основні засоби та нематеріальні активи, джерела загроз, уразливості, потенційний ступінь впливу та можливості управління ризиками. [4]

Окремим підрозділом від інформаційної безпеки виділяють кібербезпеку - розділ, в рамках якого вивчають процеси функціонування, формування та еволюції кібероб'єктів, для виявлення джерел кібернебезпеки, що утворюються при цьому, визначення їх характеристик, а також їх класифікацію та формування нормативних документів, виконання яких має гарантувати захист кібероб'єктів від усіх виявлених і вивчених джерел кібернебезпеки. [5] На рисунку 1.1 наведено класифікацію існуючих кіберзагроз.

До основних типів кіберзагроз відносять наступні:

- шкідливе програмне забезпечення або вірус – програмне забезпечення, спеціально призначене для завдання шкоди або отримання несанкціонованого доступу до комп'ютерних систем;

- черв'як – автономна шкідлива програма, здатна розмножуватися та копіювати себе на інші комп'ютерні системи;

- троянська програма – шкідлива програма, що видає себе за одну із звичайних програм, щоб уникнути виявлення;

- програма-шпигун – шкідлива програма, встановлена на комп'ютерній системі без дозволу і навіть без відома оператора/користувача для шпигунства та збору інформації;

- рекламне ПЗ – шкідлива програма, яка вводить непередбачені рекламні матеріали (наприклад, спливаючі вікна, банери, відеокліпи) в підсистему інтерфейсу користувача, які найчастіше з'являються при перегляді користувачем веб-контенту.

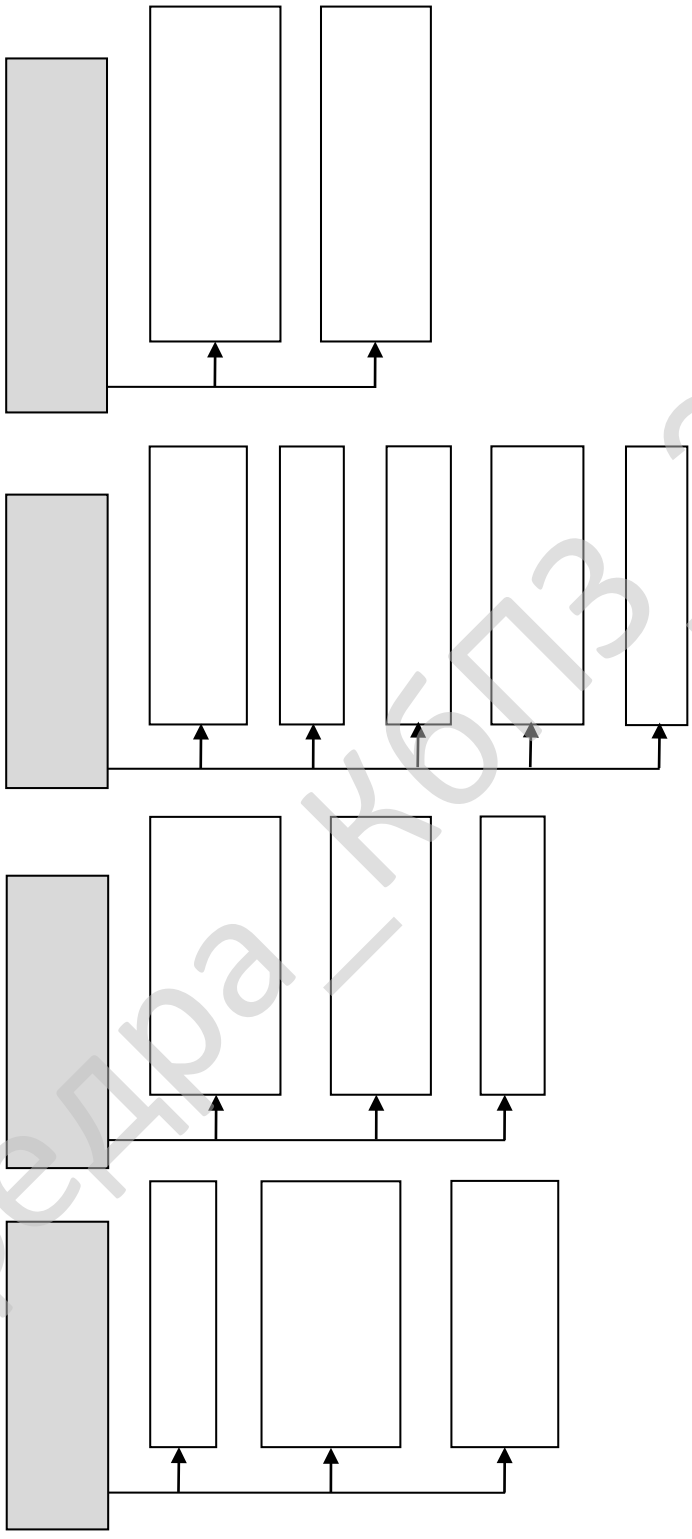

Рисунок 1.1 – Класифікація існуючих кіберзагроз

Основні типи кіберзагроз:

- програма-шантажист – шкідлива програма, спеціально призначена для обмеження функціональних можливостей комп'ютерних систем доти, доки не буде виплачено певну грошову суму (викуп);

- руткіт – комплект ПЗ низького рівня (найчастіше), спеціально призначеного для отримання доступу або повного захоплення управління комп'ютерною системою (root позначає найвищий рівень доступу та управління системою);

- бекдор, або «чорний хід» – навмисно створена або залишена лазівка («дірка»), розміщена на периметрі захисту системи і що дозволяє в майбутньому отримати доступ в обхід підсистеми зовнішнього захисту;

- бот – варіант шкідливої програми, що дозволяє атакуючому у віддаленому режимі перехопити управління комп'ютерними системами, перетворюючи в «зомбі»;

- ботнет, мережа роботів – велика мережа роботів;

- експлойт – фрагмент коду чи програма, що використовує конкретні вразливості інших прикладних програмах чи програмних середовищах;

- сканування: при цьому типі атаки на комп'ютерні системи надсилаються різноманітні запити, часто в режимі простого перебору (грубою сили), з метою виявлення слабких місць та вразливостей, а також для збору інформації;

- перехоплення та аналіз мережевого трафіку – непомітне спостереження та фіксація мережевого трафіку та внутрішнього трафіку на сервері без відома мережевих операторів;

- кейлоггер – деталь апаратури або фрагмент ПЗ (найчастіше приховані від користувача), які фіксують усі натискання клавіш на клавіатурі або дії на іншому пристрої введення;

- спам – незапитані повідомлення, що розсилаються у великих масштабах, найчастіше в рекламних цілях. Зазвичай використовується електронна пошта, але спам також може поширюватися в смс повідомленнях або через провайдера

системи обміну повідомленнями (наприклад, WhatsApp);

- атака під час процедури реєстрації – численні, зазвичай автоматизовані спроби підібрати облікові дані для систем аутентифікації, реалізовані у вигляді простого перебору (грубої сили) або які використовують викрадені/незаконно придбані облікові дані;

- захоплення облікового запису – отримання доступу до чужого облікового запису, як правило, з метою порушення комерційної діяльності, крадіжки особистих даних, викрадення коштів тощо. Зазвичай перехоплення облікового запису є метою атаки під час процедури реєстрації, але також може мати менший масштаб і більш високу цілеспрямованість (наприклад, шпигунське програмне забезпечення, соціальна інженерія).

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень по профілю теми кваліфікаційної магістерської роботи

Використання алгоритмів машинного навчання в сфері кібер- та інформаційної безпеки призвело до створення низки програмних додатків, технологій, фреймворків та рішень.

CyberLympha Thymus – програмний комплекс нового покоління із застосуванням інтелектуальних алгоритмів, що забезпечують автоматичне вивчення схеми інформаційних потоків та роботи окремих вузлів системи, яка знаходиться під захистом, при відсутності специфікацій протоколів і будь-якої додаткової інформації про особливості системи, яка знаходиться під захистом.

Проект CyberThymus – перспективна розробка компанії Сайберлімфа, в якій планується реалізувати сучасні методи навчання без учителя, які дозволяють вивчити систему в повністю автоматичному режимі, для подальшого ефективного виявлення аномалій в її роботі.

Для пошуку аномалій в CyberThymus застосовується метод багатоагентного моделювання, що забезпечує більш високу точність визначення аномалій, а також дозволяє аналізувати причини, за якими алгоритм відніс зафіксоване стан до аномального.

Алгоритм роботи CyberThymus складається з двох основних процесів:

- аналіз мережевого трафіку методом глибокої інспекції, але при цьому структура протоколів, використовуваних в мережі системи, яка підлягає захисту, визначається автоматично в процесі навчання;

- формування багатоагентної моделі системи, яка прогнозує стан системи і дозволяє аналізувати відхилення з реально спостережуваним станом.

Описаний підхід до реалізації CyberThymus володіє явною перевагою над

більшістю рішень виявлення аномалій: аномалія виявляється на моделі системи, а не на основі чисельних відхилень значень характеристичного вектора від деякого еталона.

Модель дозволяє описати аномалію, як мінімум, в такому обсязі:

- аномалія трафіку: виявлення нової адреси, нового значення семантичного пакета або нового протоколу. При цьому, в залежності від стабільності параметра трафіку, для якого виявлено відхилення, можна ранжувати ступінь аномалії: наприклад, якщо за весь час спостережень нові адреси в мережі не з'являлися, то це серйозне відхилення від еталону;

- аномалія поведінки агента: виявлено відхилення вихідного сигналу агента від реального вихідного сигналу вузла. При цьому агент відповідає інтерфейсу вузла, що в реальній системі дозволяє вказати кінцеву точку з точністю до виконуваного процесу на мережевому вузлі – а значить, можна легко виявити причину відхилення за допомогою додаткових засобів захисту або шляхом виконання визначених дій.

Інтеграція CyberThymus з системами моніторингу ІБ (інформаційної безпеки) дозволить зіставити інформацію про аномалії з зареєстрованими подіями ІБ за відповідний часовий період, наприклад, чи зафіксовано подію паралельною системою виявлення вторгнень? Якщо новий трафік не просто раніше не спостерігався, а й має явно шкідливі ознаки, то виявлений інцидент ІБ, який може бути легко локалізовано (знову-таки з точністю до виконуваного процесу на мережевому вузлі, так як адресна інформація будується для всієї ієрархії моделі OSI). Крім того, за допомогою даних про інформаційні активи системи моніторингу ІБ можна додатково отримати збагачену інформацію про виявлений вузол: мережеве ім'я, фізичне розташування, найменування процесу та ін.

Поєднання інформації від систем моніторингу ІБ з виявленими аномаліями засобами CyberThymus підвищує ефективність процесу виявлення інцидентів ІБ, а також допомагає максимально локалізувати активи, порушені

інцидентом ІБ. Це дозволяє, в тому числі, на наступних етапах виконувати автоматичні дії по усуненню наслідків інциденту ІБ і контролювати відповідність системи політиці безпеки. [6]

Amazon Macie – це повністю керований сервіс забезпечення безпеки і конфіденційності даних, що використовує машинне навчання і зіставлення з шаблонами для виявлення і захисту конфіденційних даних в AWS. Інтерфейс зображено на рисунку 2.1.

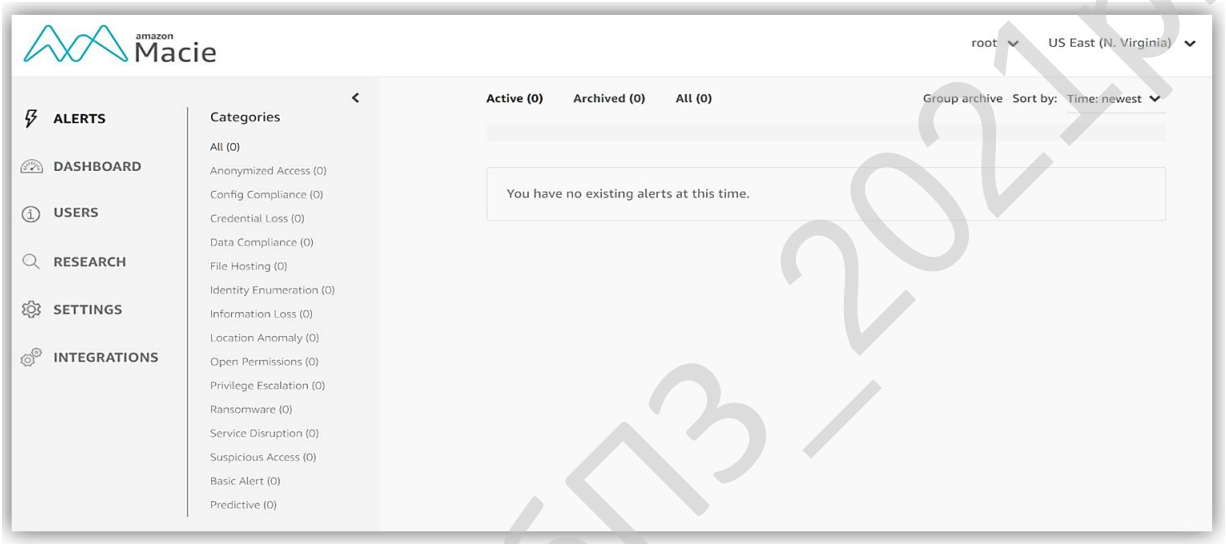

Рисунок 2.1 – Інтерфейс Amazon Macie

У міру зростання обсягу даних, з якими працюють організації, розпізнавання і захист великих обсягів конфіденційних даних стає все більш складним, дорогим і трудомістким завданням. Сервіс Amazon Macie дозволяє автоматизувати виявлення конфіденційних даних і знизити витрати на їх захист. Macie автоматично надає перелік кошиків Amazon S3, включаючи список незашифрованих і загальнодоступних кошиків, а також кошиків, доступ до яких надано акаунтам AWS, не включених до AWS Organizations. Потім сервіс Macie застосовує методи машинного навчання і зіставлення з шаблонами до обраних кошиків, щоб розпізнавати конфіденційні дані, наприклад персональну інформацію, і відправляти повідомлення про них.

Повідомлення (звіти про виявлення) Macie можна знаходити і фільтрувати через Консоль управління AWS, відправляти в Amazon EventBridge (раніше

Amazon CloudWatch Events), щоб легко інтегрувати їх в існуючі робочі процеси і системи обробки подій, або ж використовувати в поєднанні з сервісами AWS, такими як AWS Step Functions, щоб автоматизувати усунення проблем. Це може бути корисно при забезпеченні відповідності нормативним вимогам, таким як Акт про передачу та захисту даних установ охорони здоров'я (HIPAA) або Загальний регламент щодо захисту даних (GDPR).

Переваги:

- виявляє конфіденційні дані в будь-якому масштабі. Сервіс Amazon Macie використовує машинне навчання і зіставлення шаблонів для економічного виявлення конфіденційних даних в будь-якому масштабі. Macie автоматично виявляє великий і постійно розширюваний перелік типів конфіденційних даних, в тому числі персональні дані, такі як імена, адреси і номери кредитних карт. Цей сервіс дозволяє задавати призначені для користувача типи конфіденційних даних для виявлення і захисту даних, специфічних для конкретного бізнесу або прикладу використання;

- наочне уявлення рівня безпеки даних. Amazon Macie забезпечує постійне наочне представлення стану безпеки і конфіденційності даних, що зберігаються в Amazon S3. Macie безперервно оцінює всі кошика S3 аккаунта в автоматичному режимі і попереджає про всі незашифровані і загальнодоступні корзини, а також корзини, доступ до яких надано акаунтам AWS, не включених до AWS Organizations. Macie надає вбудовану підтримку декількох акаунтів, тому відслідковувати загальний рівень безпеки середовища S3 можна за допомогою одного облікового запису адміністратора Macie.

Для підтримки необхідного рівня безпеки даних важливо постійно виявляти конфіденційні дані та проводити оцінку безпеки та засобів керування доступом. Amazon Macie дозволяє робити це по всьому середовищу Amazon S3 і отримувати корисні звіти, на основі яких можна швидко вживати заходів у відповідь, якщо потрібно. Macie також дозволяє виявляти конфіденційні дані в інших сховищах даних, тимчасово переміщуючи їх в S3. Наприклад, можна

створити знімки стану в Amazon Relational Database Service (RDS) або Amazon Aurora, щоб експортувати дані з цих сервісів до Amazon S3, де Macie зможе виявити конфіденційні дані. Вказаним чином за допомогою Macie можна підтримувати конфіденційність та безпеку даних.

Фахівці відділів із забезпечення відповідності вимогам повинні відстежувати розташування конфіденційних даних, належним чином захищати їх та підтверджувати дотримання конфіденційності та безпеки даних, щоб підтримувати відповідність нормативним вимогам. Amazon Macie надає різні варіанти планування аналізу даних, наприклад одноразові, щоденні, щотижневі або щомісячні завдання щодо виявлення конфіденційних даних, які допоможуть підтримувати конфіденційність даних та забезпечувати відповідність вимогам. Macie автоматично відправляє всі вихідні дані завдань з виявлення конфіденційних даних, включаючи звіти, результати оцінки, тимчасові мітки та архівні записи просканованих об'єктів та кошиків у вказаний кошик S3. Ці докладні звіти щодо виявлення конфіденційних даних можна використовувати для аудиту безпеки та конфіденційності даних, а також для тривалого зберігання.

При перенесенні великих обсягів даних в AWS можна налаштувати захищене середовище Amazon S3 як проміжне середовище для виявлення конфіденційних даних за допомогою Macie. Крім того, можна витягти файли з програм, наприклад поштових клієнтів, спільних файлових сховищ або інструментів для спільної роботи, та перенести їх у S3 для оцінки за допомогою Macie. На основі цих результатів можна правильно розмістити дані, що переносяться на зберігання, і визначити необхідні засоби безпеки, такі як шифрування та теги ресурсів. За допомогою звітів Macie можна автоматизувати конфігурацію захисту даних та політики доступу на основі ролей під час перенесення даних до AWS. [7]

DefPloreX – це набір інструментів машинного навчання для великомасштабної судової експертизи електронних злочинів. Це гнучкий набір інструментів, заснований на бібліотеках з відкритим вихідним кодом, для

ефективного аналізу мільйонів пошкоджених веб-сторінок. Вигляд інтерфейсу програми DefPloreX зображено рисунку 2.1.

DefPloreX або Defacement eXplorer використовує поєднання методів машинного навчання та візуалізації даних для перетворення неструктурованих даних на змістовні високорівневі описи. Одним із найцікавіших аспектів DefPloreX є те, що він автоматично групує схожі зламані сторінки у кластери та організовує веб-інциденти у кампанії.

Набір інструментів DefPloreX включає:

- програмний інтерфейс для взаємодії з бекендом Elasticsearch;

- розподілений конвеєр обробки даних на основі програмного рішення Celery Task Queue;

- компонент аналізу для вилучення інформації з веб-сторінок;

- компонент для вилучення ознак (або атрибутів) даних для компактного представлення веб-сторінок у вигляді чисел або категорій;

- статистичний компонент машинного навчання для автоматичного пошуку груп подібних веб-сторінок.

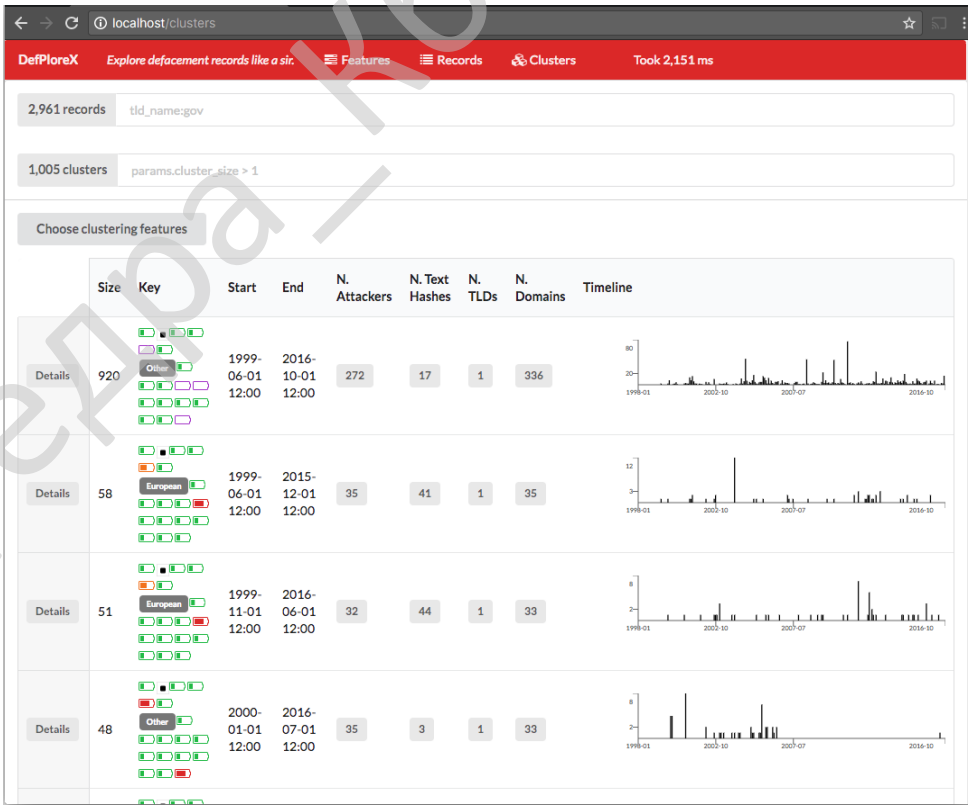

Рисунок 2.2 – Інтерфейс DefPloreX

DefPloreX приймає прості та плоскі табличні файли (наприклад, файли CSV), що містять записи метаданих проаналізованих веб-інцидентів (наприклад, URL-адреси), досліджує їхні ресурси за допомогою безголових браузерів, витягує ознаки з зіпсованих веб-сторінок і зберігає отримані дані. Розподілені браузері без графічного інтерфейсу, як і будь-яка великомасштабна операція з обробки даних, координуються через Celery, де-факто стандарт для розподіленої координації завдань. Використовуючи безліч методів та інструментів аналізу даних на основі Python, DefPloreX створює автономні «представлення» даних, що дозволяє легко їх обробляти та досліджувати. Рисунок архітектури зображено на рисунку 2.3.

