УДК 621.391

РОЗВ'ЯЗАННЯ ЗАДАЧІ КЛАСИФІКАЦІЇ МЕРЕЖНИХ ПРИСТРОЇВ НА ОСНОВІ ПАРАМЕТРІВ БЕЗПЕКИ ЗА ДОПОМОГОЮ МАШИННОГО НАВЧАННЯ

М.А. МАЙБА, О.С. ЄРЕМЕНКО
Харківський національний університет радіоелектроніки

Abstract *– This article investigates the problem of classifying network devices based on their security parameters using machine learning. Due to the constant growth of threats in cyberspace and the need to ensure a high level of network security, the relevance of using machine learning technologies to identify and classify secure devices is exceptionally high. Therefore, the article considers the specifics of applying machine learning algorithms for classification and regression tasks in network environments. Particular attention is paid to a short review of the algorithms most commonly used for classification tasks. The work describes in detail the process of developing a machine-learning model aimed at classifying network devices according to their security indicators. It considers the selection of appropriate parameters for model training, the process of data preprocessing, and the selection and adjustment of the classification algorithm. The results of model training on actual data are also presented. The process of training and evaluating the accuracy and efficiency of the model is described. The results are analyzed, and the choice of optimal hyperparameters is justified. As a result of the study, an effective machine learning model has been developed that can accurately classify network devices by security level, improving network security when selecting potentially secure devices. The study found that the Random Forest and Decision Tree models showed the highest accuracy in predicting the security state of network devices compared to other models, such as Logistic Regression, k-NN, and Gradient Boosting. The performance of the trained model was tested on a validation dataset. The Decision Tree model correctly predicted the security level of approximately 78% of network devices.*

Анотація *– У статті досліджується розв'язання задачі класифікації мережних пристроїв на основі їхніх параметрів безпеки за допомогою машинного навчання. У зв'язку з постійним зростанням загроз у кіберпросторі та необхідністю забезпечення високого рівня захисту мереж, актуальність використання технологій машинного навчання для визначення та класифікації безпечних пристроїв є надзвичайно високою. Отже, розглянуто особливості застосування алгоритмів машинного навчання для задач класифікації та регресії у мережних середовищах. Особливу увагу приділено огляду алгоритмів, які найчастіше використовуються для задач класифікації. В роботі детально описано процес розробки моделі машинного навчання, спрямованої на класифікацію мережних пристроїв за їх показниками безпеки. Розглянуто вибір відповідних параметрів для навчання моделі, процес попередньої обробка даних, а також вибір і налаштування алгоритму класифікації. Також представлено результати навчання моделі на реальних даних. Описано процес навчання, оцінювання точності та ефективності моделі. Проведено аналіз отриманих результатів та обґрунтування вибору оптимальних гіперпараметрів. У результаті дослідження розроблено ефективну модель машинного навчання, здатну точно класифікувати мережні пристрої за рівнем безпеки, що дозволяє покращити захист мережі на етапі вибору потенційно безпечних пристроїв. У процесі дослідження встановлено, що моделі Random Forest та Decision Tree показали найвищу точність прогнозування стану безпеки мережних пристроїв у порівнянні з іншими моделями – Logistic Regression, k-NN та Gradient Boosting. Проведено перевірку працездатності навченої моделі на валідаційному наборі даних. Модель Decision Tree правильно передбачила стан приблизно 78% мережних пристроїв щодо рівня їхньої безпеки.*

Вступ

Сучасні інфокомунікаційні мережі (ІКМ) – складні динамічні системи, які забезпечують передачу й обробку інформації між різними користувачами та пристроями.

ІКМ постійно розвиваються та модернізуються, щоб задовольнити зростаючі потреби сучасного суспільства у швидкості, якості, надійності та безпеці комунікацій. Одним з напрямків покращення ІКМ є застосування штучного інтелекту (ШІ) для автоматизації та оптимізації різних процесів управління мережею [1–6].

Одним з основних інструментів ШІ [1] є машинне навчання (МН, Machine Learning, ML) – процес набуття знань або навичок за допомогою даних. МН дозволяє створювати алгоритми, які можуть адаптуватися до змінюваних умов і виконувати завдання, які важко або неможливо програмувати вручну. Одним із сучасних напрямків МН є глибоке навчання [1, 5–8] (Deep Learning, DL) – підгалузь МН, що використовує багатошарові нейронні мережі для абстрагування високорозмірних даних.

Застосування МН для ІКМ відкриває нові можливості для інтелектуального управління трафіком та маршрутизації [2, 3, 8]. Інтелектуальне управління трафіком – це процес аналізу, прогнозування, оптимізації та контролю потоків даних в мережах з метою покращення якості обслуговування (Quality of Service, QoS) та якості досвіду (Quality of Experience, QoE) користувачів, а також стійкості та безпеки мережі загалом.

Базуючись на порівнянні різних моделей машинного навчання, включаючи лінійну регресію, логістичну регресію, дерева рішень і випадковий ліс, в даній роботі визначено модель для поставленої задачі класифікації мережних пристроїв на основі параметрів безпеки. Використано матрицю помилок для оцінки ефективності моделі, що дозволило виявити специфічні помилки класифікації та визначити, які класи модель класифікує найкраще. Зроблено перевірку моделі на валідаційному наборі даних, який не був використаний під час навчання моделі.

I. Особливості засобів машинного навчання для задач класифікації та регресії в мережах

Основна мета машинного навчання полягає в передбаченні результатів на основі вхідних даних. Чим більше різноманітності даних, тим простіше для алгоритму виявити закономірності, що призводить до більш точних результатів. Для того щоб навчити машину, потрібні три основні компоненти.

Перший компонент – це дані. Наприклад, якщо потрібно оптимізувати процес маршрутизації в мережі, то потрібні дані про трафік. Якщо хочемо передбачати затримки передачі даних, тоді потрібна історія затримок. Якщо хочемо з'ясувати патерни використання мережі користувачами, нам потрібні дані про їхню активність. Для ефективного навчання потрібно максимально велика кількість даних. Другий компонент – це ознаки. Це можуть бути різні характеристики, такі як пропускна здатність мережі, час відгуку сервера, кількість одночасних користувачів або навіть частота використання певних служб. Третій компонент – це алгоритм. Зазвичай одну й ту ж задачу можна розв'язати різними методами. Від вибору методу залежить точність, швидкість роботи і розмір готової моделі. Однак, якщо дані неякісні, то навіть найкращий алгоритм не допоможе.

Отже, ML можна розділити на три категорії: кероване та некероване навчання, а також навчання з підкріпленням. Розглянемо більш детально категорію керованого навчання (Supervised Learning, SL).

У випадку SL машина має так званого вчителя, тобто правильну відповідь. Наприклад, якщо існує великий набір даних про стан мережних пристроїв, у ролі вчителя може виступати цільова змінна, яка заздалегідь розділила всі дані на безпечні та небезпечні пристрої, або за будь-яким іншим параметром. Таким чином, машина може вивчити на основі навчальних даних, які пристрої є безпечними, а які – ні.

Використання навчального алгоритму дозволяє машині навчитися швидше та точніше, тому в практичних задачах його використовують значно частіше. Такі завдання можна поділити на два основні типи – класифікація та регресія. Класифікація передбачає категорію об'єкта, а регресія передбачає значення на числовій осі. Наприклад, класифікація може бути використана для визначення рівня безпечності пристрою, а регресія – для прогнозування ймовірності компрометації каналу зв'язку.

Наведемо короткий огляд алгоритмів [2, 8, 9], які найчастіше використовуються для задач класифікації.

Лінійна регресія (Linear Regression), безумовно, є одним з найвідоміших та зрозумілих алгоритмів у статистиці та машинному навчанні. Прогностичне моделювання передусім стосується мінімізації помилки моделі. Лінійну регресію можна представити у вигляді рівняння, яке описує пряму лінію, що найточніше відображає взаємозв'язок між вхідними змінними X та вихідними змінними Y. Для складання цього рівняння потрібно знайти певні коефіцієнти B для вхідних змінних. Для оцінки регресійної моделі використовуються різні методи, як-от лінійна алгебра або метод найменших квадратів.