Найцікавішим аспектом DefPloreX є те, що він автоматично групує схожі зіпсовані сторінки в кластери та організовує веб-інциденти в групи. Метод кластеризації, який використовується, вимагає лише одного проходу даних, по суті є паралельним і не прив'язаний до пам'яті. DefPloreX пропонує текстові та веб-користувацькі інтерфейси, до яких можна звертатися за допомогою простої мови для розслідувань та криміналістики. Оскільки він заснований на Elastic Search, дані, які виробляє DefPloreX, можна легко інтегрувати з іншими системами.

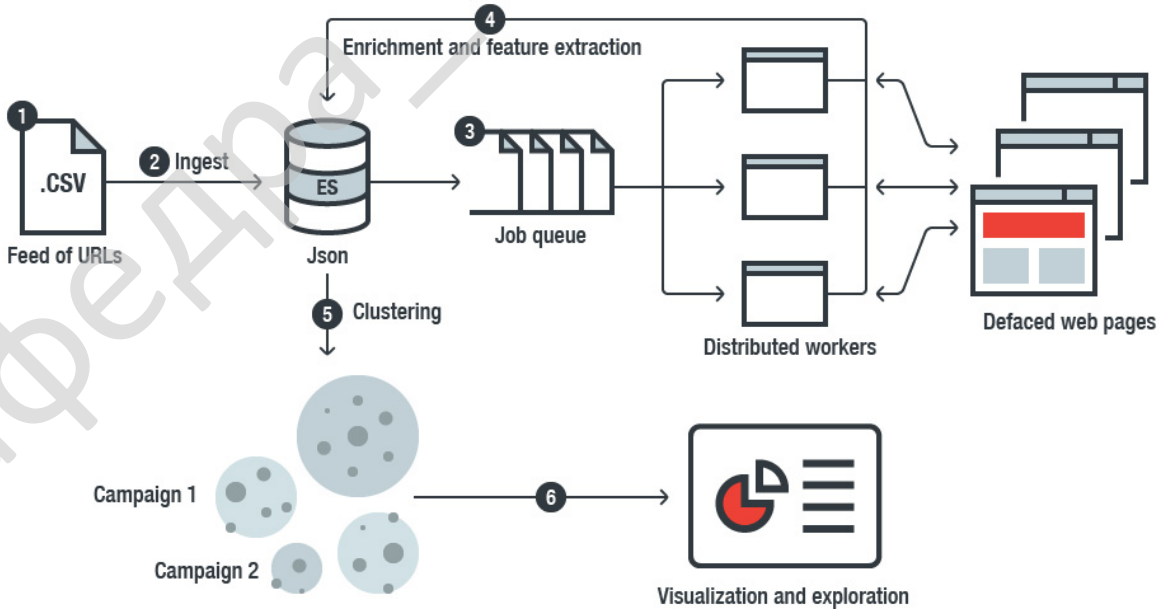

Рисунок 2.3 – Архітектура DefPloreX

DefPloreX підтримує аналітику в таких операціях:

- імпорт і експорт загальних даних до і з Elastic index;

- збагачення Elastic index різними атрибутами;

- відвідування веб-сторінок в автоматизованому, паралельному режимі для вилучення числових і візуальних функцій, які фіксують структуру сторінки HTML та її зовнішній вигляд під час візуалізації;

- постобробка числових і візуальних функцій для отримання компактного представлення, яке описує кожну веб-сторінку (в DefPloreX таке представлення носить назву «bucket»);

- виконання загального перегляду та запитів до Elastic index;

- використання компактного представлення для зведення оригінальних веб-сторінок, групування їх у кластери подібних сторінок. [8]

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Для виконання задач пов'язаних з областю науки про дані необхідно обрати мову програмування, яка забезпечувала програміста засобами для збору інформації, її зберігання, обробки, маніпуляцію, аналізу, синтезу, можливо навіть перетворення та візуалізацією. Одними з найпопулярніших мов програмування для використання у цій області є:

- Python (займає особливе місце серед всіх інших мов програмування. Це об'єктно-орієнтована, гнучка і проста у вивченні мова програмування з відкритим вихідним кодом, що має багатий набір бібліотек та інструментів, призначених для використання у сфері науки про дані. Крім того, у Python є величезна база спільноти, де розробників і дослідників даних, що можуть надавати взаємодопомогу один одному. Наука про дані давно використовує Python, і очікується, що вона залишиться найкращим вибором для вчених і дослідників даних);

- R (це дуже унікальна мова і має деякі дійсно цікаві особливості, яких немає в інших мовах. Ці особливості д уже важливі для програмних додатків для обробки даних. Будучи векторних мовою, R може робити багато речей одночасно, функції можна додаватися до одного вектору, не розміщуючи його в цикл. У міру того як популярність R зростала, вона знаходила своє застосування в самих різних областях, починаючи від фінансових досліджень до генетики, біології та медицини);

- Scala (також відомий як Scalable language, є розширенням мови Java. Він працює на віртуальній машині Java (JVM) і є одним з мов де -факто, коли мова йде практично про роботу з великими даними. Scala служить важливим інструментом для дослідників даних, оскільки він підтримує як анонімні функції, так і функції вищого порядку);

- Javascript (фахівці по роботі з даними повинні володіти знаннями Javascript, оскільки він чудово підходить при необхідності візуалізації даних. Він містить багато бібліотек, які спрощують використання js для візуалізацій, і D3.js є однією з них, і в той же час досить потужною. З випуском Tensorflow.js в 2018 році ця мова тепер здатний запропонувати машинне навчання розробникам JavaScript – як в браузері, так і на стороні сервер). [9]

Серед цих та деяких інших мов програмування було обрано мову Python, через підтримку великої кількості бібліотек (які можуть знадобитися під час виконання завдань від збору даних та до їх візуалізації), гнучкість, простоту та зрозумілість синтаксису.

Python являє собою мову програмування високого рівня, орієнтовану на виконання задач загального призначення. Його високорівневість та лаконічність забезпечують підвищення продуктивності розробки програмних додатків і полегшують (у більшості випадків) читання програмного коду. Синтаксис ядра Python виділяється своїм мінімалізмом. Стандартна бібліотек Python включає до свого складу велику кількість різноманітних корисних функцій широкого спектру призначення (від простих математичних операцій, читання фалів різних

форматів, оброблення послідовностей даних до створення простих графічних зображень).

Python забезпечує підтримку низки різноманітних парадигм програмування: структурну, об'єктно-орієнтовану, функціональну, імперативну і аспектно-орієнтовану. До його основних архітектурних рис можна віднести наступні: динамічна типізація, автоматичне керування пам'яттю (збірка сміття), повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень, високорівневі структури даних. Також в Python забезпечується підтримка модульної архітектури програм: розбиття програм на модулі, які, в свою чергу, можуть об'єднуватися в пакети.

Python приваблює до себе увагу користувачів ряснотою своєї стандартної бібліотеки. Наприклад, влаштовані модулі для написання HTTP-серверів і клієнтів; модулі для парсингу, створення і формування повідомлень електронної пошти; модулі для роботи з форматами зберігання та передачі даних JSON, XML і т.д. Також у стандартну бібліотеку включено набір модулів для забезпечення взаємозв’язку із операційною системою, що дозволяє програмістам писати крос-платформні програми. Серед інших корисних модулів можна виділити наступні: модуль для створення і опрацювання регулярних виразів; модуль для роботи із різними текстовими кодуваннями; модуль для читання і створення файлів мультимедійного типу; модуль для роботи із криптографічними протоколами, архівацією, створення серіалізованих даних, виконання і проведення різних типів тестування та ін.

В додаток до стандартної бібліотеки набору модулів до Python було створено безліч сторонніх рішень та бібліотек, які надають інтерфейс до всіх системних викликів на різних платформах; зокрема, на платформі Win32 підтримуються всі виклики Win32 API, а також COM в обсязі не меншому, ніж у Visual Basic або Delphi. Досить об'ємною є множина прикладних бібліотек для Python, кожні з яких орієнтовані на роботу у різних сферах діяльності (веб, бази даних, обробка зображень, обробка тексту, чисельні методи, додатки

операційної системи і т. д.). [10]

Бібліотека NumPy була розроблена спеціально для виконання математичних операцій лінійної алгебри з масивами різного рівня розмірності. Бібліотека дозволяє досягти продуктивності у областях, пов’язаних із науковими розрахунками, порівняно зі спеціалізованими пакетами. Як приклад, SciPy використовує NumPy і надає доступ до великого спектру математичних алгоритмів (матрична алгебра – BLAS рівнів 1-3, LAPACK, швидке перетворення Фур'є). Numarray спеціально розроблений для операцій з великими обсягами наукових даних. [11]

Бібліотека pandas слугує для обробки і аналізу даних. Бібліотека є надбудовою вищого рівня над NumPy, який є інструментом більш нижчого рівня. Pandas надає у використання своїм користувачам спеціальні структури даних і операції для роботи із числовими таблицями і часовими рядами. Назва бібліотеки походить від економетричного терміна «панельні дані», який використовується для опису багатовимірних структурованих наборів інформації.

Основним призначення бібліотеки pandas є надання інструментів для виконання і проведення робіт пов'язаними з маніпуляціями над даними: збір, очищення даних, але для задач аналізу та моделювання даних, з тим щоби користувачеві не потрібно було переходити на більш вузькоспеціалізовані та призначені суто для статистичної оброки мови (наприклад, такі як R чи Octave). Pandas перш за все призначений для очищення і первинної оцінки даних за загальними показниками, наприклад середнім стандартне відхилення, квантиль і так далі. Статистичною бібліотекою pandas не є, однак набори даних типів DataFrame і Series застосовуються в якості вхідних в більшості модулів аналізу даних і машинного навчання (SciPy, Scikit-Learn). [12]

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на кваліфікаційну магістерську роботу,

реалізації підлягає програмне забезпечення, яке призначено для системи кібербезпеки кластеризації та аналізу даних з веб-ресурсів.

Створення програмного забезпечення пов'язаного із сферами інтелектуального аналізу даних та використанням алгоритмів машинного навчання можна умовно розділити наступним чином на кілька етапів:

- дослідження предметної області;
- вибрати до якої категорії відноситься задача (наприклад,класифікація чи регресія);
- збирання та пошук даних;
- вибір алгоритмів для проведення навчання;
- оцінка продуктивності навчених моделей.

Першим чином необхідно дослідити сучасний стан у досліджуваній предметній області, вивчити які тенденції панують у ній на сучасному етапі, щоб мати інформацію про те, що дійсно варте приділенню уваги, а чим можна знехтувати.

Загалом в області машинного навчання виділяють дві основні задачі: класифікація та регресія. Під класифікацією розуміється навчання моделей розділяти один клас об'єктів від іншого. Задачі регресії полягають у прогнозуванні якогось певного значення оперуючи отриманою раніше інформацією. Так як поставленим завданням є використання алгоритмів аналізу даних для виявлення зловмисних мережевих даних, то дану задачу слід розглядати як задачу класифікації. В залежності від кількості класів вона може бути розділеною на задачі бінарної та мультиклассової класифікацій.

Пошук необхідного набору даних (чи датасету), що задовільнить потреби поставленої задачі, можна вважати чи не найголовнішим етапом, оскільки об'єм інформації у датасеті та якість самих даних має прямо пропорційний вплив на якість тренування моделей. Вибраний датасет потім піддається до форматування та змінам аби привести дані до форми, яка б відповідала вимога м до формату даних, які може сприймати алгоритм.

Далі беручи до уваги тип поставленої задачі, об'єм даних, їх розмірність, кількість класів, на які поділяються дані о бираються алгоритми. Далі проводиться процедура навчання, налаштування гіперпараметрів, після чого відбувається перевірка на тестовому піднаборі та перевірка успішності навчання.

3 ОПИС І ОБГРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Побудова моделі алгоритму машинного навчання, як правило, може складатися з таких етапів: збирання даних, попередня обробка, аналіз, тестування і т.д. На рисунку 3.1 представлено типову схему побудови моделі алгоритму.

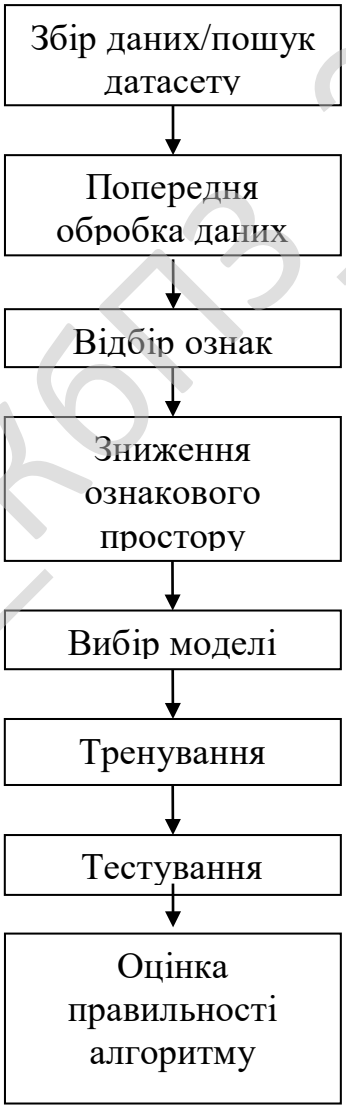

Рисунок 3.1 – Схема побудови моделі машинного навчання

Перед тим як перейти до аналізу та обробки даних, необхідно, перш за все, мати самі дані. Тому найпершим етапом роботи з даними є їх збір.

Збір даних – це процес збору інформації та вимірювання цільових показників в сформованій системі, який згодом дозволяє відповісти на актуальні питання і оцінити отримані результати. Збір даних є частиною досліджень у всіх областях пізнання, включаючи фізику, громадські науки, гуманітарні науки і бізнес. Хоча методи різні для різних дисциплін, акцент на забезпечення точної і правдивої інформації залишається тим же самим. Метою всього збору даних є отримання свідоцтва про якість даних, що дозволяє при аналізі дати переконливі і надійні відповіді на поставлені питання.

Незалежно від галузі вивчення або переваг при визначенні даних (якісних або кількісних), ретельний збір даних є суттєвою складовою для цілісності дослідження. Вибір відповідних інструментів збору даних (існуючі, модифіковані або спеціально розроблені), а також ясно визначені інструкції щодо правильного застосування інструментів скорочують можливість виникнення помилок.

Наслідком неправильно зібраних даних може бути:

- неможливість точно відповісти на питання дослідження;

- неможливість повторити і перевірити дослідження. [13]

Всесвітня мережа Інтернет – це необмежений простір для збору, обробки і передачі даних самих різних форматів.

Для тренування моделі для виявлення кібератак було обрано датасет CSE-CIC-IDS2017. Даний набір даних був підготовлений за результатами аналізу мережевого трафіку в ізольованому середовищі, в якому моделювалися дії низки звичайних користувачів, а також шкідливі дії порушників. Набір даних CICIDS2017 містить найсучасніші поширені атаки, що відповідають вигляду справжніх даних реального світу (PCAP). Він також включає результати аналізу мережевого трафіку за допомогою CICFlowMeter з позначеними потоками на основі відмітки часу, IP -адрес джерела та призначення, портів джерела та призначення, протоколів та атаки (файли CSV).

Створення реалістичного трафіку було головним пріоритетом у створенні цього набору даних. Було використано систему B-profile для профілювання абстрактної поведінки людських взаємодій і створення натуралістичного трафіку. Для цього набору даних було побудовано абстрактну поведінку 25 користувачів на основі протоколів HTTP, HTTPS, FTP, SSH та електронної пошти.

Період збору даних розпочався о 9:00 понеділка, 3 липня 2017 р., і закінчився о 17:00. у п'ятницю, 7 липня 2017 р., загалом 5 днів. Понеділок є звичайним днем і включає лише безпечний трафік. Реалізовані атаки включають брутфорс FTP, брутфорс SSH, DoS, Heartbleed, веб-атаку, інфільтрацію, ботнет та DDoS. Вони виконувалися як вранці, так і вдень у вівторок, середу, четвер і п'ятницю. [14] У таблиці 3.1 наведено опис ознак даних таблиці.

Таблиця 3.1 – Опис ознак даних

Назва ознаки	Опис
Flow ID	Ідентифікатор потоку даних
Source IP	IP-адреса відправника
Source Port	Порт відправника
Destination IP	IP-адреса місця призначення
Destination Port	Порт місця призначення
Protocol	Протокол
Flow Duration	Тривалість потоку
Total Fwd Packets	Кількість переданих у прямому напрямку пакетів
Total Backward Packets	Кількість переданих у прямому зворотному пакетів
Total Length of Fwd Packets	Сумарна довжина пакетів переданих у прямому напрямку
Total Length of Bwd Packets	Сумарна довжина пакетів переданих у зворотному напрямку
Fwd Packet Length Max	Мінімальна довжина переданих пакетів
Fwd Packet Length Min	Максимальна довжина переданих пакетів

Продовження таблиці 3.1

Назва ознаки	Опис
Fwd Packet Length Mean	Середня довжина переданих у прямому напрямі пакетів
Fwd Packet Length Std	Середньоквадратичне відхилення переданих у зворотному напрямі пакетів
Bwd Packet Length Max	Мінімальна довжина переданих пакетів
Bwd Packet Length Min	Максимальна довжина переданих пакетів
Bwd Packet Length Mean	Середня довжина переданих у зворотному напрямі пакетів
Bwd Packet Length Std	Середньоквадратичне відхилення переданих у прямому напрямі пакетів
Flow Bytes/s	Швидкість потоку даних
Flow Packets/s	Швидкість передачі пакетів
Flow IAT Mean	Середнє значення міжпакетного інтервалу
Flow IAT Std	Середньоквадратичне відхилення значення міжпакетного інтервалу
Flow IAT Max	Максимальне значення міжпакетного інтервалу
Flow IAT Min	Мінімальне значення міжпакетного інтервалу
Fwd IAT Total	Загальне значення міжпакетного інтервалу
Fwd IAT Mean	Середнє значення міжпакетного інтервалу у прямому напрямку
Fwd IAT Std	Середньоквадратичне відхилення значення міжпакетного інтервалу у прямому напрямку
Fwd IAT Max	Максимальне значення міжпакетного інтервалу у прямому напрямку
Fwd IAT Min	Мінімальне значення міжпакетного інтервалу у прямому напрямку
Bwd IAT Total	Загальне значення міжпакетного інтервалу у прямому напрямку
Bwd IAT Mean	Середнє значення міжпакетного інтервалу у зворотному напрямку
Bwd IAT Std	Середньоквадратичне відхилення значення міжпакетного інтервалу у зворотному напрямку
Bwd IAT Max	Максимальне значення міжпакетного інтервалу у зворотному напрямку

Продовження таблиці 3.1

Назва ознаки	Опис
Bwd IAT Min	Мінімальне значення міжпакетного інтервалу у зворотному напрямку
Fwd PSH Flags	Кількість разів, коли керуючий біт PSH був встановлений у пакетах, що рухаються у прямому напрямку
Bwd PSH Flags	Кількість разів, коли керуючий біт PSH був встановлений у пакетах, що рухаються у прямому напрямку
Fwd URG Flags	Кількість разів, коли керуючий біт URG був встановлений у пакетах, що рухаються у зворотному напрямку
Bwd URG Flags	Кількість разів, коли керуючий біт URG був встановлений у пакетах, що рухаються у зворотному напрямку
Fwd Header Length	сумарна довжина заголовків пакетів переданих у прямому напрямі
Bwd Header Length	сумарна довжина заголовків пакетів переданих у зворотному напрямі
Fwd Packets/s	Кількість переданих пакетів в секунду
Bwd Packets/s	Кількість переданих пакетів в секунду у зворотному напрямі
Min Packet Length	Мінімальна довжина пакета
Max Packet Length	максимальна довжина пакета
Packet Length Mean	Середня довжина пакета
Packet Length Std	Середньоквадратичне відхилення значення довжини пакета
Packet Length Variance	Дисперсія довжини пакета
FIN Flag Count	Кількість пакетів з керуючим бітом FIN
SYN Flag Count	Кількість пакетів з керуючим бітом SYN
RST Flag Count	Кількість пакетів з керуючим бітом RST
PSH Flag Count	Кількість пакетів з керуючим бітом PSH
ACK Flag Count	Кількість пакетів з керуючим бітом ACK
URG Flag Count	Кількість пакетів з керуючим бітом URG
CWE Flag Count	Кількість пакетів з керуючим бітом CWE

Продовження таблиці 3.1

Назва ознаки	Опис
ECE Flag Count	Кількість пакетів з керуючим бітом ECE
Down/Up Ratio	Співвідношення завантаження та скачувань
Average Packet Size	середня довжина поля даних пакета TCP/IP
Avg Fwd Segment Size	Середня величина сегмента що рухаються у прямому напрямку
Avg Bwd Segment Size	Середня величина сегмента що рухаються у зворотному напрямку
Fwd Avg Bytes/Bulk	Середня кількість байтів у прямому напрямку
Fwd Avg Packets/Bulk	Середня кількість пакетів у прямому напрямку
Bwd Avg Bytes/Bulk	Середня кількість байтів у зворотному напрямку
Bwd Avg Packets/Bulk	Середня кількість байтів у зворотному напрямку
Subflow Fwd Packets	Середня кількість пакетів під потоку, що передані в прямому напрямку
Subflow Fwd Bytes	Середня кількість байтів під потоку, що передані в прямому напрямку
Subflow Bwd Packets	Середня кількість пакетів під потоку, що передані в зворотному напрямку
Subflow Bwd Bytes	Середня кількість байтів під потоку, що передані в зворотному напрямку
act_data_pkt_fwd	Кількість пакетів із принаймні 1 байтом корисного навантаження даних TCP у прямому напрямку
Active Mean	Середній час, коли потік був активним, перш ніж перейти в режим простою
Active Std	Середньоквадратичне відхилення часу, коли потік був активним, перш ніж перейти в режим простою
Active Max	Максимальний час, коли потік був активним, перш ніж перейти в режим простою
Active Min	Мінімальний час, коли потік був активним, перш ніж перейти в режим простою
Idle Mean	Середній час, протягом якого потік простоював, перш ніж став активним
Idle Std	Середньоквадратичне відхилення часу часу, протягом якого потік простоював, перш ніж став активним

Продовження таблиці 3.1

Назва ознаки	Опис
Idle Max	Максимальний час, протягом якого потік простоював, перш ніж став активним
Idle Min	Мінімальний час, протягом якого потік простоював, перш ніж став активним
Label	Мітка класу

Увесь датасет представлений у вигляді кількох файлів у форматі .csv:

- Monday-WorkingHours.pcap_ISCX;
- Tuesday-WorkingHours.pcap_ISCX;
- Wednesday-workingHours.pcap_ISCX;
- Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX;
- Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX;
- Friday-WorkingHours-Morning.pcap_ISCX;
- Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX;
- Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.