Логістична регресія (Logistic Regression) – ще один алгоритм, який прийшов у машинне навчання безпосередньо зі статистики та добре підходить для задач бінарної класифікації. Логістична регресія схожа на лінійну тим, що в ній також потрібно знайти значення коефіцієнтів для вхідних змінних. Різниця полягає в тому, що вихідне значення перетворюється за допомогою нелінійної або логістичної функції. Логістична функція виглядає як велика літера S і перетворює будь-яке значення в число в межах від 0 до 1. Це дуже корисно, оскільки можливо застосувати правило до виходу логістичної функції для прив'язки до 0 і 1 (наприклад, якщо результат функції менше 0,5, то на виході отримуємо 1) та прогнозування класу.

Дерево рішень (Decision Tree) можна представити у вигляді двійкового дерева, знайомого багатьом з алгоритмів та структур даних. Кожен вузол представляє собою вхідну змінну та точку розділення для цієї змінної (за умови, що змінна – число). Листові вузли – це вихідна змінна, яка використовується для прогнозування. Прогнози виконуються шляхом проходження по дереву до листового вузла та виведення значення класу. Дерева швидко навчаються та роблять прогнози. Крім того, вони точні для широкого спектра завдань і не вимагають особливої підготовки даних.

Наївний байєсівський класифікатор (Naive Bayes) – це тип класифікатора, який обчислює ймовірність належності об'єкта до певного класу. Ця ймовірність обчислю-

ється з шансу, що певна подія відбудеться, з урахуванням подій, що вже відбулися. Кожен параметр класифікованого об'єкта вважається незалежним від інших параметрів. Це означає, що вплив кожного параметра на прогнозовану ймовірність вважається незалежним від інших параметрів. Це спрощує обчислення, але також може призвести до менш точних прогнозів, якщо в реальності параметри взаємопов'язані.

Випадковий ліс (Random Forest) – дуже популярний та ефективний алгоритм машинного навчання [9]. Це вид ансамблевого алгоритму, який називається пакуванням (bagging). У ньому використовується той самий підхід, але для оцінки всіх статистичних моделей найчастіше використовуються дерева рішень. Навчальні дані розбиваються на багато вибірок, для кожної з яких створюється модель. Коли потрібно зробити прогноз, його робить кожна модель, а потім прогнози усереднюють, щоб дати кращу оцінку вихідному значенню.

В алгоритмі випадкового лісу для всіх вибірок з навчальних даних будуються дерева рішень. При побудові дерев для створення кожного вузла вибираються випадкові ознаки. Окремо отримані моделі не дуже точні, але при їх об'єднанні якість прогнозування значно покращується. Якщо алгоритм з високою дисперсією, наприклад, дерева рішень, показує хороший результат на вхідних даних, то цей результат часто можна покращити, застосувавши пакування.

II. Розробка моделі машинного навчання для класифікації мережних пристроїв за показниками безпеки

Основною метою завдання є розробка моделі машинного навчання, яка зможе прогнозувати, чи є пристрій безпечним на основі його певних характеристик. Як було описано у розділі вище, для виконання цього завдання необхідний набір даних.

За допомогою API (Application Program Interface) NIST (National Vulnerability Database) [10] був зібраний набір даних. Цей набір включає інформацію щодо мережних пристроїв, а саме: назва пристрою (Device), виробник (Vendor), тип пристрою (Type), інформаційний ризик безпеки (Information Security Risk, ISR), кількість вразливостей, що має пристрій (Amount of vulnerabilities), назву вразливості (Common Vulnerabilities and Exposures, CVE), відображення серйозності вразливості у вигляді рівня (Level) та у вигляді числового балу (BaseScore), імовірність того, що вразливість буде використана чи реалізована (Exploitability).

Слід зазначити, що CVE – це система ідентифікації вразливостей в програмному забезпеченні. Кожна вразливість або експлойт отримує унікальний ідентифікатор CVE, що дозволяє вченим і безпековим спеціалістам легко знаходити інформацію про конкретну вразливість.

Ці ідентифікатори використовуються для обміну даними між різними продуктами безпеки та службами, що дозволяє автоматизувати процеси управління вразливостями. Крім того, система CVE допомагає у координації зусиль щодо виявлення та виправлення вразливостей, що забезпечує більшу безпеку цифрових систем на всіх рівнях [11–13].

В межах проведеного дослідження розмірність початкового набору даних включає 24820 рядків та 9 стовпців. Перші 5 рядків набору даних можна побачити на рис. 1. Робота з набором даних виконана у Google Colab – сервісі, який надає можливість запускати блокноти Jupyter в хмарі. Він дозволяє користувачам редагувати та виконувати код Python без необхідності встановлення його на своєму комп'ютері. Основною причиною вибору цього сервісу стало те, що Google надає безкоштовний доступ до обчислювальних ресурсів, включаючи графічні процесори (GPU) та тензорні процесори (TPU), а це є особливо корисним для завдань машинного навчання.

Перші 5 рядків:

	Device	Vendor	Type	ISR	Mount of vulnerabilities	CVE	BaseScore	Level	Exploitability
1	Cisco 1000V	Cisco	switch	2.07	1	CVE-2020-3508	7.4	HIGH	0.28
2	Cisco 1100	Cisco	router	6.64	5	CVE-2019-12647	7.5	HIGH	0.39
3	Cisco 1100	Cisco	router	6.64	5	CVE-2020-24587	2.6	LOW	0.12
4	Cisco 1100	Cisco	router	6.64	5	CVE-2020-24588	3.5	LOW	0.21
5	Cisco 1100	Cisco	router	6.64	5	CVE-2020-26139	5.3	MEDIUM	0.16

Рис. 1. Перші 5 рядків набору даних

Також сервіс сумісний з більшістю популярних бібліотек Python, що використовуються для аналізу даних та машинного навчання, таких як Pandas, NumPy, Matplotlib, Scikit-Learn тощо [14]. Повний список бібліотек, які використовувались для МН у межах дослідження, можна побачити на рис. 2.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from google.colab import drive
import joblib
from google.colab import files
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

Рис. 2. Бібліотеки, використані під час розробки та дослідження

Набір даних спочатку було отримано у форматі CSV (Comma-Separated Values, CSV). Формат CSV – це простий формат файлу, який використовується для зберігання табличних даних, наприклад, бази даних або електронних таблиць. Він використовує коми для розділення значень у кожному рядку і може бути легко зчитаний і оброблений більшістю програм. Однак, для зручності обробки, набір даних був перетворений

у формат звичайної таблиці Excel, бо він дозволяє легко переглядати та редагувати дані у візуально зручному форматі.

З рис. 1 видно, що дані зберігаються у незручному форматі, бо кожен рядок відображає окрему вразливість CVE, але для одного пристрою може бути декілька вразливостей. Це означає, що інформація про один пристрій розподілена по декількох рядках. Основна проблема полягає у тому, що модель машинного навчання буде обробляти ці рядки, як інформацію про різні пристрої, а не як різні вразливості одного й того ж пристрою. Тому для ефективного аналізу даних і побудови моделі машинного навчання було прийнято рішення залишити для кожного пристрою лише одну вразливість з найвищим значенням BaseScore. Це дозволяє зосередитись на найбільш критичних вразливостях для кожного пристрою та спрощує подальшу обробку даних. У майбутніх роботах буде реалізовано варіант з урахуванням усіх вразливостей для мережного пристрою.

Наступним кроком є підготовка даних для подальшого навчання. Програмна реалізація та результат підготовки даних, представлено на рис. 3 – 4.

```
# Видалення пробілів на початку та наприкінці всіх строкових значень
df = df.applymap(lambda x: x.strip() if isinstance(x, str) else x)
# Видалення пробілів на початку та наприкінці назв стовпців
df.columns = df.columns.str.strip()
# Видалення стовпчика Vendor та (A)Mount of vulnerabilities
df = df.drop('Vendor', axis=1)
df = df.drop('Mount of vulnerabilities', axis = 1)


print(df.head())
print('\n Розмірність набору даних:',df.shape[0])
# Збереження тільки одного рядка Device, у якого найбільший BaseScore
df=df.loc[df.groupby('Device')
['BaseScore'].transform('idxmax')]

df = df.drop_duplicates()
print(df.head())
print(f"\n\nРозмірність набору даних після змін: {df.shape}")
total_devices = len(df)
print(f"Усього пристроїв: {total_devices}")
print('Мінімальне значення ISR:', df['ISR'].min())
print('Максимальне значення ISR:', df['ISR'].max())
```