Кожен файл частково чи повністю містить дані лише про певні види мережевих атак, тож їх було об'єднано у один єдиний файл за для забезпечення простоти у подальших маніпуляціях, змінах та перетвореннях даних.

Загальні кількість атрибутів, які описують кожний окремий зразок даних, становить близько вісімдесяти. Така велика кількість характеризуючих ознак, хоча і послугує для якнайкраще якіснішого відділення зразків між собою та класів, може в виявитися надлишковою оскільки не кожна ознака може слугувати для виявлення унікальності, що буде виявляти відмінність одного класу від іншого; деякі ознаки взагалі можуть не нести ніякої корисної інформації, яка б описувала дані. Беручи до уваги велику кількість зразків даних та розмір ознакового простору буде доцільним провести відбір ознак. Тобто буде доречно провести процедуру створення такої підмножини ознак, яка буде, при значно меншій порівняно з попередньою кількістю змінних, зведе к мінімуму втрату вагомої для набору даних інформації.

Зниження ознакового простору набору даних дозволить також отримати низку вагомих переваг, як наприклад: підвищення ступеню інтерпретації моделі; збільшити швидкість навчання; зменшення кількості надлишкових даних може призвести до того, що знизиться можливість приймання рішень моделлю з урахуванням «шумів», що буде мати позитивний вплив на якість навчання.

Оскільки самих лише експертних знань (тобто прийняття рішень щодо формування чи відсіювання ознак оперуючись суто апріорними знання про певну сферу знань) може виявитися замало для прийняття рішення щодо відкидання того чи іншого атрибуту даних, то для виявлення важливості (та не важливості) тих чи інших атрибутів слід також прибігти до можливостей, які надають алгоритми машинного навчання. Загалом виділяють три основні стратегії для відбору ознак: одновимірні статистики, відбір на основі моделі та ітеративний відбір. Для вирішення завдання з відбору ознак у рамках реалізації даного програмного забезпечення (ПЗ) було вирішено обрати стратегію відбір на основі моделі за допомогою одного з ансамблевих методів машинного навчання випадковий ліс. Відбір ознак на основі моделі використовує алгоритм машинного навчання з учителем, щоб обчислити важливість кожної ознаки, і залишає лише найважливіші з них. Модель машинного навчання з учителем, яка використовується для відбору ознак, не повинна використовуватись для побудови підсумкової моделі. Модель, що застосовується для відбору ознак, вимагає обчислення певного показника важливості для всіх ознак, щоб характеристики можна було ранжувати за цією метрикою.

Алгоритм випадкового лісу — алгоритм машинного навчання, що полягає у використанні сукупності дерев ухвалення рішень (таку сукупність ще називають ансамблем). Основними напрямками застосування алгориту є задачі класифікації, регресії та кластеризації. Основна ідея полягає у використанні великого ансамблю вирішальних дерев, кожне з яких саме собою дає дуже невисоку якість класифікації, але за рахунок того, що їх використовується велика кількості кінцевий результат виходить хорошим.

Одним із типових напрямків використання випадкових лісів може бути визначення оцінки важливості ознак в задачах регресії і класифікації. З початку для надання оцінки важливості ознак у тренувальному наборі проводиться тренування випадкового лісу на цьому наборі. В продовж процесу побудови моделі для кожного елемента тренувального набору записується так звана помилка невідібраних елементів (ПНЕ). Наступним чином така помилка усереднюється у всьому випадковому лісі для кожної із сутності.

За для того, аби винести оцінку важливості для і-го параметра після тренування, значення і-го параметра випадковим чином перемішуються для всіх записів тренувального набору і виконується обчислення ПНЕ знову. Призначення важливості параметру відбувається шляхом усереднення по всіх деревах різниці показників ПНЕ до перемішування значень та опісля виконання. Під час проведення цього процесу відбувається нормалізація на стандартне відхилення значення для всіх таких помилок.

Важливість параметру вибірки для тренувального набору визначається величиною його значення. Один із вагомих недоліків метода полягає в тому, що для категоріальних змінних із великою кількістю значень метод схильний вважати такі змінні важливішими. Часткове перемішування значень у разі може знижувати вплив цього ефекту. З груп параметрів, які корелюються, важливість яких виявляється однаковою, вибираються менші за чисельністю групи. [15]

Після проведення етапу попередньої обробки даних, зміни значень даних, заповнення пропущених місць, формування ознак, видалення ознак, форматування значень атрибутів наступає етап розділення набору даних на дві різні підмножини: тренувальний та тестовий набори.

Для тренування моделі були вибрані наступні алгоритми машинного навчання:

- наївний баєсів класифікатор. Цей класифікатор називається «наївним», тому що ґрунтується на вельми суворих статис тичних вихідних передумовах, а саме: ознаки вибираються незалежно з деякого (невідомого заздалегідь)

розподілу. Основна ідея, закладена в основу наївного байєсівського класифікатора, полягає в наступному: взявши елемент (точку) даних з набору ознак необхідно визначити ймовірність того, що мітка для цього елемента представляє якийсь певний клас;

- k-найближчих сусідів. Даний алгоритм виділяється з понад інших алгоритмів машинного навчання своєю відносною простотою. О основі принципу роботи лежить наступний процес: запам'ятовування тренувального набору даних. Для проведення прогнозу для нового (невідомого) зразку даних, алгоритм знаходить найближчих сусідів – тобто найближчі до нового зразку точки з навчального набору. Цей тип методики машинного навчання відкладає більшу частину обчислень на час класифікації, замість того щоб виконувати цю роботу під час навчання. Моделі лінивого навчання не навчаються узагальнення даних під час тренувальної стадії. Натомість вони фіксують (записують) усі передані їм точки тренувальних даних і використовують цю інформацію для створення локальних узагальнень на тестовій вибірці під час класифікації;

- дерева рішень. Дерева рішень являють собою досить універсальні та гнучкі моделі навчання з учителем, які мають дуже важливу властивість – простоту інтерпретації. Даний алгоритм – це структура даних у вигляді бінарного дерева, що використовується для прийняття рішень. Дерева є цілком інтуїтивно зрозумілим способом візуального представлення та аналізу даних, тому дуже широко використовуються навіть поза межами області машинного навчання. Ці структури надають можливість прогнозування як категоріальних значень (дерева класифікації), так і значення у форматі дійсних чисел (дерева регресії), а також здатні містити і числові, і категоріальні дані без операцій нормалізації або створення фіктивних змінних; [16]

- метод опорних векторів (SVM). В найпростішому варіанті SVM представляє з себе лінійний класифікатор, тобто дозволяє створити гіперплощину у векторному просторі, щоб спробувати розділити два класи в заданому наборі даних. Метод опорних векторів використовує функцію залежних втрат, яка

штрафує лише ті точки, які розташовані на неправильній стороні відносно гіперплощини або дуже близькі до гіперплощини, але знаходяться на правильній стороні. Більш точно, SVM класифікатор намагається знайти максимальну гіперплощину, що розділяє два класи, де «кордон» позначає відстань від площини, що розділяє векторний простір навпіл, до найближчих точок даних на кожній стороні. У тому випадку, коли дані розділені не прямою лінією, точки всередині цієї межі штрафуються пропорційно їх віддаленості від кордону;

- Adaptive Boosting. Один із способів, яким новий прогнозатор може виправляти свого попередника, полягає в тому, що він приділяє трохи більше уваги навчальним зразкам, на яких попередник мав недонавчання. В результаті нові прогнозатори все більше концентруються на важких випадках. Саме такий прийом застосовує метод Adaptive Boosting. Наприклад, при побудові класифікатора Adaptive Boosting спочатку навчається перший базовий класифікатор, який використовується для вироблення прогнозів на навчальному наборі. Відносна вага некоректно класифікованих ним навчальних зразків збільшується. Другий класифікатор навчається вже із застосуванням оновлених ваг, після чого він використовується для створення прогнозів на навчальному наборі, ваги знову оновлюються і т.д;

- дерева рішень з прискоренням (бустинг градієнта). Такий алгоритм застосовує витонченіші комбінації прогнозів окремих дерев рішень на формування поліпшених узагальнених прогнозів. При використанні методики прискорення або бустингу градієнта кілька слабких об'єктів, що навчаються вибірково об'єднуються за допомогою виконання оптимізації градієнтного спуску в функції втрат, щоб отримати в результаті набагато більш потужну модель навчання. Основною методикою прискорення або бустингу градієнта є додавання окремих дерев до лісу по одному з використанням процедури градієнтного спуску для мінімізації втрат при додаванні дерев. Процедура додавання дерев у ліс зупиняється при досягненні встановленої граничної кількості, коли валідаційний набір втрат досягає прийнятного рівня або якщо

подальше додавання дерев не може поліпшити (мінімізувати) рівень втрат. Основна методика бустинг градієнта була дещо вдосконалена з метою покращення продуктивності, підвищення рівня узагальнення та створення більш ефективних моделей. [17]

Після проведення навчання моделі необхідним є надання оцінки його узагальнюючої здатності, тобто перевірка того наскільки він на основі отриманих знань з навчального набору може їх ефективно використовувати для розпізнавання нових досі невідомих даних. Таку оцінку ефективності ще називають метрики якості чи показник продуктивності моделі.

Найпростішим показником є оцінка правильності моделі. Для її підрахунку необхідно знайти відсоткове відношення правильно класифікованих разків даних.

Більш ефективним показником може слугувати матриця помилок. Суть методу полягає в тому, щоб підрахувати, скільки разів зразки класу А були віднесені до класу В. При розглядані задачі класифікації мережевих даних як задачі бінарної класифікації необхідно буде з'ясувати скільки разів класифікатор плутав звичайні дані з зловмисними (не звичайними). Для розрахунку матриці неточностей спочатку потрібно мати набір прогнозів, щоб їх можна було порівнювати з фактичними цілями.

Кожен рядок у матриці помилок представляє фактичний клас, а кожен стовпець – спрогнозований клас. Перший рядок матриці враховує не звичні мережеві дані (негативний клас): підраховується кількість, які були коректно класифіковані як не звичні дані (істинно негативні класифікації (TN)); далі рахується кількість, що були помилково класифіковані як звичайні дані (хибні класифікації (FP)). Другий рядок матриці враховує звичайні дані (позитивний клас); підраховується кількість помилково класифікованих як не звичні (хибнонегативні класифікації (FN)) і наостанок рахується кількість даних, що були коректно класифіковані як звичайні дані (істинно позитивні класифікації) (TP)). Приклад матриці помилок для співвідношення правильно та не правильно

класифікованих звичайних та зловмисними (не звичайних) даних наведено у вигляді таблиці 3.2.

Таблиця 3.2 – Матриця помилок

		Спрогнозований клас	
		Звичайні	Зловмисні
Фактичний клас	Звичайні	TN	FP
	Зловмисні	FN	TP

Далі, на основі матриці помилок, відбувається розрахунок мір точність, повнота та F1 Точність показує, скільки з передбачуваних позитивних прикладів виявилися справді позитивними. Таким чином, точність – це частка істинно позитивних прикладів від загальної кількості передбачених позитивних прикладів.

$$\text{точність} = \frac{TP}{FP + TP} \tag{3.1}$$

Оцінка повноти показує, скільки від загальної кількості фактичних позитивних прикладів було передбачено як позитивний клас. Повнота – це частка позитивних прикладів від загальної кількості фактичних позитивних прикладів.

$$\text{повнота} = \frac{TP}{FN + TP} \tag{3.2}$$

На основі розрахованих значень повноти та точності розраховується F1. F1 – це середнє гармонійне точності та повноти. У той час, як звичайне середнє трактує всі значення однаково, середнє гармонійне надає низьким значенням більшої ваги. В результаті класифікатор отримає високу міру F1 тільки якщо високими є повнота і точність.

$$F_1 = \frac{2}{\frac{1}{\text{точність}} + \frac{1}{\text{повнота}}} \tag{3.3}$$

3.2 Розробка структурної схеми

Структурна схема дозволяє відобразити всю множину, яку складають елементарні ланки програми і описати сукупність зв'язків, які пов'язують їх, і за допомогою їх представити програмне забезпечення у графічному вигляді. Елементарна ланка системи представляє одну із частин програми, яка реалізує/опису ту чи іншу її складову. На рисунку 3.2 зображено структурну схему програмного забезпечення для обробки та аналізу даних.

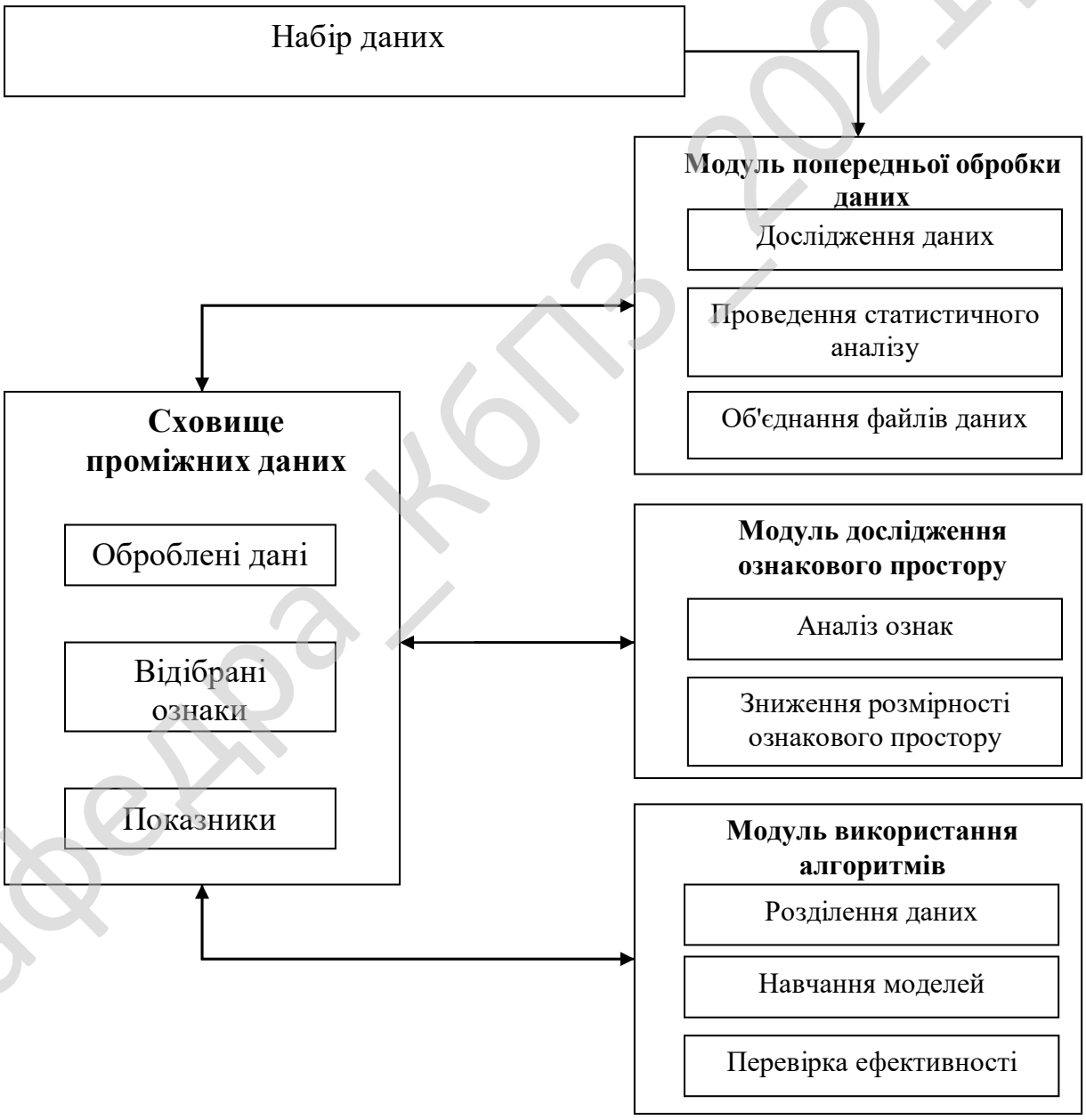

Рисунок 3.2 – Структурна схему ПЗ обробки та аналізу датасету.

Програмне забезпечення обробки та аналізу даних складається з декількох модулів:

- модуль попередньої обробки даних;
- модуль дослідження ознакового простору;
- модуль використання алгоритмів.

Модуль попередньої обробки даних є першою частиною програмного забезпечення, яка має проводити будь-які дії із набором даних, оскільки сам датасет складається із декількох окремих файлів, які цей модуль об'єднує в один, і всі інші модулі будуть працювати виходячи з того, що вони виконують роботу з єдиним цілісним файлом, а не з сукупністю. Окрім поєднання файлів даний модуль, ще виконує низку важливих дій пов’язаних із підготуванням даних до подальшої роботи алгоритмами. Так із самого початку відбувається перевірка всіх файлів набору даних на наявність у них нульових значень, пропусків, типів даних, які не будуть сприйматися алгоритмами машинного навчання і т.д. Також проводиться простий статистичний аналіз кількості даних дата сету з подальшою візуалізацією у вигляді графіків. Після проведення попередньої обробки даних відбувається злиття новоутворених даних в один файл із подальшим збереженням.

Збереження проміжних файлів необхідно для того аби забезпечити модульність функціонування системи, оскільки це дозволить виконувати окремі операції у довільному порядку. Таке рішення необхідне оскільки виконання деяких дій пов’язаних з аналізом та обробкою інформації може потребувати великої кількості часу, а збереження деяких проміжних даних та результатів виконання операцій дозволить уникнути необхідності у поступовому і безперервному виконанні системи і зарадить втраті більшої кількості вагомих даних у разі збою. Також таке зберігання даних на кожному етапі роботи системи має позитивний вплив на економію ресурсів персонального комп’ютера та часу кінцевого користувача.

Наступним компонентом системи є модуль дослідження ознакового

простору. В цьому модулі відбувається виконання операцій пов'язаних із дослідженням ознакового простору даних. З усіх ознаків вибираються найбільш вагомі, значущі, найбільш представницькі для зразків свого класу. При необхідності відбувається формування нових (наприклад за допомогою об'єднання двох), видалення незначущих і т.д.

Модуль використання алгоритмів відповідає за розділення всього дата сету на дві підмножини: навчальний та тестовий набори даних. Далі розпочинається етап тренування моделей алгоритмів машинного навчання та подальша перевірка їх узагальнюючої здатності на тестовому масиві даних за допомогою показників перевірки ефективності.

3.3 Розробка функціональної схеми

Функціональна схема програмного забезпечення для аналізу даних зображено на рисунку 3.3.

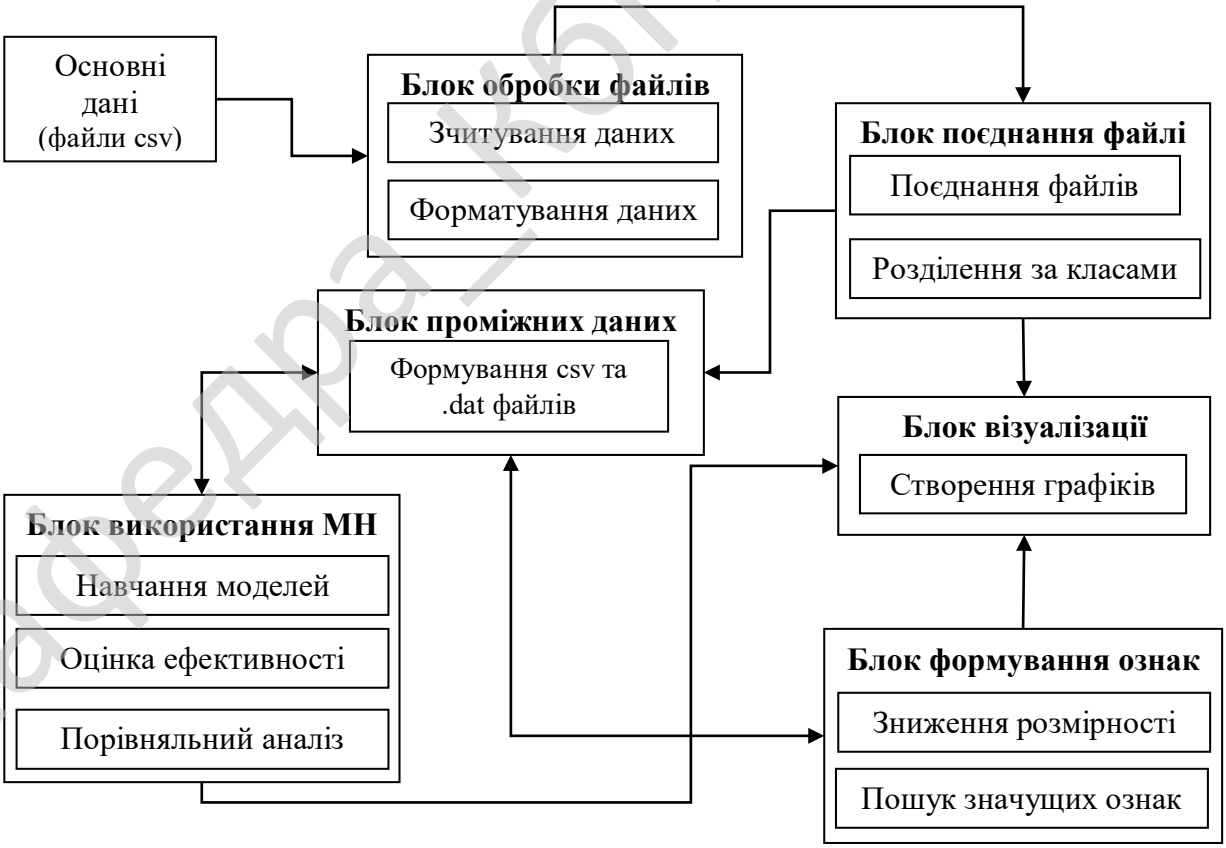

Рисунок 3.3 – Функціональна схема ПЗ аналізу та обробки даних

3.4 Розробка діаграми процесів

Діаграма процесів – візуальне представлення графу діяльностей. Граф діяльностей є різновидом графу станів скінченного автомату, вершинами якого є певні дії, а переходи відбуваються по завершенню дій. Динамічна діаграма процесів програмного забезпечення для аналізу даних зображено на рисунку 3.4.