Рис. 3. Програмний код для підготовки, очищення набору даних і збереження лише найкритичнішої вразливості для кожного пристрою

Стовпець «Vendor» видаляється з набору даних, оскільки всі значення в ньому однакові. Він не вносить варіативності в дані, тому не впливає на модель машинного навчання. Стовпець «Mount of vulnerabilities» також видаляється, оскільки він може заважати навчанню, бо було прийнято рішення розглядати пристрій лише за однією,

найкритичнішою вразливістю. Це дозволяє зосередитись на найбільш критичних вразливостях для кожного пристрою та спрощує подальшу обробку даних.

Вказані кроки є важливою частиною підготовки даних для аналізу та моделювання машинного навчання. Вони допомагають забезпечити те, що дані «чисті», організовані та готові до подальшого аналізу.

```
        Device    Type   ISR             CVE  BaseScore   Level  Exploitability
1  Cisco 1000V  switch  2.07   CVE-2020-3508        7.4    HIGH            0.28
2   Cisco 1100  router  6.64  CVE-2019-12647        7.5    HIGH            0.39
3   Cisco 1100  router  6.64  CVE-2020-24587        2.6     LOW            0.12
4   Cisco 1100  router  6.64  CVE-2020-24588        3.5     LOW            0.21
5   Cisco 1100  router  6.64  CVE-2020-26139        5.3  MEDIUM            0.16

 Розмірність набору даних: 24820
                                Device     Type    ISR             CVE  BaseScore   Level  Exploitability
1                           Cisco 1000V   switch   2.07   CVE-2020-3508        7.4    HIGH            0.28
2                            Cisco 1100   router   6.64  CVE-2019-12647        7.5    HIGH            0.39
10  Cisco 1100 Integrated Services Router   router  10.47   CVE-2020-3474        8.1    HIGH            0.28
12  Cisco 1100 Terminal Services Gateways  unknown   1.82   CVE-2020-3465        6.5  MEDIUM            0.28
13                          Cisco 1100-4p   router  10.72  CVE-2019-12646        7.5    HIGH            0.39


Розмірність набору даних після змін: (2918, 7)
Усього пристроїв: 2918
Мінімальне значення ISR: 0.32
Максимальне значення ISR: 182.54
```

Рис. 4. Результат підготовки даних

Після підготовки відбувається перехід до наступного етапу – аналізу даних. Як видно з рис. 4, розмірність набору даних змінилася з 24820 до 2918 рядків, тепер кожен рядок має інформацію про один пристрій. Також видно, що ISR має дуже великий діапазон від 0,32 до 182, а це може ускладнити аналіз та моделювання даних. Через це, було прийнято рішення розділити цей великий діапазон на менші класи, для того, щоб виявити, яка кількість пристроїв належить до певного класу. Для реалізації поділу даних на діапазони, був реалізований код, показаний на рис. 5.

Код, що представлено на рис. 5, спочатку сегментує дані ISR в різні діапазони. Кожен діапазон представлений окремим кольором. Отже, код обчислює кількість пристроїв, що належать до кожного діапазону, та будує діаграму (рис. 6), відсоткове значення для кожного діапазону представлено у табл. 1.

```
conditions = [
    (df['ISR'] <= 10), (df['ISR'] > 10) & (df['ISR'] <= 20),
    (df['ISR'] > 20) & (df['ISR'] <= 30),
    (df['ISR'] > 30) & (df['ISR'] <= 40),
    (df['ISR'] > 40) & (df['ISR'] <= 50), (df['ISR'] > 50)
]
colors = ['green', 'yellow', 'orange', 'red', 'brown', 'purple']
ranges = ['<=10', '<=20', '<=30', '<=40', '<=50', '50+']
df['Color'] = np.select(conditions, colors)
# підраховуємо кількість пристроїв у кожному діапазони
device_counts = df['Color'].value_counts()
# функція для відображення кількості пристроїв
def absolute_value(val):
    a  = np.round(val/100.*device_counts.sum(), 0)
    return int(a)
fig, axs = plt.subplots(1, 2, figsize=(20, 10))
# будуємо круговую діаграму з відсотками на сегментах

axs[0].pie(device_counts.values, colors=colors, autopct='%1.1f%%')
axs[0].set_title('Розподіл пристроїв за діапазонами ISR (%)')

# будуємо круговую діаграму з числовими значеннями на сегментах
axs[1].pie(device_counts.values, colors=colors, autopct=absolute_value)
axs[1].set_title('Розподіл пристроїв за діапазонами ISR (числове значення)')
# створюємо легенду з діапазонами ISR
legend_labels = [f'{range}' for color, range in zip(colors, ranges)]
fig.legend(legend_labels, title="Діапазони", loc="center right")
plt.figtext(0.5, 0.01, f"Всього пристроїв: {df.shape[0]}", ha="center", fontsize=12)
plt.show()
print('\n\n', df.head())
df = df.drop('Color', axis=1)
```

Рис. 5. Приклад коду для поділу даних за діапазонами

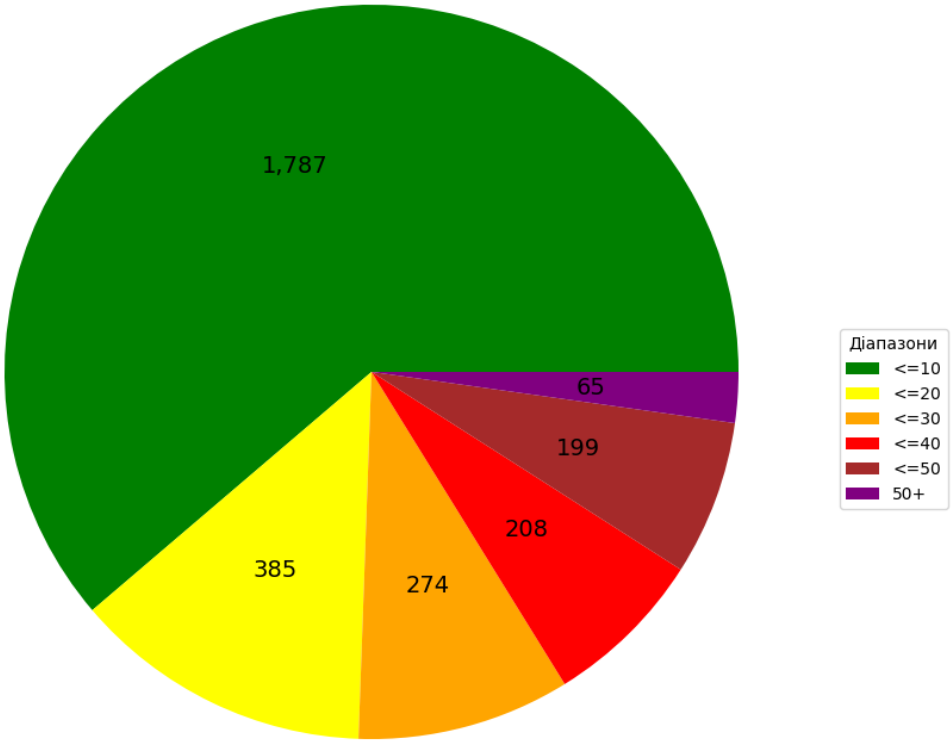

Рис. 6. Кругова діаграма, що відображає розподіл 2918 пристроїв за різними діапазонами ISR

Таблиця 1. Відповідність діапазонів та кількості пристроїв

Діапазон	Кількість пристроїв	Відсоток від загальної кількості пристроїв, %
ISR ⩽ 10	1787	61,3
ISR ⩽ 20	385	13,2
ISR ⩽ 30	274	9,4
ISR ⩽ 40	208	7,1
ISR ⩽ 50	199	6,8
ISR > 50	65	2,2

З рис. 6 видно, що більшість пристроїв у наборі даних має ISR менше 10, що відображено зеленим кольором. Моделі машинного навчання працюють краще, коли є достатньо даних для навчання, тому надалі робота буде тільки з діапазоном даних, де ISR менше або дорівнює 10.