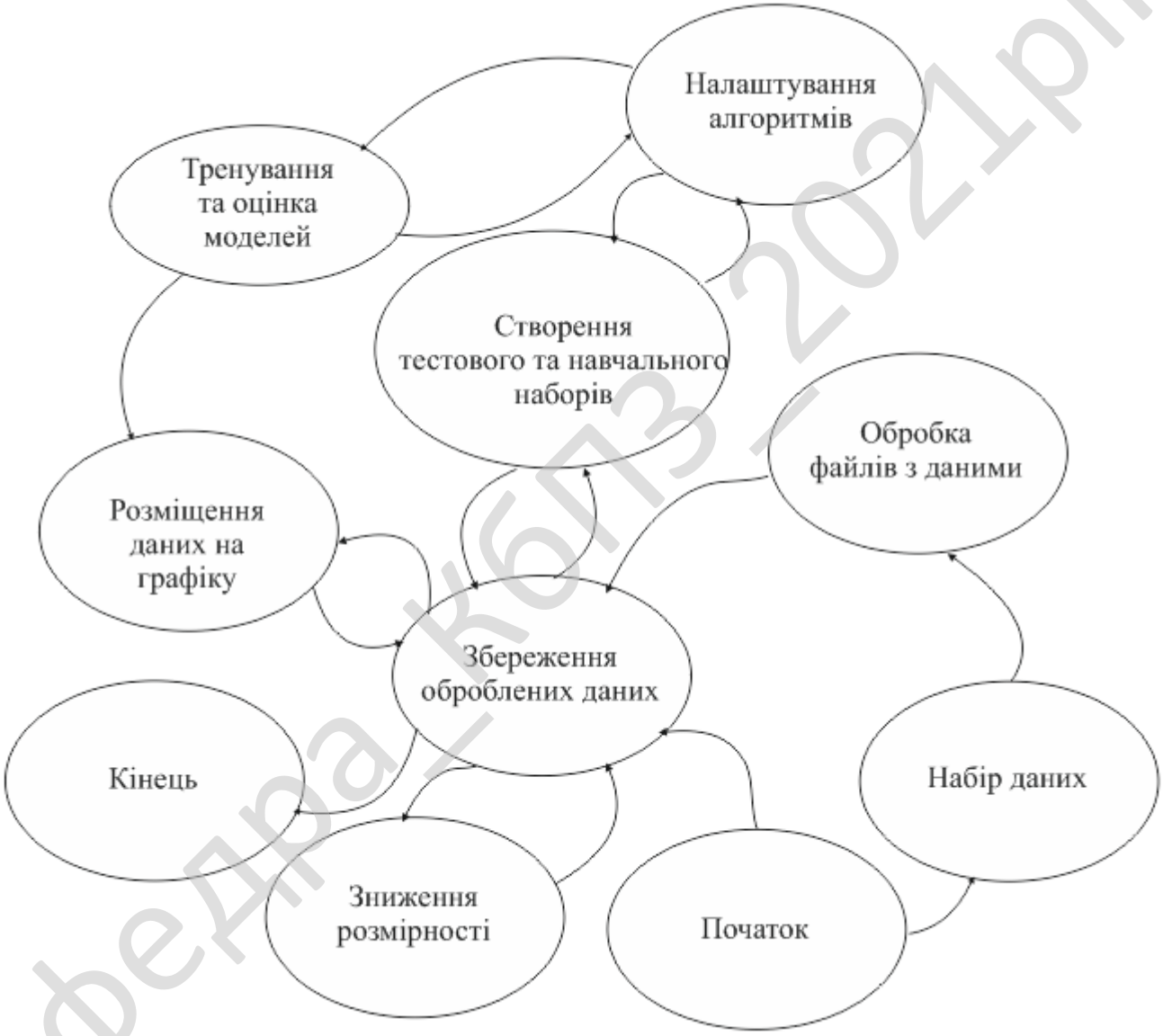

Рисунок 3.4 – Діаграма процесів ПЗ для аналізу та оброблення даних

4 РЕАЛІЗАЦІЯ ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Розробка блок-схем та опис алгоритмів функціонування системи

Для того щоб спроектувати роботу програмного забезпечення, було розроблено алгоритм його роботи. Блок-схему зображено на рисунку 4.1. Алгоритм – набір інструкцій, які описують порядок дій виконавця, щоб досягти результату розв'язання задачі за скінченну кількість дій; система правил виконання дискретного процесу, яка досягає поставленої мети за скінченний час.

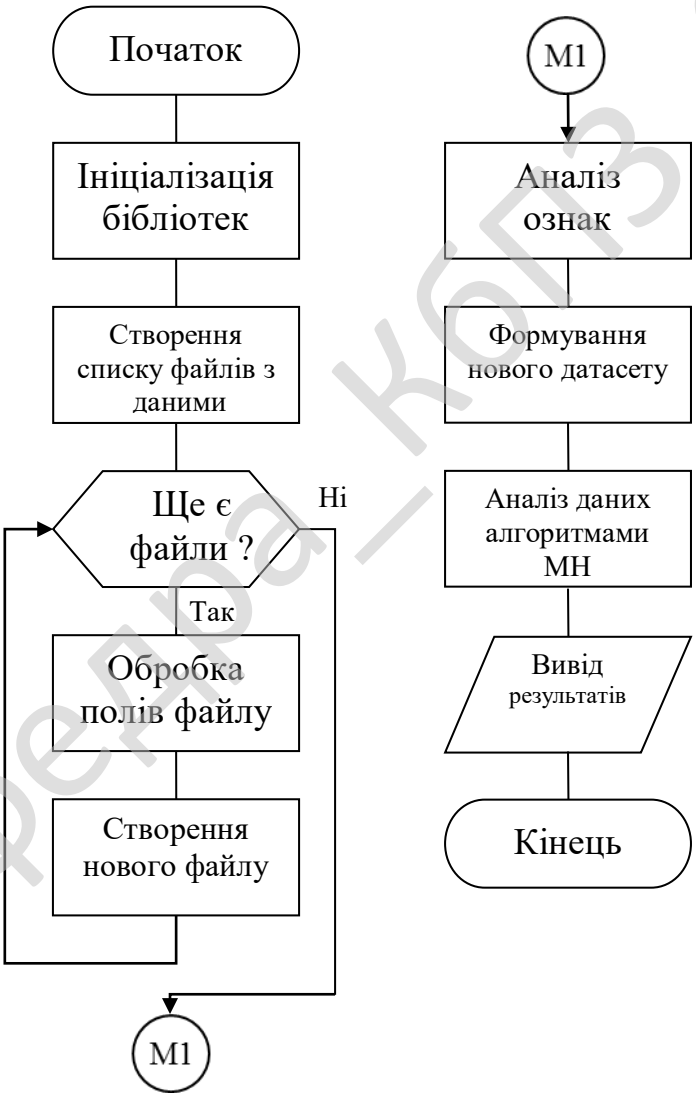

Рисунок 4.1 – Блок-схема роботи програми обробки даних

Початок роботи програми розпочинається з ініціалізації необхідних бібліотек. Вони, бібліотеки, необхідні для виконання операцій по відкриттю файлів, роботи з матрицями та виконанню операцій з ними, представлення даних у форматі необхідному для роботи з ними алгоритмами машинного навчання, візуального представлення даних, виведення даних на графік.

Програма починає роботу із пошуку необхідних файлів у каталозі та формування списку. Далі відбувається відкриття одного файлу з даними по інтернет трафіку і програма починає проводити його обробку. Кожин рядок файлу представляє собою опис певного виду трафіку (зразок даних) , де у колонці вказано значення його ознаки, сукупність яких, тим чи ін акшим чином, відрізняє один зразок від іншого. Відкриття та обхід полів відбувається за допомогою бібліотеки для обробки даних pandas, оскільки вона допомагає зчитувати файли формату csv та перетворювати їх на дані формату Serial та DataFrame – формат представлення даних, який значно полегшує проведення будь-яких операцій з даними (вилучення, сортування, видалення, зміна і т.д.). Проведення даної процедури необхідно для того, щоб здійснити форматування даних, привести їх до виду зрозумілого для язика програмування або навіть позбутися деяких надлишкових ознак, тих, які не несуть якої-небудь вагомої інформації. Також відбувається сортування ранніх видів сутностей за значення поля label, яке вказує чи є сутність шкідливою чи ні (benign – звичайний трафік, а усе інше можна віднести до трафіку зловмісного характеру).

Після завершення операцій форматування та класифікацій відбувається статична обробка новостворених файлів, яка необхідно для того, щоб мати точне представлення про види та кількість полів, що відносяться до тієї чи іншої сутності.

Кожна сутність набору даних має велику кіль кість ознак, яка так чи інакше дозволить алгоритмам машинного навчання відрізнити один вид трафіку від іншого. Проте, незважаючи, на неписане правило, що більша кількість даних та більша кількість їх описових озн ак можуть посприяти кращому навчанню

алгоритмів, слід також брати до уваги два важливих фактори: час виконання та надлишковість ознак відносно кожної окремої сутності. Справа у тому, що не кожна ознака може нести вагому інформацію щодо відділення однієї сутності від іншої, причому деяка описова властивість ознак може бути настільки низька, що її можна не брати до уваги та, як наслідок, видалити. Тож наступним етапом буде виявлення таких ознак для кожного окремого набору сутностей, які несуть найбільшу про них інформації. Для втілення цієї мети використовується алгоритм Random forest, алгоритм машинного навчання, що полягає у використанні ансамблю вирішальних дерев. Алгоритм в даному випадку використовується для обчислення значущості ваги об'єктів у наборі даних.

Важливе місце у аналізі та досліджені даних відводиться їх попередній обробці. Блок-схема роботи алгоритму попередньої оброки даних зображено на рисунку 4.2. Якість даних є першочерговим завданням під час проведення їхнього аналізу. Часто попередня обробка даних стає найважливішою фазою проекту машинного навчання. Під час тренування моделей, за великої кількості зайвої інформації, «зашумлених» і недостовірних даних, отримане знань стає недостовірним та хибним. Дослідження набору даних CSE-CIC-IDS2017 [18] виявили наявність значень ознак, які можуть значно ускладнити роботу алгоритмів машинного навчання, чи призвести до помилок зчитування самих даних. Етап попередньої обробки даних у даному випадку буде включати в себе перевірку кожного значення атрибуту зразку даних, заповнення відсутніх значень, кодування чи/та перероблення даних у формат доступний та зрозумілий для моделей машинного навчанню Так, зокрема, необхідно провести дослідження ознак і замінити значення Infinity на значення -1, замість значення inf поставити 0, теж саме із значеннями типу NaN. Наостанок проводиться відбір усіх атрибутів не числового типу (категоріальні, строкові, змішані і т.і.) та провести їх перетворення.

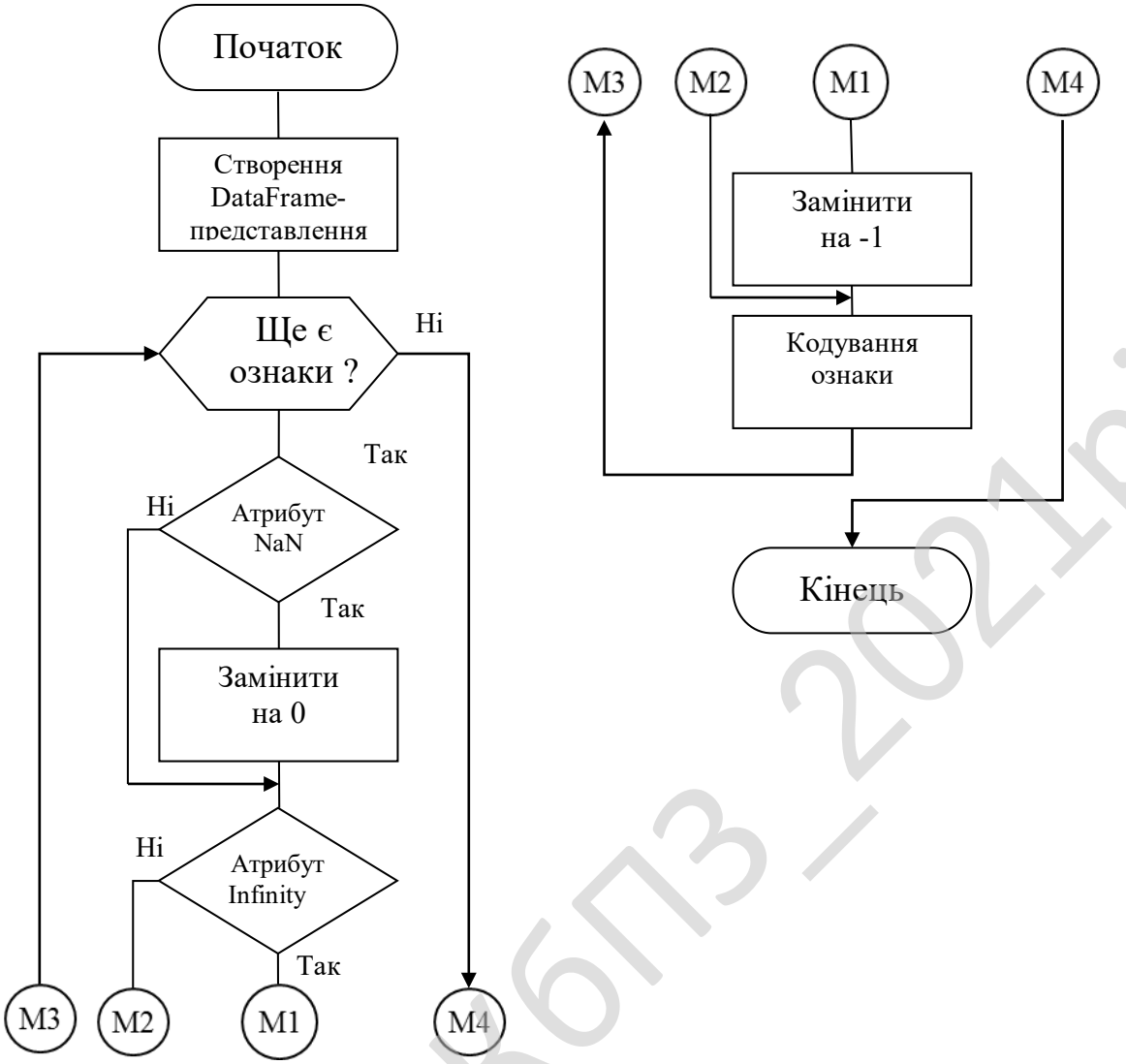

Рисунок 4.2 – Блок-схема роботи алгоритму попередньої обробки даних

Програмна реалізація самого алгоритму виглядає наступним чином. Вагомими кроками є функція read_csv, яка приймає строкове значення шляху до файлу .csv формату і повертає його перетвореним у формат DataFrame. Функція process_data і відповідає за попередню обробку даних. Вона приймає список, який включає імена файлів і далі обходить його, перетворюючи кожний файл один за одним та проводить перевірку значень атрибутів.

```
import os
plots_dir_name = "plot"
stats_dir_name = "stats"
stats_path = plots_dir_name + "\\" + stats_dir_name
```

```
        os.makedirs(processed_data_dir, exist_ok=True)
        os.makedirs(plots_dir_name, exist_ok=True)
        os.makedirs(stats_path, exist_ok=True)

        """
        df = pd.read_csv(csv_data_dir + csv_data[0], sep=",")

        print(df.info())
        print(df.shape)
        """

        df_list = []

        temp_data_file.write(df)
        while True:
                    try:
                        file_data_string=file.readline()
                        if  file_data_string[0] in "".join(map(str, range(1,10)))#
                            if     "     -     "     in     str(file_data_string):
file_data_string=(str(file_data_string).replace(" - "," - "))

file_data_string=(str(file_data_string).replace("inf","0"))

file_data_string=(str(file_data_string).replace("Infinity","0"))


file_data_string=(str(file_data_string).replace("NaN","0"))

                            data.write(file_str))
                        else:
                            continue
                    except Exception as E:
                         print(E)
                        break
        def process_data(files_list):
            for file_name in files_list:
                print("Currently processing file",file_name)
                path_to_file = csv_data_dir + file_name
                datadf = pd.read_csv(
                    path_to_file,
                    sep=',',
                    engine='python',
```

```
                    encoding="cp1250"
                )
                datadf.columns = df.columns.str.strip()
                datadf = datadf.fillna(0)

                datadf.replace("inf", 0, inplace=True)
                datadf.replace("Infinity", -1, inplace=True)
                datadf.replace("NaN", 0, inplace=True)
                datadf.replace(" - "," - ", inplace=True)

                mask = ["Flow Bytes/s", "Flow Packets/s"]
                datadf[mask] = datadf[mask].apply(pd.to_numeric)
                object_features                                    =
list(df.select_dtypes(include=['object']).columns)
                object_features.remove('Label')

                label_encoder = LabelEncoder()
                datadf [object_features] = datadf [object_features].apply(
        lambda feature: label_encoder.fit_transform(feature.astype(str))
        )
                datadf = datadf.drop(data_labels[61], axis=1)

                datadf.to_csv(processed_data_dir + "\\" + file_name, index=False)

                datadf_list.append(df)
```

Тут відбувається створення директорій для збереження трансформованих даних, проведення операції замін значень атрибуті, формування списку перероблених файлів для їх подальшої конкатенації.

Далі наведено частину коду, який відповідає за графічну відображення статистичної інформації про дані. Приклад графіку наведено на рисунку 4.3.

```
        def plot_pie(x, y, name, show=False):
            plt.figure()
            plt.title(name)
            plt.pie(x, labels=y)
            plt.legend()
            plt.savefig(stats_path + "\\" + name + ".png",
                orientation = 'portrait',
                format = 'png',
```

```
            facecolor=(.94, .94, .94)
        )
        if show:
            plt.show()

    def get_data_from_file(file_name):
        path_to_file = processed_data_dir + "\\" + file_name
        datadf = ds.read_csv(path_to_file, sep=',',
                             engine='python',
                             usecols=['Label'])
        datadf = datadf.compute()
        attack_mask = datadf['Label'] != 'BENIGN'
    def bytespdate2num(fmt):
       return mdates.datestr2num(fmt.decode('utf-8'))

    def create_graph():

        fig = plt.figure()
        axis1 = plt.subplot2grid((1,1), (0,0))

        source = requests.get(stock_price_url).text
        data = []

        split_source = source.split('\n')

        for line in split_source[1:]:
            split_line = line.split(',')
            if len(split_line) == 7:
                if "value" not in line:
                    data.append(line)

        date, closep, highp, lowp, openp, adj_closep, volume = np.loadtxt(
            data,
            delimiter=',',
            unpack=True,
            converters={0: bytespdate2num}
        )

        threshold = 100

        axis1.plot_date(date, closep, '-', color="k", label='Price')
        axis1.plot([], [], linewidth=2, label="lose", color='r', alpha=0.5)
```

```
            axis1.plot([], [], linewidth=2, label="gain", color="y", alpha=0.5)
            axis1.axhline(closep[0], color="#090909", linewidth=2)
            axis1.fill_between(date, closep, threshold,
                               where=(closep > threshold), facecolor='y', alpha=.5)
            axis1.fill_between(date, closep, threshold,
                               where=(closep < threshold), facecolor="r", alpha=.5)
            for lbl in axis1.xaxis.get_ticklabels():
               lbl.set_rotation(45)
            axis1.grid(True, color="green", linestyle='dashed', linewidth=0.7)
            axis1.xaxis.label.set_color('c')
            axis1.yaxis.label.set_color('y')
            print(datadf.info())
            print(datadf ["Label"].value_counts())
            df_attack = datadf [attack_mask]

            data_series = datadf.iloc[:, 0].value_counts()
            all_data_stats =  pd.DataFrame({
                'Label': data_series.index,
                'Quantity': data_series.values
            })

            df_attack_series = df_attack.iloc[:, 0].value_counts()
            all_attack_stats =  pd.DataFrame({
                'Label': df_attack_series.index,
                'Quantity': df_attack_series.values
            })

            all_data_stats_sorted = all_data_stats.sort_values(
               by='Quantity', ascending=True
            )

            all_attack_stats_sorted = all_attack_stats.sort_values(
                by='Quantity', ascending=True
            )
            print(all_data_stats)
            print(all_attack_stats)

            all_data_stats.to_csv(stats_path + f"stats_{file_name}", index=False)
            all_attack_stats.to_csv(stats_path    +    f"attack_stats_{file_name}",
index=False)

            plot_data_barh(all_data_stats_sorted['Label'],
```

```
                all_data_stats_sorted['Quantity'],
                name='All data'
)

plot_data_barh(all_attack_stats_sorted['Label'],
                all_attack_stats_sorted['Quantity'],
                name='All attacks'
)
attack_mask_lt_10 = all_attack_stats['Quantity'] < 10000
attack_lt_10 = all_attack_stats[attack_mask_lt_10]
print(attack_lt_10)

plot_pie(attack_lt_10['Quantity'],
          attack_lt_10['Label'],
          name='Attacks less than 10k'
)
```

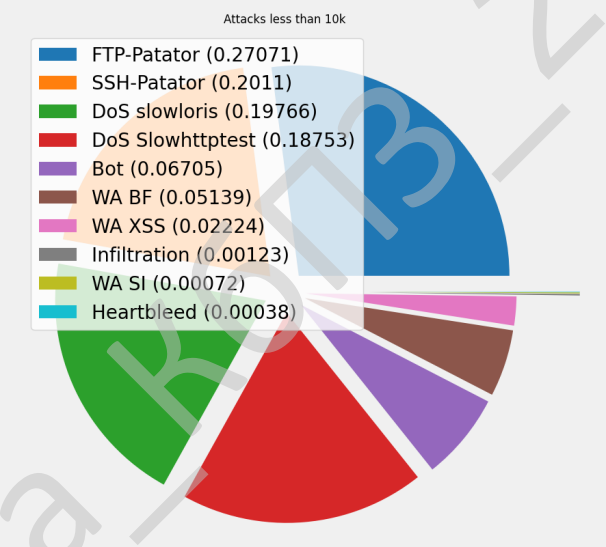

Рисунок 4.3 – Частка кожної атаки

Відбір ознак розпочинається з роботи функцій make_train_test та StratifiedShuffleSplit. Вони відповідають за перетворення вхідного масиву даних у багатовимірні матриці, що складаються із векторів значень ознаків окремих зразків даних. Далі відбувається розділення даних на дві підмножини: тренувальний та тестовий. Розбиття відбувається за допомогою методу стратифікаційного семплювання. Принцип формування стратифікованої вибірк наступний: дані ділиться на однорідні підгрупи, які називають страти, і з кожної

страти витягується правильне число зразків, щоб гарантувати репрезентативність випробувального набору для мережевих даних.

```
os.makedirs(db_dir_name, exist_ok=True)
mlDb = shelve.open(path_db_file)
path_to_data_file = processed_data_dir + "\\" + all_data_files_name
dataset = dk.read_csv(
    path_to_data_file + ".csv",
    engine='python',
    usecols=["Label"]
    )
#dataset = dataset.compute()
#print(dataset["Label"].unique())
#print(dataset["Label"].value_counts())
def make_train_test(data, test_ratio, rand_state=0):
    np.random.seed(rand_state)
    shuffle_ind = np.random.permutation(len(data))
    test_len = int(len(data)* test_ratio)
    test_ind = shuffle_ind[:test_len]
    train_ind = shuffle_ind[test_len: ]
    return data.iloc[train_ind], data.iloc[test_ind]
split = StratifiedShuffleSplit(
    n_splits=1,
    test_size=0.2,
    random_state=0
)
for train_idx, test_idx in split.split(iris.data,iris.target):
    strat_train_data = iris.data.loc[train_idx]
    strat_test_data = iris.data.loc[test_idx]

    strat_train_targ = iris.target.loc[train_idx]
    strat_test_targ = iris.target.loc[test_idx]
```

Далі сформований навчальний підмасив передається до алгоритму випадкового дереву задля визначення вагомості ознак. Вивід роботи випадкового дерева зображено на рисунку 4.4.

```
def data_labels_bin_class(data_labels):
    data_labels_bin = []
    for label in data_labels:
        if label == "BENIGN":
```

```
            data_labels_multi.append(0)
        else:
            data_labels_multi.append(1)

        return data_labels_multi

data_labels_bin_class_list = data_labels_bin_class(dataset["Label"])
data_labels_bin_class_arr = np.asarray(
    data_labels_bin_class_list,
    dtype="int8"
)
rc_forest = RFC(n_estimators=150,random_state=0)
rc_forest.fit(X, y)
featur_importances = rc_forest.feature_importances_
std_rfc_list = []
for tree in rc_forest.estimators_:
        std_rfc_list.append(tree.feature_importances_)
std_rfc = np.std(std_rfc_list,
                 axis=0)
 ind = np.argsort(featur_importances)[::-1]
```

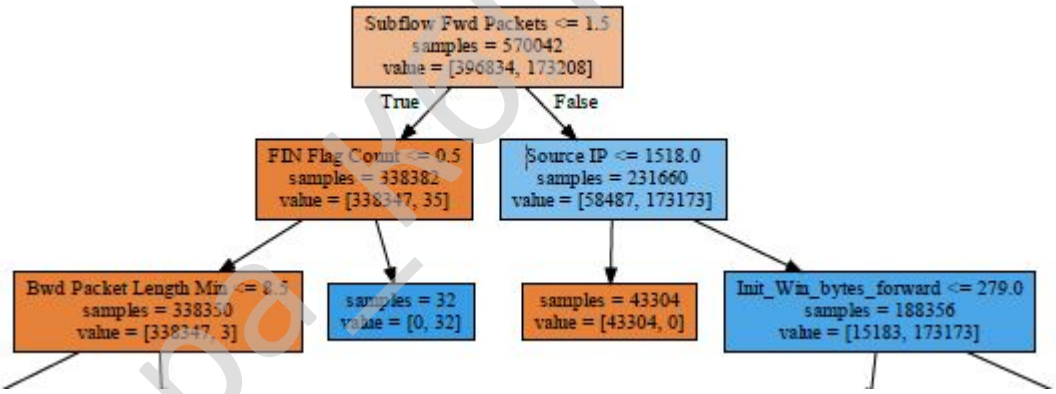

Рисунок 4.4 – Вивід роботи випадкового дерева

Тренування та тестування моделей машинного навчання відбувається із формування списку алгоритмів МН. Потім кожен із них буде проходити процедури навчання, тестування та оцінки ефективності.

```
ml_ alg_list ={
      "Naive Bayes":GaussianNB(),
      "KNN":KNeighborsClassifier(n_neighbors=6),
      "GBC" :GradientBoostingClassifier(),
```

```
            "AdaBoost":AdaBoostClassifier(),
            "DTC":DecisionTreeClassifier(
                  max_depth=4,
                  criterion="entropy"),
            "SVC": SVC(C=1000)
      }

      for ml in ml_list:
          ml_clf = ml_list[ml]

          ml_clf.fit(strat_train_data, strat_train_targ)

          predictor = ml_clf.predict(strat_test_data)
          accuracy_value = ml_clf.score(strat_test_data, strat_test_targ)
          accuracy_value_train = ml_clf.score(strat_train_data, strat_train_targ)
          f_1 = f1_score(strat_test_targ, predictor, average='macro')
          precision_value    =    precision_score(strat_test_targ,    predictor,
average='macro')
          recall_value      =      recall_score(strat_test_targ,      predictor,
average='macro')

          print(f"Name {ml} \n"
                f"Accuracy: {accuracy_value}\n"
                f"Accuracy (train): {accuracy_value_train}\n"
                f"F1 score: {f_1}\n"
                f"Precission: {precision_value}\n"
                f"Recall: {recall_value}\n"
          )
          print("-" * 60)
```

4.2 Захист розробленого програмного забезпечення

Програмне забезпечення поширюється згідно умов вільної ліцензії

Вільна ліцензія – такий ліцензійний, умови якого містять дозвіл користувачеві від власника авторських прав на конкретний перелік способів використання його виробу, які дають йому чотири найважливіші свободи (або свободи, засновані на них і які їх включають – згідно з різними критеріями і видам творів). Щоб вважатися вільною, ліцензія повинна дозволяти:

використовувати в будь-яких цілей виробу автора, вивчати його (в разі ПЗ потрібно доступність початкових кодів), створювати і поширювати копії виробу, вносити у нього зміни, публікувати і поширювати такі змінені похідні вироби (в разі ПЗ потрібно доступність початкових кодів і можливість внесення в них змін). Без такої спеціальної ліцензії ці види використання заборонені законами про захист авторського права, незалежно від того, що про це думає або подумав би автор, тому що майже у всіх країнах світу твори захищаються автоматично без дотримання будь-яких формальностей, всі права закріплюються за автором, а використання його твору забороняється.

Права, перераховані в вільної ліцензії, як правило, надаються будь-якій людині в світі. Як правило, вільні ліцензії безстрокові (на термін дії виключного авторського права), всесвітні, невиключні і безвідкличні (або пов'язані із забезпеченням подібного стану). Більшість вільних ліцензій і законодавств країн вимагає в тому чи іншому вигляді вказувати авторство творців виробу, а також захищати репутацію авторів і їх право на захист виробу від спотворень, нібито зроблених від їх імені. Вільні ліцензії не суперечать авторському права, а саме використовують термінологію законів про авторське право, діють згідно і на підставі цих законів і застосовуються тільки до виробі, на які поширюється захист авторським правом.

Не з точки зору юридичних наук (згідно з якою автор просто розпоряджається своїм твором як хоче), а з точки зору філософії руху вільного програмного забезпечення вільні ліцензії захищають права та свободи користувача на установку, вільне використання, поширення та зміну. [19]

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Дане програмне забезпечення розроблялося з певною та чіткою метою – збору даних та їх подальшої обробки та аналізу з метою виявлення аномалій у веб-трафіку. То ж основними вимогами до користувача будуть наступні:

- знання різних видів кібератак та особливостей кожної з них;
- володіння принципами статистичного аналізу;
- знання принципів машинного навчання та алгоритмів.

Аналіз та обробка отриманих даних проводилася за допомогою використання мови програмування Python та бібліотек написаній на ній. Отже кінцевому користувачеві знадобиться мати на своєму персональному комп'ютері наступні програми та додатки:

- інтерпретатор Python не нижче версії 3;
- бібліотеку для обробки даних pandas;
- бібліотеку scikit-learn;
- бібліотеку для виконання математичних операцій numpy;
- бібліотеку візуалізації та виведення даних на графік matplotlib.

Програма складається з наступних файлів:

- applying_ml_algs.py;
- data_preprocessing.py;
- feature_engineering.py;
- utils.py.

Програма не надає графічного інтерфейсу, то ж робота із нею передбачена лише із терміналу (командний рядок, консоль, shell і т.д.). Запускати файли необхідно у наступній послідовності: data_preprocessing, feature_engineering, applying_ml_algs. Файл utils запуску не потребує оскільки призначений лише для експорту деяких змінних. По завершенню виконання тієї чи іншої операцію відбувається вивід її результату (приклад на рисунку 5.1) чи звичайне

повідомлення про її успішне завершення.

```
|--- Total Fwd Packets <= 1.50
|   |--- FIN Flag Count <= 0.50
|   |   |--- Bwd Packet Length Min <= 8.50
|   |   |   |--- class: 1
|   |   |--- Bwd Packet Length Min >  8.50
|   |   |   |--- Init_Win_bytes_backward <= 232.00
|   |   |   |   |--- class: 1
|   |   |   |--- Init_Win_bytes_backward >  232.00
|   |   |   |   |--- class: 0
|   |--- FIN Flag Count >  0.50
|   |   |--- class: 0
|--- Total Fwd Packets >  1.50
|   |--- Source IP <= 1518.00
|   |   |--- class: 1
|   |--- Source IP >  1518.00
|   |   |--- Init_Win_bytes_forward <= 279.00
|   |   |   |--- Fwd IAT Min <= 106867.00
|   |   |   |   |--- URG Flag Count <= 0.50
```

Рисунок 5.1 – Приклад виводу результату роботи дерева рішень.

6 НАУКОВА НОВИЗНА

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи кібербезпеки кластеризації та аналізу даних з веб-ресурсів.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

- огляд існуючих систем кібербезпеки кластеризації та аналізу даних з веб-ресурсів;

- дослідження системи кібербезпеки кластеризації та аналізу даних з веб-ресурсів;

- програмна реалізація системи кібербезпеки кластеризації та аналізу даних з веб-ресурсів.

Об’єктом дослідження є процес аналізу даних з веб-ресурсів у системах кібрберзпеки.

Предметом дослідження є методи та алгоритми аналізу даних з веб-ресурсів засновані на лінійних моделях та ансамблевих рішеннях.

Методи дослідження базуються на методах розробки програмного забезпечення, функціональній парадигмі програмування, теорії ймовірності та теорії статистики.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

- удосконалено метод кластеризації та аналізу даних з веб-ресурсів, що був розроблений на основі алгоритмів машинного навчання, серед яких було визначено найбільш ефективні для виявлення мережевих атак та проведено їх подальшу оптимізацію;

- створено вітчизняний аналог реалізації програм аналізу даних, який невибагливіший до ресурсів комп'ютера та більш ефективно їх використовує і легший у застосуванні.

7 ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми дипломного проекту

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 48 днів (два місяці).

В магістерській роботі проведене дослідження та програмна реалізація системи кібербезпеки кластеризації та аналізу даних з веб-ресурсів.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп’ютері баз даних;

г) зручність у користуванні та надійність

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт	N	1
2. Кількість екземплярів програм, шт	Ne	160
3. Запланований термін розробки, днів	Fpq	48 (2 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2
7. Кількість макетів вхідної інформації	–	3

Продовження таблиці 7.1

1	2	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	6
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	2
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	2
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	2
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	2
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	2
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПО для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн	–	160000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП є найбільш тривалою і трудомісткою, що робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A\ \text{Size}^B \qquad (7.1)$$

де А – коефіцієнт Боема, А=2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням

$$B = 1{,}01 + 0{,}001\sum W_i \qquad (7.2)$$

де Wi – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

B=1,01+0,001(2,43+3,64+3,38+3,95+2,73) = 1,026

$$T_{ном} = 2{,}45 \cdot 2{,}3^{1{,}026} = 5{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \qquad (7.3)$$

де $ПВj$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 5{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$
$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1) = 7{,}98 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \qquad (7.4)$$

де C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4); S - коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПО згідно встановленим вимогам. Вибираємо в межах (25…350)%

$$T_{РП} = 0{,}3 \cdot 2{,}75 \cdot 7{,}98^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 125 = 206 \text{ люд/день}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	206	Ф 7.1-7.4
Впровадження	13	Д13
Всього	247	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів,

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні,

$$Ч = \frac{247 * 1}{48 - 5} = 6 \text{ ставки}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість обладнання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	385	12	4620	77
Монітор	160	12	1920	32
Клавіатура	140	12	1680	28
Маніпулятор «мишка»	30	12	360	6
Принтер матричний	185	1	185	3
Принтер лазерний	355	2	710	12
Принтер струминний	300	1	300	5
Сканер	155	2	310	5
Концентратор–маршрутизатор	155	2	310	5
Кабельні господарства ЛВС на 1 м. п.	2,5	100	250	4
Кабельне господарство електромережі	48	50	2400	40
Копіювальний апарат	285	2	570	10
Усього за рік:			$З_ч$	227

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{міс}}{1,2} \quad (7.6)$$

$$Ф_{др}^{с} = \frac{227 \cdot 2}{1,2} = 378 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф^{с}_{др}}{F_{др} \cdot T_{зм}} \quad (7.7)$$

$Ч_{ел}$=378/(48·6)=1,31 ставки

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів–електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп’ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2012 R2, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN, PPPoE, Frame Relay, Wi-Fi	0,8	0,2
	Налаштування і конфігурування базової станції безпровідного зв’язку (CMTS)	0,2	
	Розробка та впровадження проектів з організації зв’язку між віддаленими об’єктами, ЛОМ	0,2	
	Забезпечення цілодобової роботи зв’язку клієнтів до мережі Інтернет	0,4	
Всього		1,6	

Продовження таблиці 7.4

Посада	Вид роботи	Час	Кількість штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	2	0,5
	Підтримка постійних клієнтів	1	
	Оформлення договорів, ведення тендерів	0,5	
	Контроль взаєморозрахунків з постачальниками	0,5	
Всього		4	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	0,5	0,2
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,3	
	Розміщення графіки і контенту на Інтернет сторінках	0,3	
Всього		1,6	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,2
	Верстка друкованих видань	0,2	
	Додрукова підготовка макетів	0,2	
	Розміщення графіки і контенту на Інтернет сторінках	0,2	
Всього		1,6	

Складемо штатний розклад виконавців у таблицю 7.5.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньо–місячний оклад, грн.	Всього за період розробки, грн.
Керівник (IT-менеджер)	1	15000	30000
Продакт-менеджер	0,5	12000	12000
Інженер-програміст	6	18600	224000
Інженер-електронщик	1,31	9000	18000
Інженер-системотехнік	0,2	9000	3600
Адміністратор мережі	0,2	11000	4400
Системний програміст	0,2	9000	3600
Дизайнер WEB	0,2	9000	3600
Інженер-верстальник	0,2	9000	3600
Бухгалтер-економіст	0,2	10000	4000
Всього за період розробки	$R_{сп}$=10,01	-	$Ф_{роб}$=306800

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{од} = \frac{306800}{10{,}01 * 48} = 546 \text{ грн}$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 8 робочих місць.

$S_у$ – питома площа на одне робоче місце, ,

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» ціна одного квадратного метра площі новобудови, вік якої не перевищує 25 років, по місту складає 800…1600 . Враховуючи, що курс складає 1 у.о. = 2 6 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м2. На кожне робоче місце у середньому потрібно 8 . З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн на одне робоче місце. Тобто

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де Цм – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7. Дані по оптовій ціні на обладнання та комплектуючі вибирались за комерційною пропозицією фірми Brain за 05.11.21 – джерело http://brain.com.ua/

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп'ютер		11771
Системний блок		7771
Процесор	Intel Core i7-2700K (S1155/4x3.5GHz /5GT/s/8MB/95 Вт)	2500
Системна плата	Huanan B75 (s1155, Intel B75, PCI-Ex16) DVI/VGA/HDMI	1100
Жорсткий диск	HDD 500 Gb SAMSUNG Barracuda HD502HJ (3.5", 500ГБ, 16МБ, SATA II-300)	1290
Оперативна пам'ять	DIMM 4096Mb DDR3 PC3-12800 Kingston, 1600MHz, 512M x 64, CL9-9-9-27, 1.65V, w/heatsink, HyperX	834
DVD-привод	DVD±RW ASUS DRW-24B5ST Black Bulk	416
Корпус	Logicpower 8702 - 550w 12cm	1411
Кардрідер внутрішній	Transcend TS-RDF8K USB 3.0	220
інше	Клавіатура, мишка	Подарунок
Монітор	Монітор BenQ GL2450HM Black	2600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Сканер	Epson Perfection V37	2800
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійного живлення	Powercom BNT-600AP USB	1400

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 - Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп'ютери	6	11771	9416,8	80042,8
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2800	280	3080
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	98594,3

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400

Продовження таблиці 7.8

1	2	3	4
Група 4			
3. Обчислювальна техніка	98594	-	-
Всього по групі	98594	50	49297
Група 5,6			
4. Вимірювальні пристрої	5190	-	-
5. Транспортні засоби	0	-	
6. Господарський інвентар	28000	-	-
Всього по групі	33190	20	35238
7. Нематеріальні активи	160000	10	16000
Разом	$К_р$= 1699784		$А_р$= 170935

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де $N_е$ – Кількість екземплярів програм, шт.

$$З_о = 546 \cdot 247/160 = 842 \text{ грн}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де $Н_q$ – норматив додаткової зарплати, %

$$З_д = 842 \cdot 10 \cdot 0{,}01 = 84 \text{ грн}$$

Відрахування на соціальні потреби за нормативом Нс=22% від суми основної та додаткової зарплати

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де H_c – відрахування на соціальні потреби, %

$$C_{оц} = 0{,}01 \cdot 22(842+84) = 203 \text{ грн}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом Нг=15% від основної зарплати

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де $Н_г$ – загальногосподарські витрати, %

$$Г_{осп} = 842 \cdot 15 \cdot 0{,}01 = 126 \text{ грн}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де $З_{М1}$ – вартість паперу, грн., $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн., $З_{М3}$ – вартість фарби, картріджей, тонеру, грн., $N_е$ – кількість екземплярів програм, шт.

Згідно виданих викладачем норм приймаємо одну пачку паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=105 грн., визначаємо вартість паперу за період розробки $N_м$=2 міс:

$$З_{М1} = Ц_п \cdot n_р \cdot N_м. \quad (7.16)$$

$$З_{М1} = 105 \cdot 0{,}33 \cdot 2 = 69 \text{ грн.}$$

Згідно виданих викладачем норм до вартості запам'ятовуючих пристроїв входить вартість CD дисків в кількості, що дорівнює кількості ек земплярів програм та одного DVD диска для збереження резервної копії програми:

$$З_{М2} = \sum Ц_д., \quad (7.17)$$

де $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 3 грн/шт., DVD-R LG 4,7Gb, 16x speed Cake box - 10 грн/шт.

$$З_{М2} = 160 \cdot 3 + 10 = 490 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та

заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{МЗ} = 574+558+570=1702 \text{ грн.}$$

$$З_М = (69 + 490 + 1702)/160=12 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де $Н_п$ – норматив витрат на освоєння нових мов програмування, %

$$О_п = 842 \cdot 15 \cdot 0{,}01=126 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм (Ne = 160 прим.)

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де $А_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 170935 \cdot 2/(160 \cdot 12)=178 \text{ грн}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції

$$С_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$С_п = 842+84+203+126+12+126+178=1571 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності (Рп) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%

$$П_р = 0{,}01 \cdot Р_п \cdot С_п, \quad (7.22)$$

де $Р_с$ – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 50 \cdot 1571 = 786 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн.
1	2	3
1. Основна зарплата виконавців	$З_о$	842
2. Додаткова зарплата виконавців	$З_д$	84
3. Відрахування на соціальні потреби	$С_{оц}$	203
4. Загальногосподарські витрати	$Г_{осп}$	126
5. Витрати на матеріали	$З_М$	69
6. Освоєння нових операційних систем, мов програмування	$О_п$	126
7. Амортизація основних фондів	$А_м$	178
8. Повна собівартість програмного забезпечення	$С_п$	1571
9. Плановий прибуток	$П_р$	786
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	2357
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	472
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	2829

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об'єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн	
	Базовий	Новий
Вартість програмної продукції	–	2829
Всього капітальних витрат	–	2829

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	29524	16104
2. Витрати на електроенергію	$З_{ел}$	0	0
3. Витрати на амортизацію	$З_{ам}$	0	1415
Всього витрат за рік	$І$	29524	17509

Витрати на технічне обслуговування:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_c), \qquad (7.23)$$

де $Т_р$ – кількість годин обслуговування системи за рік, *год.*,

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 220 годин на рік до 120 годин на рік, тому витрати на технічне обслуговування зменшилися з

$$З_{р\,баз} = 220 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 29524\ грн.$$

до

$$З_{р\,нов} = 120 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 16104\ грн.$$

Витрати на електроенергію визначаються з урахуванням спожитої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$).

$$З_{ел}=П_{ел} \cdot Т_р \cdot Ц_{ел}. \quad (7.24)$$

Зел баз= Зел нов

Витрати на електроенергію не змінюються.

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн., за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	50	–	2829	–	1415
Всього відрахувань	-	–	2829	–	1415

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою

$$E_в = (Ц_п - С_п) \cdot N_е - \sum_{i=1}^{m} E_{р_m} \cdot К_{р_m}, \quad (7.25)$$

де: $К_р$ – балансова вартість основних фондів розробника, грн.; $Е_р$ – розрахунковий коефіцієнт капіталовкладень.

$$Е_в = (2357 - 1571) * 160 - (0.05 * 1408000 + 0,5 * 98594 + \\ +0,2 * 33190 + 0,1 * 160000) * 2/12 = 102038$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$Т_в = \frac{К_р^*}{(Ц_п - С_п) \cdot N_е}, \quad (7.26)$$

де: $К_р^*$ – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$Т_в = \frac{1699784}{(2357 - 1571) * 160 * 12/2} = 2,25 \text{ років}$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$Е_{сп} = (І_б - І_н) - Е_н(К_н - К_б), \quad (7.27)$$

де $І_б$, $І_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно, $К_б$, $К_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$Е_{сп} = (29524\text{-}17509)\text{-}0,5 \cdot 2829 = 10600 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат

$$Т_{сп} = \frac{К_н - К_б}{І_б - І_н} \quad (7.28)$$

$$Т_в = \frac{2829}{29524 - 17509} = 0,3 \text{ років}$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	160
2. Повна собівартість розробленої програми	Грн.	1571
3. Ціна розробленої програми	Грн.	2357
4. Плановий прибуток від реалізації розробленої програми	Грн.	768
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1699784
7. Загальний прибуток від реалізації програмної продукції	Грн.	122880
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	102038
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	2,25
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	2829
11.Величина економічного ефекту у користувача програмної продукції	Грн.	10600
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,3

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості приймаючих управлінських рішень.

8 ЗАХОДИ ПО ОХОРОНІ ПРАЦІ І ТЕХНІЦІ БЕЗПЕКИ

8.1 Вступ

Охорона праці — система збереження життя і здоров'я працівників у процесі трудової діяльності, що включає правові, соціально-економічні, організаційні, технічні, санітарно-гігієнічні, лікувально-профілактичні, реабілітаційні та інші заходи.

Загальні положення державної політики, щодо галузі охорони праці зазначені у Законі України “Про охорону праці”. Цей Закон визначає основні положення щодо реалізації конституційного права працівників на охорону їх життя і здоров'я у процесі трудової діяльності, на належні, безпечні і здорові умови праці, регулює за участю відповідних органів державної влади відносини між роботодавцем і працівником з питань безпеки, гігієни праці та виробничого середовища і встановлює єдиний порядок організації охорони праці в Україні [20]. Законодавство про працю містить норми і вимоги з техніки безпеки і виробничої санітарії, норми, що регулюють робочий час і час відпочинку, звільнення та переведення на іншу роботу, норми праці щодо жінок, молоді, гігієнічні норми і правила тощо. Загальний нагляд за додержанням норм охорони праці покладено на прокуратуру, спеціальний — на професійні спілки. Контроль за безпекою праці здійснюють також, державні й відомчі спеціалізовані інспекції (Держгіртехнагляд, Держенергонагляд, тощо). Науково-технічний прогрес вніс серйозні зміни в умови виробничої діяльності робітників розумової діяльності. Їх праця стала більш інтенсивною, напруженою і вимагає значних витрат розумової, емоційної і фізичної енергії. Це призвело до необхідності у знаходженні комплексного рішення проблем ергономіки, гігієни і організації праці, регламентації режимів праці та відпочинку. Охорона здоров'я робітників, забезпечення безпеки умов праці, ліквідація та профілактика професійних

захворювань і виробничого травматизму складає одну з головних турбот людського суспільства.

8.2 Аналіз умов праці на робочому місці ІТ-фахівця

На робочому місці ІТ-фахівця (або програміста) виникають небезпечні та шкідливі для безпечної життєдіяльності фактори:

- підвищений рівень шуму;
- несприятливі мікрокліматичні умови;
- недостатній рівень освітленості;
- шкідливі речовини;
- підвищений рівень електромагнітних випромінювань радіочастот;
- висока напруга електричної мережі;
- статична електрика та інші.

Робота програміста супроводжується також підвищеним ступенем напруженості трудового процесу. При систематичному впливі виробничих факторів, які не відповідають нормативним показникам, зростає рівень професійно зумовленої захворюваності працюючих та можуть виникнути професійні захворювання органів зору, руху, нервової системи. Таким чином, вивчення умов праці на робочому місці програміста є необхідною умовою запобігання негативних наслідків впливу небезпечних та шкідливих факторів. Робоче місце, добре пристосоване до трудової діяльності інженера, правильно і доцільно організоване, щодо простору, форми, розміру забезпечує йому зручне положення при роботі і високу продуктивність праці при найменшому фізичному і психічному напруженні.

Нормування параметрів проводиться в залежності від періоду року та категорії важкості виконуваних робіт. Для постійних робочих місць, якими є робочі місця ІТ-фахівців, встановлені оптимальні параметри мікроклімату, а за неможливості їх дотримання використовують допустимі параметри. Робота ІТ-

фахівця за важкістю відноситься до Іа (роботи, що виконуються сидячи і не потребують фізичного напруження) та Іб (роботи, що виконуються сидячи, стоячи або пов'язані з ходінням та супроводжуються деяким фізичним напруженням) категорій. В таблиці 8.1. наведені оптимальні параметри мікроклімату в приміщеннях.

Таблиця 8.1 – Параметри мікроклімату для приміщень з ПК

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	22...24°С; 40... 60%; до 0,1 м/с
Теплий	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	23...25 °С 40...60% 0,1...0,2 м/с

Виміряні за допомогою приладів температура та вологість у приміщеннях праці ІТ-фахівців повинні відповідати зазначеним у таблиці для теплого періоду року. Слід зазначити, що для нормалізації параметрів мікроклімату слід використовувати у приміщеннях кондиціювання повітря, або забезпечити подачу свіжого повітря системами вентиляції. Норми подачі свіжого повітря наведені у таблиці 8.2.

Таблиця 8.2 – Норми подачі свіжого повітря в приміщення

Характеристика приміщення	Об'ємна витрата свіжого повітря, що подається в приміщення, $м^3$ на одну людину в годину
Об'єм до 20 $м^3$ на людину 20... 40 $м^3$ на людину Більше 40 $м^3$на людину	Не менше 30 Не менше 20 Може біти використана природна вентиляція

Створення сприятливих умов праці і правильне естетичне оформлення робочих місць на виробництві має велике значення як для полегшення праці, так і для підвищення його привабливості, позитивно впливає на продуктивність праці. Забарвлення приміщень і меблів повинні сприяти створенню сприятливих умов для зорового сприйняття, гарного настрою. У службових приміщеннях, у яких виконується одноманітна розумова робота, що вимагає значної нервової напруги і великого зосередження, забарвлення повинно бути спокійних тонів – малонасичені відтінки холодного зеленого або блакитного кольорів.

При розробці оптимальних умов праці програміста необхідно враховувати освітленість. Раціональне освітлення робочого місця є одним з найважливіших факторів, що впливають на ефективність трудової діяльності людини, що попереджають травматизм і професійні захворювання. Правильно організоване освітлення створює сприятливі умови праці, підвищує працездатність і продуктивність праці. Освітлення на робочому місці програміста повинно бути таким, щоб працівник міг без напруги зору виконувати свою роботу. Стомлюваність органів зору залежить від ряду причин: недостатність освітленості; надмірна освітленість; неправильний напрям світла. Недостатність освітлення приводить до напруги зору, ослабляє увагу, приводить до настання передчасної стомленості. Надмірно яскраве освітлення викликає засліплення, роздратування і різь в очах. Неправильний напрямок світла на робочому місці може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Всі ці причини можуть призвести до нещасного випадку або профзахворювань. [21]

8.3 Пропозиції щодо підвищення працездатності ІТ-фахівця

Практичне значення заходів щодо підвищення працездатності випливає із закономірностей її динаміки і зводиться ось до чого:

збільшення фази стійкого стану у фонді робочого часу;

прискорення процесу впрацювання;

віддалення фази розвитку втоми;

забезпечення високої продуктивності праці за нормальних фізіологічних затрат.

Комплекс заходів щодо підвищення і збереження працездатності працівників на оптимальному рівні реалізується на техніко-організаційному, соціально-економічному, санітарно-гігієнічному, медико-біологічному, психологічному напрямках.

Вагомим фактором високої працездатності і продуктивності праці є оптимізація трудових навантажень на основі механізації і автоматизації виробничих процесів, удосконалення технології, скорочення і ліквідації важкої ручної праці. Доведено, що при правильній організації праці на легких роботах спостерігається найбільша тривалість фази стійкого стану, а на важких роботах вона нетривала.

Високий рівень працездатності безпосередньо залежить від умов праці, оскільки поліпшення їх супроводжується зменшенням енергетичних затрат організму на подолання несприятливого впливу факторів виробничого середовища.

Важливим напрямком підвищення працездатності працюючих є ритмізація трудових процесів, оптимізація темпу роботи, а також раціоналізація трудових рухів на фізіологічній основі, що сприяє формуванню і закріпленню робочих динамічних стереотипів, а отже зменшенню м’язових і вольових зусиль. Ритмічна робота підвищує функціональні можливості організму, сприяє його тренованості і забезпечує економізацію енергетичних затрат. [22]

Багатьом програмістам постійно доводиться працювати з великою кількістю програм одночасно. Часте перемикання туди-сюди між IDE та довідкою суттєво зменшує продуктивність фахівця. Однак вирішення цієї проблеми досить просте та очевидне: встановлення більшої кількості моніторів.

Оптимальним варіантом є два монітори. Все ж таки це найпростіший з апаратної точки зору варіант. Крім того, якби їх було більше, то ними було б

важче керувати, та й столі просто не вистачить місця на ще один монітор. Але тут ще залежить розміру моніторів. Є системи із 4 або 6 відносно невеликими екранами, які кріпляться на кронштейні. Але оптимальнім є два 27-дюймові монітори, на яких все добре видно, особливо коли працювати доводиться в основному з текстом. [23]

8.4 Пожежна безпека

Вимоги до пожежної безпеки на підприємстві неухильно повинен дотримуватися кожен співробітник, а організаційна складова при цьому покладається на посадових осіб за відповідним рішенням керівництва і прописується в посадових інструкціях і положеннях по структурним підрозділам.

Зокрема, вказуються конкретні території, ділянки, зони, об’єкти, цілі будівлі і їх частини, поверхи, на яких відповідального співробітника повинне проводити такі організаційні роботи.

Відповідальні особи зобов’язуються розробити, впровадити та підтримувати в певному інструкцією і положенням на ввірених їм об’єктах протипожежний режим і інструкції відповідно до вимог, викладених в нормативних актах.

Передбачено також створення підрозділу добровільної пожежної охорони та пожежно-рятувальної команди в його складі.

Встановлений режим включає порядки з описом місць спеціального призначення та правила їх користування та утримання, наприклад:

- евакуаційних шляхів;
- так званих «курилок»;
- місць складування продукції та сировини;
- стоянки транспорту.

Також встановлюється порядок роботи та технічного обслуговування:

- вентиляційного устаткування;

- засобів пожежогасіння і захисту від загорянь;

- нагрівальних приладів;

- електрообладнання.

Розробляються і впроваджуються правила роботи з відкритим вогнем і горючими матеріалами. Створюються графіки проходження інструктажів з пожежної безпеки співробітників, а також порядок і терміни перевірок знань пожежно-технічного мінімуму, в тому числі, тих працівників, які відповідальні за цю ділянку роботи на підприємстві. При цьому можуть передбачатися внутрішні лекції, семінари, тренінги та практичні заняття на підприємстві, а також зовнішні – на базі спеціалізованих навчальних центрів з професійними викладачами.

Важливою складовою протипожежного режиму на будь-якому об’єкті є розробка і впровадження порядку дій при виникненні пожежі. Неодмінно має бути план евакуації, описано, як повинні відключатися електроустановки, що і в якій послідовності необхідно робити співробітникам.

Відповідно, для кожного об’єкта, кожного приміщення (крім коридорів, санвузлів, басейнів і подібних приміщень), окремих видів робіт складаються інструкції, за якими повинен працювати персонал, залучений на певних ділянках і в виконанні окремих видів робіт. За інструкціями проводиться навчання (інструктаж) персоналу з подальшим контролем знань.

Детально про те, як розробити протипожежний режим, прописати порядки та інструкції, пояснюють на тематичних курсах і семінарах. [24]

8.5 Розрахункова частина

Розрахувати занулення глухозаземленої нейтралі трансформатора виробничого приміщення в якому працюють ІТ-фахівці.

Початкові дані:

− довжина магістрального кабеля − 85, м.

− довжина розгалуження − 20, м.

− номінальна потужність електроприймачів: 10 кВт.

− потужність освітлювальних приладів: 3 кВт.

Визначаємо силу номінального струму електроустановки.

$$I = \frac{P}{\sqrt{3} * U_л * \cos\phi}, \quad (8.1)$$

де Uл – лінійна напруга, В. (380В); cosф – коефіцієнт потужності, приймається в залежності від типу електрообладнання в межах 0,8..0,87.

$$I = \frac{10 * 1000}{\sqrt{3} * 380 * 0,85} = 17,87\ A$$

Визначаємо силу пускового струму електродвигуна

$$I_{пус} = 5 * I, \quad (8.2)$$

$$I_{пус} = 5 * 17,87 = 89,37\ A$$

Визначаємо номінальну силу струму апарата захисту

$$I_н = \frac{I_{пус}}{\beta}, \quad (8.3)$$

де β − коефіцієнт пуску електродвигуна, приймається: 2,5..3.

$$I = \frac{89,37}{2,6} = 34,37\ A$$

Визначаємо найменше допустиме по умовам спрацьовування захисту значення сили струму короткого замикання.

$$I_{кн} = I_н * K, \quad (8.4)$$

де К − коефіцієнт надійності; значення коефіцієнта К приймається: К = 1,25 (при Iн < 100А, К = 1,4);

$$I_{кн} = 34,37 * 1,4 = 48,12\ A$$

Знаходимо площу переріза провода або кабеля розгалуження із умови допустимого нагрівання

$$I_{доп} \geq I_{max}, \quad (8.5)$$

де Ідоп − тривало допутимий із умови нагрівання струм навантаження провідника, А.

$$I_{доп} \geq 34{,}37\ А$$

Було вибрано запобіжник типу ПН 2-100 з плавкою вставкою Іном =60 А.

$$I_{доп} \geq \frac{I_{вст}}{\alpha}, \quad (8.6)$$

де α – коефіцієнт відповідності, який залежить від умов прокладання і нагляду за мережею і приймається 3.

$$I_{доп} \geq \frac{60}{3} = 20\ А$$

Вибираємо площу перерізу 10 мм2 (Sф) при числі проводів і = 4, кабель АПРТО розташований у повітрі.

Визначаємо максимальний робочий струм

$$I_{роб} = I_{max} = К_о * \Sigma_1^n (К_з * I_н) \quad (8.7)$$

де Іном – номінальний струм кожного електроприймача, А.; Ко – коєфіцієнт одночасності роботи групи електроприймачів (0,7..0,8); Кз – коєфіцієнт завантаження (0,8..0,9).

$$I_{роб} = I_{max} = 0{,}75 * \left(0{,}85 * 17{,}87 + 0{,}85 * \frac{3 * 1000}{\sqrt{3} * 380 * 0{,}85}\right) = 156{,}451\ А$$

Визначаємо струм короткочасного перевантаження магістрального кабеля.

$$I_{пер} = К_о * \Sigma_1^{n-1} (К_з * I_н) + I_{пус} \quad (8.8)$$

$$I_{пер} = 0{,}75 * \left(0{,}85 * 17{,}87 + 0{,}85 * \frac{3 * 1000}{\sqrt{3} * 380 * 0{,}85}\right) + 89{,}37 = 230{,}63\ А$$

Визначаємо струм спрацювання теплового або електромагнітного розчеплювалювача автоматичного вимикача.

$$I_{спр} \geq I_{max} \quad (8.9)$$

$$I_{спр} \geq 156{,}451\ А$$

Струм спрацювання електромагнітного розчеплювача додатково перевіряємо по максимальному струму перевантаження лінії.

$$I_{спр} \geq 1.25 * I_{пер} \quad (8.10)$$

$$I_{спр} \geq 288{,}28\ А$$

Приймаємо Іспр = 400 А. Вимикач: А3734Б.

Вибираємо площу перерізу Sф магістрального кабеля (провідника) 70 мм, − кабель АПРТО розташований у повітрі, і=4 (число проводів).

Визначаємо потужність трансформатора.

$$N_{\text{ТР}} = \frac{\text{К}_\text{п} * \Sigma_1^n \text{Р}_\text{дном}}{\cos\varphi} \quad (8.11)$$

де Рд.ном − номінальна потужність електроприладу або іншого електроприймальника, кВт.; cosф − середній коефіцієнт потужності електроприймальника (0,9); Kn − коефіцієнт пуску (0,7).

$$N_{\text{ТР}} = \frac{0{,}7 * 103}{0{,}8} = 90{,}1 \text{ кВ} * \text{А}$$

Вибираємо трансформатор на 100 кВА (Zт =0,226 Ом).

Визначаємо площу перерізу нульового захисного провідника із умов:

$$I_{min} \geq K * I_\text{н} \quad (8.12)$$

Для магістрального кабеля Sn1 > 0,5*Sф=0,5*70=35 мм2. Для розгалуження Sn2 > 0,5*6=5 мм2. Округляємо ці значення до найближчих більших 35 мм2 (Sn1), і 5 мм2 (Sn2).

Визначення активного опору. Провідники із кольорового металу:

$$R_\text{ф} = \rho \frac{L_\text{М}}{S_{\text{ф}1}} + \rho \frac{L}{S_{\text{ф}2}} \quad (8.13)$$

$$R_n = \rho \frac{L_\text{М}}{S_{n1}} + \rho \frac{L}{S_{n2}} \quad (8.14)$$

де р − питомий опір матеріалу провідника, Ом*мм2/м., для міді 0,0175; для алюмінія 0,028; Lм та L − довжина дільниць (мігістрального кабеля та розгалуження);S − площа перерізу фазного (Sф) і нульового (Sn) провідників.

$$R_\text{ф} = 0{,}0175\frac{85}{70} + 0{,}0175\frac{20}{6} = 0{,}08 \text{ Ом}$$

$$R_n = 0{,}0175\frac{85}{35} + 0{,}0175\frac{20}{5} = 0{,}11 \text{ Ом}$$

Знаходимо дійсне значення (модуль) струма однофазного короткого замикання.

$$I_{кр} = \frac{U_Ф}{\frac{Z_T}{3} + \sqrt{(R_п + R_ф)^2 + (X_п + X_ф + X_п^{`})^2}} \quad (8.15)$$

$$I_{кр} = \frac{220}{\frac{0,226}{3} + \sqrt{(0,08 + 0,11)^2}} = 859,3 \text{ А}$$

Визначення максимальної напруги на корпусі обладнання відносно землі при замиканні фази на корпус.

$$U_{kmax} = I_k * Z_H < U_{доп.д} \quad (8.16)$$

де $U_{дан.д.}$ – це допустима напруга, що нормується по ГОСТ 12.1.038-82. При часі дії більше 1 с. приймається 36 В.

Z_H– повний опір нульового провідника.

$$U_{kmax} = 859,3 * 0,11 = 94,5 \text{ В} > 36 \text{ В}$$

Умова не виконується. Необхіно збільшити перерізи S_{n1} та S_{n2} до $S_{ф1}$ та $S_{ф2}$ і зробити перерахунок.

$$R_п = 0,08 \text{ Ом}$$

$$I_{кр} = 241 \text{ А}$$

$$U_{kmax} = 241 * 0,08 = 19,8 \text{ В} < 36 \text{ В}$$

Відповідь: значення перерізу нульового захисного провідника рівні 70 мм^2 для магістрального кабеля та 6 мм^2 для розгалуження. Дійсне значення струма однофазного короткого замикання 241 А. В якості запобіжника було обрано запобіжник моделі ПН 2-100.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання кваліфікаційної магістерської роботи, призначено для системи кібербезпеки кластеризації та аналізу даних з веб-ресурсів.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– був проведений огляд існуючих систем кібербезпеки кластеризації та аналізу даних з веб-ресурсів;

– досліджена система кібербезпеки кластеризації та аналізу даних з веб-ресурсів;

– на основі отриманих результатів досліджень створена програмна реалізація системи кібербезпеки кластеризації та аналізу даних з веб-ресурсів.

Розроблені під час виконання кваліфікаційної магістерської роботи алгоритми дозволяють успішно вирішувати завдання кластеризації та аналізу даних з веб-ресурсів.

Розроблене програмне забезпечення побудоване за принципом модульної організації структури програм, що забезпечує незалежність окремих компонентів, легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

Програмного забезпечення розроблювалося за принципами парадигми функціонального програмування, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма створена на мові Python. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи кібербезпеки кластеризації та аналізу даних з веб-ресурсів. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його

розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows /7/8/10 та сімейства Linux.

Даються необхідні рекомендації з установки та експлуатації розробленого програмного забезпечення.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування в галузі дослідження даних.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Візуалізація даних [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Візуалізація_даних

2. Аналіз даних [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Аналіз_даних.

3. Інтелектуальний аналіз даних [Електронний ресурс] – Режим доступу до ресурсу https://wiki.loginom.ru/articles/data-analysis.html

4. Інформаційна безпеки [Електронний ресурс] – Режим доступу до ресурсу https://habr.com/ru/hub/infosecurity/

5. Комп'ютерна безпека [Електронний ресурс] – Режим доступу до ресурсу: http://www.nbuv.gov.ua/node/300.

6. CyberThymus [Електронний ресурс] – Режим доступу до ресурсу: https://www.ussc.ru/news/novosti/na-chto-sposoben-ii-v-sfere-ib/

7. Amazon Macie [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/ru/macie/

8. DefPloreX [Електронний ресурс] – Режим доступу до ресурсу: https://blog.trendmicro.com/trendlabs-security-intelligence/defplorex-machine-learning-toolkit-large-scale-ecrime-forensics/

9. Мови програмування для роботи з даними [Електронний ресурс] – Режим доступу до ресурсу: https://www.analyticsinsight.net/top-10-data-science-programming-languages-for-2020/

10. Python [Електронний ресурс] – Режим доступу до ресурсу: https://ru.wikiversity.org/wiki/ Python /.

11. NumPy [Електронний ресурс] – Режим доступу до ресурсу: https://numpy.org/

12. Pandas [Електронний ресурс] – Режим доступу до ресурсу: https://pandas.pydata.org/

13. Збір даних [Електронний ресурс] – Режим доступу до ресурсу https://ru.wikiversity.org/wiki/ Збір даних /.

14. CSE-CIC-IDS2017 [Електронний ресурс] – Режим доступу до ресурсу : https://www.unb.ca/cic/datasets/ids-2017.html

15. Випадковий лісовий [Електронний ресурс] – Режим доступу до ресурсу: https://uk.education-wiki.com/2890284-random-forest-algorithm

16. Найпопулярніші алгоритми машинного навчання [Електронний ресурс] – Режим доступу до ресурсу: https://www.analyticsvidhya.com/blog /2017/09/common-machine-learning-algorithms/

17. Ансамбль методів [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f

18. Дослідження набору даних CSE-CIC-IDS2017 [Електронний ресурс] – Режим доступу до ресурсу: https://www.gnu.org/philosophy/free-sw.uk.html

19. Вільна ліцензія [Електронний ресурс] – Режим доступу до ресурсу: https://www.researchgate.net/publication/328512658_Anomaly_Detection_in_Networks_Using_Machine_Learning

20. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII [Електронний ресурс] – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12

21. Робоче місце програміста [Електронний ресурс] – Режим доступу до ресурсу: https://cpo.stu.cn.ua/Oksana/dipl_bak/140.html

22. Покращення умов праці [Електронний ресурс] – Режим доступу до ресурсу: https://buklib.net/books/25142/

23. Підвищення продуктивності [Електронний ресурс] – Режим доступу до ресурсу: http://itnotesblog.ru/note.php?id=158

24. Пожежна безпека [Електронний ресурс] – Режим доступу до ресурсу: https://profiteh.ua/pozhezhna-bezpeka-na-pidpryiemstvi-pravyla-ta-orhanizatsiia/

25. Лутц М. Программирование на Python, том II, 4-е издание. – Пер. с англ. – СПб.: Символ-Плюс, 2011. – 992 с., ил.

26. Андреас Мюллер. Введение в машинное обучение с помощью Python. Руководство для специалистов по работе с данными. : Пер. с англ. — СПб. : 000 “Альфа-книга”.

27. Чио К., Фримэн Д. Машинное обучение и безопасность / пер. с анг. А. В. Снастина. – М.: ДМК Пресс, 2020. – 388 с.: ил.

28. Шалев-Шварц Ш., Бен-Давид Ш. Идеи машинного обучения: от теории к алгоритмам / пер. с англ. А. А. Слин- кина. – М.: ДМК Пресс, 2019. – 436 с.: ил.

29. Эриксон Д. Хакинг: искусство эксплойта. 2-е изд. - СПб.: Питер, 2018. - 496 с.: ил. - (Серия «Библиотека программиста»).

30. Таненбаум Э., Уэзеролл Д. Компьютерные сети. 5-е изд. — СПб.: Питер, 2012. — 960 с.: ил.

31. Дуда Р., Харт П. Распознавание образов и анализ сцен. – М.: Мир, 1976.

32.Breiman, L., Friedman, J., Olshen, R. and Stone, C. Classification and Regression Trees - CRC Press 1984.

33. Quinlan, J. R. C4.5: Programs for Machine Learning - Morgan Kaufmann 1993.

34. Lbov, G. and Berikov, V. Recognition of a Dynamic Object and Prediction of Quantitative Characteristics in the Class of Logical Functions. // Pattern Recognition and Image Analysis. Vol 7, N 4, 1997, pp. 407-413.

35. Лбов Г.С., Бериков В.Б. Устойчивость решающих функций в задачах распознавания образов и анализа разнотипной информации. - Новосибирск: Изд-во Ин-та математики, 2005. - 218 с.

36. Боровиков, В.П. Популярное введение в современный анализ данных в системе STATISTICA: Учебное пособие для вузов / В.П. Боровиков. — М.: Гор. линия-Телеком, 2013. — 288 с.

37. Боровиков, В.П. Популярное введение в современный анализ данных в системе Statistica: Учебное пособие / В.П. Боровиков. — М.: ГЛТ, 2013. — 288 с.

38. Боровиков, В.П. Популярное введение в современный анализ данных в системе STATISTICA. Учебное пособие для вузов. +CD / В.П. Боровиков. — М.: РиС, 2015. — 288 с.

39.Воскобойников, Ю.Е. Регрессионный анализ данных в пакете MATHCAD + CD / Ю.Е. Воскобойников. — СПб.: Лань, 2011. — 224 с.

40. Горяинова, Е.Р. Прикладные методы анализа статистических данных: Учебное пособие / Е.Р. Горяинова, А.Р. Панков, Е.Н. Платонов. — М.: ИД ГУ ВШЭ, 2012. — 310 с.

41. Дайитбегов, Д.М. Компьютерные технологии анализа данных в эконометрике: Монография / Д.М. Дайитбегов. — М.: Вузовский учебник, НИЦ ИНФРА-М, 2013. — 587 с.

42. Кабаков, Р. R в действии. Анализ и визуализация данных в программе R / Р. Кабаков. — М.: ДМК, 2016. — 588 с.

43. Кацко, И.А. Практикум по анализу данных на компьютере / И.А. Кацко, Н.Б. Паклин. — М.: КолосС, 2009. — 278 с.

44. Козлов, А.Ю. Статистический анализ данных в MS Excel: Учебное пособие / А.Ю. Козлов, В.С. Мхитарян, В.Ф. Шишов. — М.: ИНФРА-М, 2013. — 320 с.

45. Корячко, В.П. Анализ и проектировапние маршрутов передачи данных в корпоративных сетях / В.П. Корячко, Д.А. Пепепелкин. — М.: ГЛТ, 2012. — 236 с.

46. Крих, С.Б. Советская историография древности в контексте мировой историографической мысли: Анализ текстов, созданных в советский период. Разбор «периферии научно / С.Б. Крих, О.В. Метель. — М.: Ленанд, 2014. — 256 с.

47. Крянев, А.В. Метрический анализ и обработка данных / А.В. Крянев, Г.В. Лукин, Д.К. Удумян. — М.: Физматлит, 2012. — 308 с.

48. Кулаичев, А.П. Методы и средства комплексного анализа данных: Учебное пособие / А.П. Кулаичев. — М.: Форум, НИЦ ИНФРА-М, 2013. — 512 с.

49. Лебедев, Ю.А. Характеристики углеводородов: Анализ численных данных и их рекомендованные значения. Справочное издание / Ю.А. Лебедев, А.Н. Кизин, Т.С. Папина, И. Сайфуллин. — М.: Ленанд, 2012. — 560 с.

50. Маккинли, У. Python и анализ данных / У. Маккинли. — М.: ДМК, 2015. — 482 с.

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРМ-123.21.0016.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Прокопов В.В.			Дослідження та програмна реалізація системи кібербезпеки кластеризації та аналізу даних з веб-ресурсів	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Мелешко Є.В.				М	
Н. Контр.		Гермак В.С.				ЦНТУ КІ-20М	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи кібербезпеки кластеризації та аналізу даних з веб-ресурсів.

2 Підстава для розробки

Підставою для розробки служить завдання на магістерську роботу, видане на кафедрі програмування та захисту інформації (нак. №42-13 від 02.08.2021 року).

3 Мета та призначення розробки

Метою магістерської роботи є дослідження та програмна реалізація системи кібербезпеки кластеризації та аналізу даних з веб-ресурсів.

4 Джерела розробки

Джерелом цієї магістерської роботи є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи кібербезпеки кластеризації та аналізу даних з веб-ресурсів;

– цілісність даних у процесі роботи та при зберіганні;

– простоту у використанні.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ архітектури IBM PC, працювати в ОС Windows XP/Vista/7/8/10 і з сумісними з цією платформою пристроями і прикладним програмним забезпечення.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows XP/Vista/7/8/10.

5.8.1 Обладнання

Комп'ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14" 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище Python.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2021 року.

8 Вимоги щодо охорони праці

В частині охорони праці магістерської роботи повинна бути розглянута стан умов праці та поліпшення працездатності ІТ-фахівця.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 97 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі магістерської роботи. Постановка задачі на виконання магістерської роботи (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень магістерської роботи.

10.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання магістерської роботи на попередній захист 04.12.2021 р.

11.2 Подання магістерської роботи на захист 20.12.2021 р.

Додаток Б

(обов'язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ
Керівник випускної кваліфікаційної роботи
за другим (магістерським) рівнем вищої освіти
_____ Є.В. Мелешко

Дослідження та програмна реалізація системи кібербезпеки кластеризації та аналізу даних з веб-ресурсів

Лістинг програми

Код документу 12

Носій: DVD-диск

Загальна кількість аркушів: 20

Літера: РП

Кропивницький – 2021 року

Основна програма

```
import pandas as pd
import numpy as np
import dask.dataframe as ds
import matplotlib.pyplot as plt
from utils import ( csv_data, csv_data_dir, data_labels,
                    processed_data_dir, all_data_files_name,
                    attacks_data_file_name )
from sklearn.preprocessing import LabelEncoder
import os


plots_dir_name = "plot"
stats_dir_name = "stats"
stats_path = plots_dir_name + "\\" + stats_dir_name

os.makedirs(processed_data_dir, exist_ok=True)
os.makedirs(plots_dir_name, exist_ok=True)
os.makedirs(stats_path, exist_ok=True)

"""
df = pd.read_csv(csv_data_dir + csv_data[0], sep=",")

print(df.info())
print(df.shape)
"""

df_list = []



def process_data(files_list):

      """
      Вагомими кроками є функція read_csv, яка приймає строкове значення шляху
      до файлу .csv формату і повертає його перетвореним у формат DataFrame.
       Функція process_data і відповідає за попередню обробку даних. Вона
      приймає
       список, який включає імена файлів і далі обходить його, перетворюючи
      кожний
        файл один за одним та проводить перевірку значень атрибутів.
      """

    for file_name in files_list:
        print("Currently processing file",file_name)

        path_to_file = csv_data_dir + file_name

        df = pd.read_csv(
            path_to_file,
            sep=',',
            engine='python',
            encoding="cp1250"
        )
        df.columns = df.columns.str.strip()
        df=df.fillna(0)

        df.replace("inf", 0, inplace=True)
        df.replace("Infinity", -1, inplace=True)
        df.replace("NaN", 0, inplace=True)

        df.replace(" - "," - ", inplace=True)

        nump_nan_val_list = [np.inf, -np.inf, np.nan]
```

```
        df.replace(nump_nan_val_list, -1, inplace=True)

        mask = ["Flow Bytes/s", "Flow Packets/s"]
        df[mask] = df[mask].apply(pd.to_numeric)

        object_features = list(df.select_dtypes(include=['object']).columns)
        object_features.remove('Label')

        label_encoder = LabelEncoder()
        df[object_features] = df[object_features].apply(lambda feature:
label_encoder.fit_transform(feature.astype(str)))

        df=df.drop(data_labels[61], axis=1)

        zero_mask = (df.Label != 0) & (df.Label != "0")
        df = df[zero_mask]

        df.to_csv(processed_data_dir + "\\" + file_name, index=False)

        df_list.append(df)
        print("Processing of file ",file_name, "completed")


def combine_files(df_list):
    all_df = pd.concat(df_list, axis=0, ignore_index=True)
    print(all_df['Label'].unique())

    all_df["Label"].replace({
        "Web Attack - Sql Injection": "WA SI",
        "Web Attack - XSS": "WA XSS",
        "Web Attack - Brute Force": "WA BF",
    }, inplace=True)

    print(all_df['Label'].unique())
    attacks_only_mask = all_df['Label'] != 'BENIGN'
    attacks_only_data = all_df[attacks_only_mask]


    all_df.to_csv(processed_data_dir + "\\" + all_data_files_name + '.csv',
                  index=False
    )
    attacks_only_data.to_csv(
        processed_data_dir + "\\" + attacks_data_file_name + '.csv',
        index=False
    )
    print(attacks_only_data['Label'].unique())
    print("Files combined!")


def plot_data_barh(x, y, name, show=False):

    plt.figure()

    plt.rcParams['figure.figsize'] = (10, 5)
    plt.title(name)
    plt.barh(x,y, align="edge")

    plt.xlabel('Quantity')
    plt.ylabel('Type')

    plt.savefig(
        stats_path + "\\" + name + ".png",
        orientation = 'portrait',
        format = 'png',
        facecolor=(.94, .94, .94)
    )

    if show:
        plt.show()
```

```
def plot_pie(x, y, name, show=False, explode_list=None):

    list_size = len(x)
    list_sum = x.sum()

    if not explode_list:
        explode_list = [0 for _ in range(list_size)]

    prop_rate = [round(i/list_sum, 5) for i in x ]

    legend_labels = [f"{label} ({prop})" for label, prop in zip(y, prop_rate)]
    plt.figure(figsize=(10,10))

    plt.title(name)
    plt.pie(x, explode=explode_list)
    plt.legend(labels=legend_labels, loc="upper left", fontsize=20)
    plt.savefig(stats_path + "\\" + name + ".png",
        orientation = 'portrait',
        format = 'png',
        facecolor=(.94, .94, .94)
    )
    if show:
        plt.show()



def get_data_from_file(file_name):

      """
      Після завершення операцій форматування та класифікацій відбувається
      статична
      обробка новостворених файлів, яка необхідно для того, щоб мати точне
      представлення про види та кількість полів, що відносяться до тієї чи
       іншої сутності.
      """

    path_to_file = processed_data_dir + "\\" + file_name

    df = ds.read_csv(path_to_file, sep=',',
                     engine='python',
                     usecols=['Label'])

    #zero_mask = (df.Label != 0) & (df.Label != "0")
    #df = df[zero_mask]
    df = df.compute()
    attack_mask = df['Label'] != 'BENIGN'

    #print(df.info())
    #print(df["Label"].value_counts())
    df_attack = df[attack_mask]

    data_series = df.iloc[:, 0].value_counts()

    all_data_stats =  pd.DataFrame({
        'Label': data_series.index,
        'Quantity': data_series.values
    })

    df_attack_series = df_attack.iloc[:, 0].value_counts()

    all_attack_stats =  pd.DataFrame({
        'Label': df_attack_series.index,
        'Quantity': df_attack_series.values
    })

    all_data_stats_sorted = all_data_stats.sort_values(
       by='Quantity', ascending=True
```

```
    )

    all_attack_stats_sorted = all_attack_stats.sort_values(
        by='Quantity', ascending=True
    )


    #print(all_data_stats)
    #print(all_attack_stats)

    all_data_stats.to_csv(stats_path + f"stats_{file_name}", index=False)

    all_attack_stats.to_csv(stats_path + f"attack_stats_{file_name}",
index=False)

    plot_data_barh(all_data_stats_sorted['Label'],
                   all_data_stats_sorted['Quantity'],
                   name='All data'
    )

    plot_data_barh(all_attack_stats_sorted['Label'],
                   all_attack_stats_sorted['Quantity'],
                   name='All attacks'
    )

    attack_mask_lt_10 = all_attack_stats['Quantity'] < 10000
    attack_lt_10 = all_attack_stats[attack_mask_lt_10]

    #print(attack_lt_10)

    explode_list=[0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.4, 0.4, 0.4]
    plot_pie(attack_lt_10['Quantity'],
             attack_lt_10['Label'],
             name='Attacks less than 10k',
             explode_list=explode_list
    )




if __name__ == "__main__":
    #process_data(csv_data)
    #combine_files(df_list)
    get_data_from_file(all_data_files_name + '.csv')


csv_data = [
    "Monday-WorkingHours.pcap_ISCX.csv",
    "Tuesday-WorkingHours.pcap_ISCX.csv",
    "Wednesday-workingHours.pcap_ISCX.csv",

    "Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv",
    "Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv",

    "Friday-WorkingHours-Morning.pcap_ISCX.csv",
    "Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv",
    "Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv"
]
data_labels=[
    "Flow ID","Source IP","Source Port","Destination IP",
    "Destination Port","Protocol","Timestamp","Flow Duration",
    "Total Fwd Packets", "Total Backward Packets","Total Length of Fwd Packets",
    "Total Length of Bwd Packets","Fwd Packet Length Max",
    "Fwd Packet Length Min","Fwd Packet Length Mean","Fwd Packet Length Std",
    "Bwd Packet Length Max","Bwd Packet Length Min","Bwd Packet Length Mean",
    "Bwd Packet Length Std", "Flow Bytes/s","Flow Packets/s","Flow IAT Mean",
    "Flow IAT Std","Flow IAT Max","Flow IAT Min","Fwd IAT Total","Fwd IAT Mean",
    "Fwd IAT Std","Fwd IAT Max", "Fwd IAT Min","Bwd IAT Total","Bwd IAT Mean",
    "Bwd IAT Std","Bwd IAT Max","Bwd IAT Min","Fwd PSH Flags","Bwd PSH Flags",
```

```
    "Fwd URG Flags","Bwd URG Flags","Fwd Header Length","Bwd Header Length",
    "Fwd Packets/s","Bwd Packets/s","Min Packet Length","Max Packet Length",
    "Packet Length Mean","Packet Length Std","Packet Length Variance",
    "FIN Flag Count","SYN Flag Count","RST Flag Count","PSH Flag Count",
    "ACK Flag Count","URG Flag Count","CWE Flag Count", "ECE Flag Count",
    "Down/Up Ratio","Average Packet Size","Avg Fwd Segment Size",
    "Avg Bwd Segment Size","Fwd Header Length.1","Fwd Avg Bytes/Bulk",
    "Fwd Avg Packets/Bulk","Fwd Avg Bulk Rate","Bwd Avg Bytes/Bulk",
    "Bwd Avg Packets/Bulk","Bwd Avg Bulk Rate","Subflow Fwd Packets",
    "Subflow Fwd Bytes", "Subflow Bwd Packets","Subflow Bwd Bytes",
    "Init_Win_bytes_forward","Init_Win_bytes_backward","act_data_pkt_fwd",
    "min_seg_size_forward","Active Mean","Active Std","Active Max",
    "Active Min","Idle Mean","Idle Std","Idle Max","Idle Min",
    "Label","External IP"
]

csv_data_dir = 'csv_files\\'

processed_data_dir = 'processed_data'
all_data_files_name = 'data'
attacks_data_file_name = "attaks"

attack_list = ['BENIGN', 'FTP-Patator', 'SSH-Patator', 'DoS slowloris',
               'DoS Slowhttptest', 'DoS Hulk', 'DoS GoldenEye',
               'Heartbleed', 'WA BF', 'WA XSS', 'WA SI',
               'Infiltration', 'Bot', 'PortScan', 'DDoS']

db_dir_name = "temp"
db_file_name = "mlDb"
      if method == 'PCA':
            transform_data = PCA(data=data,
                                           dims_rescaled_data = n_cmnts)

      elif method == 'MDS':
            mds    =    MDS(n_components=n_cmnts,    dissimilarity="precomputed",
random_state=1)
            transform_data = mds.fit_transform(data)
      return transform_data


def vectorizing_data(data_to_vctrz, n_components, stop_words=None, use_idf=True,
tokenizer=None, max_df=1.0, ngram_range=(1, 1), method='PCA'):

      vectorizer = TfidfVectorizer(stop_words=stop_words,
                                                 max_df=max_df,
                                  use_idf=use_idf,
                                  tokenizer=tokenizer,
                                  ngram_range=ngram_range)
      tfidf_data = vectorizer.fit_transform(data_to_vctrz)

      data_norm = normalize(tfidf_data)
      data_array = data_norm.toarray()

      transform_data = dim_method(method=method,
                                                 data=data_array,
                                                 n_cmnts=n_components)
      return tfidf_data, transform_data, vectorizer


def elbow_method(Y_sklearn, clust_num):
      num_clust = range(1,clust_num)

      kmean = list()
      score = list()
      for i in num_clust:
            kmnClstr = KMeans(n_clusters=i,
                                        init='k-means++',
                                        max_iter=100,
                                        algorithm = 'auto')
```

```
            kmean.append(kmnClstr)
      for i in range(len(kmean)):
            fitted = kmean[i].fit(Y_sklearn )
            scored = fitted.score(Y_sklearn )
            score.append(scored)

      return num_clust,score



def get_top_features_cluster(array, prediction, n_feats):
      """
      Повертає найбільш об'єкти кластерів, що найчастіше зустірчаються
      """
    labels = np.unique(prediction)

    dfs = []
    for label in labels:
        id_temp = np.where(prediction==label)
        x_means = np.mean(array[id_temp], axis = 0)
        sorted_means = np.argsort(x_means)[::-1][:n_feats]
        pprint( np.argsort(x_means))
        features = vectorizer.get_feature_names()
        best_features = [(features[i], x_means[i]) for i in sorted_means]
        df = pd.DataFrame(best_features, columns = ['features', 'score'])
        dfs.append(df)
    return dfs


def colorGen(lim):
      """
      Випадково генерує кольори для позначення кажного кластеру
      """
      alph = list(map(chr, range(ord('a'), ord('f')+1)))
      variation = alph + list(range(0,10))
      colors = dict()
      for i in range(0, lim):
            colors[i] = "#"
            for _ in range(0,6):
                  colors[i] += str(random.sample(variation, k=1)[0])
      return colors


path_db_file = db_dir_name + "\\" + db_file_name

import pandas as pd
import numpy as np
import os
from utils import ( processed_data_dir, all_data_files_name,
                    path_db_file, db_dir_name )

import dask.dataframe as dk
from sklearn.feature_selection import SelectFromModel as SFM
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

import shelve
mlDb = shelve.open("dataDb")

feature_importance_list = mlDb['ftr_imprt']



feature_importance_name = [name for name, _ in feature_importance_list]
#print(len(feature_importance_name))
df = pd.read_csv('balanced.csv', engine='python')

df['Label'] = df['Label'].apply(lambda x: 0 if x == 'BENIGN' else 1)
```

```
#print(df.info())
y = df['Label'].values
X = df.drop(columns=['Label'])
#print(X)
X = X[feature_importance_name]

print(X.shape, y.shape)

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)


ml_list={
"Naive Bayes":GaussianNB(),
"KNN":KNeighborsClassifier(n_neighbors=3),
"GBC" :GradientBoostingClassifier(),
"AdaBoost":AdaBoostClassifier(),
"DTC":DecisionTreeClassifier(max_depth=4,criterion="entropy"),
"SVC": SVC(C=10000)
}

scoring = ['accuracy','precision_macro', 'recall_macro', 'f1_macro']


"""
for ml in ml_list:
    ml_clf = ml_list[ml]
    score_cv = cross_validate(ml_clf, X_train, y_train,
                              scoring=scoring, cv=7)
    print(f"Name {ml}")
    for key in score_cv.keys():
        if 'time' not in key:
            score = round(score_cv[key].mean(), 3)
            print(f"{key}: {score}")

    print("-" * 60)
"""

from sklearn.model_selection import RandomizedSearchCV

gbc = GradientBoostingClassifier()
param_grid = [
{ 'n_estimators' : [ 50, 100, 150,200,250],
  'max_depth': [5, 10, 15,20,25],
  'learning_rate': [0.01,0.05,0.1,0.5,0.9,1.0],
  'max_features' : [ 10, 15, 20 ] } ,
]

rand_search = RandomizedSearchCV(gbc, param_grid,cv=7, refit='recall_macro',
                                 scoring=scoring,  return_train_score=True)
rand_search.fit(X, y)

print(rand_search.best_estimator_ )
print(rand_search.best_params_ )
print(rand_search.best_score_ )

"""

gbc = GradientBoostingClassifier()
gbc.fit(X_train, y_train)
predictor = gbc.predict(X_test)

accuracy_value = gbc.score(X_test, y_test)
accuracy_value_train = gbc.score(X_train, y_train)

f_1 = f1_score(y_test, predictor, average='macro')
precision_value = precision_score(y_test, predictor, average='macro')

recall_value = recall_score(y_test, predictor, average='macro')
```

```
print(f"Name GBC \n"
          f"Accuracy: {accuracy_value}\n"
          f"Accuracy (train): {accuracy_value_train}\n"
          f"F1 score: {f_1}\n"
          f"Precission: {precision_value}\n"
          f"Recall: {recall_value}\n"
    )
    """

os.makedirs(db_dir_name, exist_ok=True)
def analysis_data(dmn_method,settings, stop_words_ls=None):

      global stemmer, vocab_frame

      stop_words = ENGLISH_STOP_WORDS.union(stop_words_ls)
      stemmer = SnowballStemmer("english")

      dbCurs.execute("SELECT title, plot FROM movies")
      titles = list()
      texts = list()

      for title, text in dbCurs.fetchall():
            titles.append(title)
            texts.append(text)


      totalvocab_stemmed = []
      totalvocab_tokenized = []
      max_df, n_comp, lim_gram, use_idf, tokenize_and_stem = settings

      for i in texts:
          allwords_stemmed = tokenize_and_stem(i)
          totalvocab_stemmed.extend(allwords_stemmed)

          allwords_tokenized = tokenize_only(i)
          totalvocab_tokenized.extend(allwords_tokenized)

#mlDb = shelve.open(path_db_file)
path_to_data_file = processed_data_dir + "\\" + all_data_files_name
dataset = dk.read_csv(
    path_to_data_file + ".csv",
    engine='python'
    )

#dataset = dataset.compute()
#print(dataset["Label"].unique())
#print(dataset["Label"].value_counts())

def visualize_data(clusters, data, dmn_method, titles, n_cmnts):
      """
      Налаштуваня графіку для виведення на систему координат кластерів
      з назвою кжоного їх об'єкта
      """

      pos = dim_method(dmn_method, data, n_cmnts)

      xs, ys = pos[:, 0], pos[:, 1]

      cluster_colors = colorGen(5)

      cluster_names = {0: 'Family, home, war',
                       1: 'Story, murdering, times',
                       2: 'World,space, team',
                       3: 'Gangs, destroy, war',
                       4: 'Boy, friends, new'}

      df = pd.DataFrame(dict(x=xs, y=ys, label=clusters, title=titles))
```

```
      groups = df.groupby('label')

      return groups, df, cluster_names, cluster_colors


def data_labels_multi_class(data_labels):

    data_labels_multi = []
    for label in data_labels:
        if label == "BENIGN":
            data_labels_multi.append(0)
        elif label == "DoS Hulk":
            data_labels_multi.append(1)
        elif label == "DDoS":
            data_labels_multi.append(2)
        elif label == "DoS GoldenEye":
            data_labels_multi.append(3)
        elif label == "FTP-Patator":
            data_labels_multi.append(4)
        elif label == "SSH-Patator":
            data_labels_multi.append(5)
        elif label == "DoS slowloris":
            data_labels_multi.append(6)
        elif label == "DoS Slowhttptest":
            data_labels_multi.append(7)
        elif label == "WA BF":
            data_labels_multi.append(8)
        elif label == "WA XSS":
            data_labels_multi.append(9)
        elif label == "WA SI":
            data_labels_multi.append(10)
        elif label == "Bot":
            data_labels_multi.append(11)
        elif label == "Infiltration":
            data_labels_multi.append(12)
        elif label == "Heartbleed":
            data_labels_multi.append(13)
        elif label == "PortScan":
            data_labels_multi.append(14)

    return data_labels_multi


def data_labels_bin_class(data_labels):

    data_labels_bin = []
    for label in data_labels:
        if label == "BENIGN":
            data_labels_bin.append(0)
        else:
            data_labels_bin.append(1)

    return data_labels_bin

data_labels_bin_class_list = data_labels_bin_class(dataset["Label"])
data_labels_bin_class_arr = np.asarray(
    data_labels_bin_class_list,
    dtype="int8"
)

feature_to_delete = ['Label','Flow ID', "External IP", "Timestamp", "Destination
IP", 'Destination Port']

dataset = dataset.drop(columns=feature_to_delete)

dataset = dataset.astype(np.float32)
dataset = dataset.compute()
m = 6168
L = []
```

```
l = 0

while l <=m:
      L.append(True)
      l+=1
i = 2
while i < m:
      if L[i] is True:
            j = i**2
            while j < m:
                  L[j] = False
                  j+=i
      i+=1
P = []
for i in range(2,m):
      if L[i] is True:
            P.append(i)
print("Prime number: ", P)


G = [] # зведені лишки
for i in range(1,m):
      if gcd(m,i)==1:
            G.append(i)
print('G: ', G)


PrR= []
E = len(G)
N = True
if m%2==0:
      M=m/2
      if M%2!=0:
            for i in P:
                  while M%i==0:
                        t = M/i
                        deg = 1
                        if t%i!=0:
                              print( "%s = 2*%s^%s" % (m,M,deg))
                              m = M
                              break
                        else:
                              while t%i==0:
                                    t=t/i
                                    deg+=1
                        print( "%s = 2*%s^%s" % (m,i,deg))
                        m = i
                        break

      else:
            N = False
            print(m,' парне число')
else:
      for i in P:
            if m%i==0:
                  t = m/i
                  deg = 1
                  if t%i!=0:
                        N = False
                        break
                  else:
                        while t%i==0:
                              t=t/i
                              deg+=1

                  m = i

if N is False:
```

```
else:
      G = [] # зведені лишки
      for i in range(1,int(m)):
            if gcd(m,i)==1:
                  G.append(i)


      for i in G:
            I = i
            j=2
            while j<m:
                  if (I**j)%m==1 and j < (m-1):

                        break
                  elif (I**j)%m==1 and j == (m-1) :

                        PrR.append(i)
                        break
                  j +=1

while N:

      while True:
            r = int(input('Ведіть число з проміжку %s: ' % PrR))
            if r not in PrR:

            else:
                  break


      Lg = []
      Lm = list(range(2,int(m)))
      for i in Lm:
            g = r ** i
            if g > r:
                  g = g %m
                  Lg.append(g)
            else:
                  Lg.append(g)
      Nr = True
      while Lg[0] in Lg[1:]:

            Nr = False
            break
      if Nr == True:
            print('Таблиця індексів:', Lg)
      break

input()
X_data = dataset.values
X_data = np.nan_to_num(df.values)
print(np.any(np.isnan(X_data)),
      np.all(np.isfinite(X_data))
    )
#mlDb['X_data'] = X_data

#mlDb['data_labels'] = data_labels_multi
#mlDb.close()

print(X_data.shape, data_labels_bin_class_arr.shape)

## мета цього коду — визначити, які функції використовувати на етапі машинного
навчання.
## для цієї мети обчислюються ваги важливості атак.
## цей розрахунок було зроблено за допомогою RandomForestRegressor.
class RandomForest:

    '''
     Клас, який реалізує алгоритм Random Forest з нуля.
```

```
     '''
def __init__(self, num_trees=25, min_samples_split=2, max_depth=5):
        self.num_trees = num_trees
        self.min_samples_split = min_samples_split
        self.max_depth = max_depth

        self.decision_trees = []

    @staticmethod
    def _sample(X, y):

            '''
         Допоміжна функція, яка використовується для відбору проби.

         :param X: np.array, особливості
         :param y: np.array, target
         :return: кортеж (зразок функцій, зразок цілі)
         '''         n_rows, n_cols = X.shape
               samples = np.random.choice(a=n_rows, size=n_rows, replace=True)
        return X[samples], y[samples]

    def fit(self, X, y):
        '''
         Навчає класифікатор випадкових лісів.

         :param X: np.array, особливості
         :param y: np.array, target
         :return: Жодного
         '''         # Reset
        if len(self.decision_trees) > 0:
            self.decision_trees = []

                num_built = 0
        while num_built < self.num_trees:
            try:
                clf = DecisionTree(
                    min_samples_split=self.min_samples_split,
                    max_depth=self.max_depth
                )

                _X, _y = self._sample(X, y)

                clf.fit(_X, _y)

                self.decision_trees.append(clf)
                num_built += 1
            except Exception as e:
                continue

    def predict(self, X):
        '''
        Прогнозує мітки класів для нових екземплярів даних.

         :param X: np.array, нові екземпляри для передбачення
         :повернення:
         '''         # Make predictions with every tree in the forest
        y = []
        for tree in self.decision_trees:
            y.append(tree.predict(X))

        y = np.swapaxes(a=y, axis1=0, axis2=1)

        predictions = []
        for preds in y:
            counter = Counter(x)
            predictions.append(counter.most_common(1)[0][0])
        return predictions

X_train, X_test, y_train, y_test = train_test_split(
```

```
    X_data,
    data_labels_bin_class_arr,
    random_state=0
)


select_feature = SFM(
    RFC(n_estimators=150, random_state=0),
    threshold="median"
)


import os, sys

currentdir = os.path.dirname(os.path.realpath(__file__))

parentdir = os.path.dirname(currentdir)
sys.path.append(parentdir)

import shelve
from utils import db_file_name


mlDb = shelve.open(db_file_name)

#X_data = mlDb['X_data']
def str_column_to_float(dataset, column):
      for row in dataset:
            row[column] = float(row[column].strip())

# Перетворити стовпець рядка на ціле число
def str_column_to_int(dataset, column):

      class_values = [row[column] for row in dataset]
      unique = set(class_values)
      lookup = dict()

      for i, value in enumerate(unique):
            lookup[value] = i
            print('[%s] => %d' % (value, i))

      for row in dataset:
            row[column] = lookup[row[column]]
      return lookup

# Знайдіть мінімальне та максимальне значення для кожного стовпця
def dataset_minmax(dataset):

      minmax = list()
      for i in range(len(dataset[0])):
            col_values = [row[i] for row in dataset]
            value_min = min(col_values)

            value_max = max(col_values)
            minmax.append([value_min, value_max])
      return minmax

# Rescale dataset columns to the range 0-1
def normalize_dataset(dataset, minmax):
      for row in dataset:
            for i in range(len(row)):
                  row[i] = (row[i] - minmax[i][0]) / (minmax[i][1] -
minmax[i][0])


# Обчисліть евклідову відстань між двома векторами
def euclidean_distance(row1, row2):

      distance = 0.0
```

```
      for i in range(len(row1)-1):
            distance += (row1[i] - row2[i])**2
      return sqrt(distance)


# Знайдіть найбільш схожих сусідів
def get_neighbors(train, test_row, num_neighbors):

      distances = list()
      for train_row in train:
            dist = euclidean_distance(test_row, train_row)
            distances.append((train_row, dist)
)
      distances.sort(key=lambda tup: tup[1])
      neighbors = list()

      for i in range(num_neighbors):
            neighbors.append(distances[i][0])
      return neighbors


 # Зробіть передбачення з сусідами
def predict_classification(train, test_row, num_neighbors):

      neighbors = get_neighbors(train, test_row, num_neighbors)
      output_values = [row[-1] for row in neighbors]

      prediction = max(set(output_values), key=output_values.count)
      return prediction


#print(X_data.shape, X_data.nbytes / 1000000)
"""
mlDb['all_data_rate'] = 2830743
mlDb['benign_rate'] = 2359289
mlDb['attacks'] = 471454

mlDb.close()
"""
def svmTrain(X, Y, C, kernelFunction, tol=1e-3, max_passes=5, args=()):

    """
   Навчає класифікатор SVM за допомогою спрощеної версії алгоритму SMO.
     Параметри
    ---------
    X : numpy ndarray
         (m x n) Матриця навчальних прикладів. Кожен рядок є навчальним
прикладом, а

         j-й стовпець містить j-у функцію.

     Y : numpy ndarray
         (m, ) Вектор (1-D numpy масив), що містить 1 для позитивних прикладів і
0 для негативних прикладів.

     C : плавати
         Стандартний параметр регуляризації SVM.
     kernelFunction : func
         Дескриптор функції, який обчислює ядро. Функція повинна приймати два
вектори як
         вхідних даних і повертає скаляр як вихід.

     tol : float, необов'язково
         Значення допуску, що використовується для визначення рівності чисел з
плаваючою комою.

     max_passes: int, необов'язково
         Контролює кількість ітерацій над набором даних (без змін на альфа-
версію)
```

```
        до завершення роботи алгоритму.

     args : кортеж
         Додаткові аргументи, необхідні для функції ядра, як-от параметр sigma
для a
         Гауссове ядро.
    Returns
    -------

    model :
        The trained SVM model.

    """
    # make sure data is signed int
    Y = Y.astype(int)
    # Dataset size parameters
    m, n = X.shape

    passes = 0
    E = np.zeros(m)
    alphas = np.zeros(m)
    b = 0

    # Map 0 to -1
    Y[Y == 0] = -1

# Попередньо обчисліть матрицю ядра, оскільки наш набір даних невеликий
     # (на практиці оптимізовані пакети SVM, які обробляють великі набори даних
     # граціозно **не** зробить цього)

     # Ми реалізували оптимізовану векторизовану версію ядра
     # що навчання SVM буде виконуватися швидше
    if kernelFunction.__name__ == 'linearKernel':
         # Векторизоване обчислення для лінійного ядра
          # Це еквівалентно обчисленню ядра на кожній парі прикладів
        K = np.dot(X, X.T)
    elif kernelFunction.__name__ == 'gaussianKernel':

        # векторизоване ядро RBF
         # Це еквівалентно обчисленню ядра на кожній парі прикладівs
        X2 = np.sum(X**2, axis=1)
        K = X2 + X2[:, None] - 2 * np.dot(X, X.T)

        if len(args) > 0:
            K /= 2*args[0]**2

        K = np.exp(-K)
    else:
        K = np.zeros((m, m))

        for i in range(m):
            for j in range(i, m):
                K[i, j] = kernelFunction(X[i, :], X[j, :])

                K[j, i] = K[i, j]

    while passes < max_passes:

        num_changed_alphas = 0
        for i in range(m):

            E[i] = b + np.sum(alphas * Y * K[:, i]) - Y[i]

            if (Y[i]*E[i] < -tol and alphas[i] < C) or (Y[i]*E[i] > tol and
alphas[i] > 0):

                # select the alpha_j randomly
                j = np.random.choice(list(range(i)) + list(range(i+1, m)),
size=1)[0]
```

```
                E[j] = b + np.sum(alphas * Y * K[:, j]) - Y[j]

                alpha_i_old = alphas[i]
                alpha_j_old = alphas[j]

                if Y[i] == Y[j]:
                    L = max(0, alphas[j] + alphas[i] - C)
                    H = min(C, alphas[j] + alphas[i])

                else:
                    L = max(0, alphas[j] - alphas[i])
                    H = min(C, C + alphas[j] - alphas[i])

                if L == H:
                    continue

                eta = 2 * K[i, j] - K[i, i] - K[j, j]

               # цільова функція позитивно визначена, уздовж напрямку буде
мінімум
                # обмеження лінійної рівності, і eta буде більше нуля
                # ми фактично обчислюємо -eta тут (тому ми пропускаємо eta >= 0)

                if eta >= 0:
                    continue

                alphas[j] -= Y[j] * (E[i] - E[j])/eta
                alphas[j] = max(L, min(H, alphas[j]))

                if abs(alphas[j] - alpha_j_old) < tol:
                    alphas[j] = alpha_j_old
                    continue

                alphas[i] += Y[i]*Y[j]*(alpha_j_old - alphas[j])

                b1 = b - E[i] - Y[i]*(alphas[i] - alpha_i_old) * K[i, j] \
                     - Y[j] * (alphas[j] - alpha_j_old) * K[i, j]

                b2 = b - E[j] - Y[i]*(alphas[i] - alpha_i_old) * K[i, j] \
                     - Y[j] * (alphas[j] - alpha_j_old) * K[j, j]

                if 0 < alphas[i] < C:
                    b = b1
                elif 0 < alphas[j] < C:
                    b = b2
                else:
                    b = (b1 + b2)/2

                num_changed_alphas += 1
            if num_changed_alphas == 0:
                passes += 1

            else:
                passes = 0

    idx = alphas > 0
    model = {'X': X[idx, :],
             'y': Y[idx],
             'kernelFunction': kernelFunction,
             'b': b,
             'args': args,
             'alphas': alphas[idx],
             'w': np.dot(alphas * Y, X)}
    return model


def svmPredict(model, X):
```

```
"""
     Повертає вектор прогнозів за допомогою навченої моделі SVM.
     Параметри
     ----------
     модель: dict
         Параметри навченої моделі svm, повернуті функцією svmTrain
     X : схожий на масив
         Матриця (m x n), де кожен приклад є рядком.

     Повертає
     -------
     pred : array_like

         Вектор розміру (m,) значень передбачення {0, 1}.
     """

     # перевірити, чи ми отримуємо вектор. Якщо так, то припустимо, що нам
потрібно робити лише передбачення
     # для окремого прикладу

    if X.ndim == 1:
        X = X[np.newaxis, :]

    m = X.shape[0]
    p = np.zeros(m)
    pred = np.zeros(m)

    if model['kernelFunction'].__name__ == 'linearKernel':
       # ми можемо використовувати ваги та зміщення безпосередньо, якщо працюємо
з лінійним ядром
        p = np.dot(X, model['w']) + model['b']
    elif model['kernelFunction'].__name__ == 'gaussianKernel':
        # векторизоване ядро RBF
         # Це еквівалентно обчисленню ядра на кожній парі прикладів
        X1 = np.sum(X**2, 1)
        X2 = np.sum(model['X']**2, 1)
        K = X2 + X1[:, None] - 2 * np.dot(X, model['X'].T)

        if len(model['args']) > 0:
            K /= 2*model['args'][0]**2

        K = np.exp(-K)
        p = np.dot(K, model['alphas']*model['y']) + model['b']
    else:
        # інше нелінійне ядро
        for i in range(m):
            predictions = 0
            for j in range(model['X'].shape[0]):
                predictions += model['alphas'][j] * model['y'][j] \
                               * model['kernelFunction'](X[i, :], model['X'][j,
:])
            p[i] = predictions

    pred[p >= 0] = 1
    return pred
mlDb = shelve.open(db_file_name)
all_data_rate = mlDb['all_data_rate']
benign_rate = mlDb['benign_rate']
attacks = mlDb['attacks']
print(f"All data: {all_data_rate} \n"
      f"Benign: {benign_rate } "
      f"({round(benign_rate/all_data_rate, 3)}) \n"
       f"attacks: {attacks} ({round(attacks/all_data_rate, 3)})"
)
mlDb.close()
from sklearn.datasets import load_breast_cancer
from sklearn.neighbors import KNeighborsClassifier
import numpy as np
from sklearn.model_selection import train_test_split, StratifiedShuffleSplit
```

```
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import AdaBoostClassifier, GradientBoostingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC

from sklearn.metrics import f1_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import confusion_matrix

def gaussianKernel(x1, x2, sigma):
      sim = 0
      sim = np.exp(-np.sum((x1 - x2) ** 2) / (2 * (sigma ** 2)))

      return sim

def dataset3Params(X, y, Xval, yval):
      C = 1
      sigma = 0.3

      C_array = np.array([0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30])
      sigma_array = np.array([0.01, 0.03, 0.1, 0.3, 1, 3, 10, 30])

      err_array = np.zeros([C_array.size, sigma_array.size])

      for i in np.arange(C_array.size):
            for j in np.arange(sigma_array.size):
                  model = svmTrain(X, y, C_array[i], gaussianKernel,
args=(sigma_array[j],))
                  pred = svmPredict(model, Xval)
                  pred_error = np.mean(pred != yval)

                  err_array[i, j] = pred_error
      ind = np.unravel_index(np.argmin(err_array, axis=None), err_array.shape)
      C = C_array[ind[0]]
      sigma = sigma_array[ind[1]]

      return C, sigma


iris = load_breast_cancer(as_frame=True)


def make_train_test(data, test_ratio, rand_state=0):
    np.random.seed(rand_state)
    shuffle_ind = np.random.permutation(len(data))
    test_len = int(len(data)* test_ratio)
    test_ind = shuffle_ind[:test_len]
    train_ind = shuffle_ind[test_len: ]
    return data.iloc[train_ind], data.iloc[test_ind]

X_train, X_test, y_train, y_test = train_test_split(
    iris.data,iris.target, random_state=0)



print(iris.target.value_counts() / len(iris.target))
print(y_train.value_counts() / len(y_train))

split = StratifiedShuffleSplit(
    n_splits=1,
    test_size=0.2,
    random_state=0
)

for train_idx, test_idx in split.split(iris.data,iris.target):
```

```
    strat_train_data = iris.data.loc[train_idx]
    strat_test_data = iris.data.loc[test_idx]

    strat_train_targ = iris.target.loc[train_idx]
    strat_test_targ = iris.target.loc[test_idx]


print(strat_train_targ.value_counts() / len(strat_train_targ))

alg_list = [GaussianNB, KNeighborsClassifier, AdaBoostClassifier,
           GradientBoostingClassifier, DecisionTreeClassifier, SVC]


ml_list={
"Naive Bayes":GaussianNB(),
"KNN":KNeighborsClassifier(n_neighbors=6),
"GBC" :GradientBoostingClassifier(),
"AdaBoost":AdaBoostClassifier(),
"DTC":DecisionTreeClassifier(max_depth=4,criterion="entropy"),
"SVC": SVC(C=1000)}

for ml in ml_list:
    ml_clf = ml_list[ml]

    ml_clf.fit(strat_train_data, strat_train_targ)

    predictor = ml_clf.predict(strat_test_data)
    accuracy_value = ml_clf.score(strat_test_data, strat_test_targ)
    accuracy_value_train = ml_clf.score(strat_train_data, strat_train_targ)
    f_1 = f1_score(strat_test_targ, predictor, average='macro')
    precision_value = precision_score(strat_test_targ, predictor,
average='macro')
    recall_value = recall_score(strat_test_targ, predictor, average='macro')

    print(f"Name {ml} \n"
          f"Accuracy: {accuracy_value}\n"
          f"Accuracy (train): {accuracy_value_train}\n"
          f"F1 score: {f_1}\n"
          f"Precission: {precision_value}\n"
          f"Recall: {recall_value}\n"
    )
    print("-" * 60)
```