Дані, які належать першому діапазону, зберігаються у новому датафреймі для зручності роботи. Це дозволить зосередитись на цьому конкретному діапазоні без утрати загальної структури даних. Результат та перевірка виконання розподілу ISR до нового набору даних представлена на рис. 7. Після цього цей діапазон знову поділяється на 5 класів. Це робиться для того, щоб мати більш детальний розподіл ISR у межах цього діапазону. Водночас кожен з класів відображає певний рівень безпеки пристрою.

```
[ ]  newDf = df[(df['ISR'] >= 0) & (df['ISR'] <= 10)].copy()
     print(newDf.head())
     print('\n\nМінімальне значення ISR:', newDf['ISR'].min())
     print('Максимальне значення ISR:', newDf['ISR'].max())
     print('Всього пристроїв:', newDf.shape[0])
```

```
                                  Device     Type   ISR            CVE  BaseScore   Level  Exploitability
1                             Cisco 1000V   switch  2.07   CVE-2020-3508        7.4    HIGH            0.28
2                              Cisco 1100   router  6.64  CVE-2019-12647        7.5    HIGH            0.39
12  Cisco 1100 Terminal Services Gateways  unknown  1.82   CVE-2020-3465        6.5  MEDIUM            0.28
49                         Cisco 1111-4PWE   router  6.71   CVE-2020-3559        8.6    HIGH            0.39
51                     Cisco 1111-8PLTEEEAWB   router  6.71   CVE-2020-3559        8.6    HIGH            0.39


Мінімальне значення ISR: 0.32
Максимальне значення ISR: 9.92
Всього пристроїв: 1787
```

Рис. 7. Приклад розділення даних за ISR та збереження у новий набір даних

Після розділення даних на діапазони знову проаналізовано кількість пристроїв у кожному з них. Для цього використовувалась та ж сама функція, що й для попереднього набору даних, тільки на вхід було надано вже нові значення діапазонів та дані. Результат розподілу пристроїв у межах нового діапазону можна побачити на рис. 8.

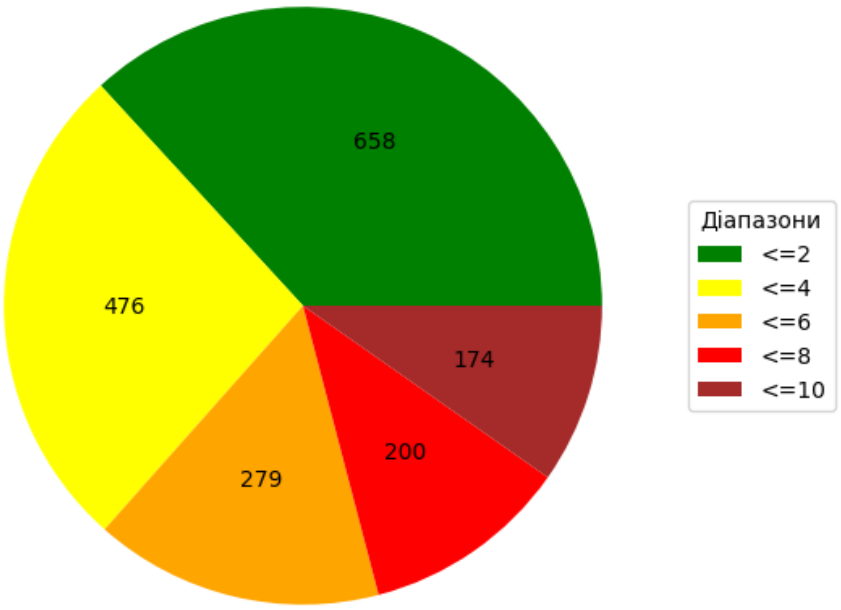

Рис. 8. Кругова діаграма, що відображає розподіл 1787 пристроїв за різними діапазонами ISR у межах першого діапазону (0-10)

Після аналізу даних і визначення розподілу пристроїв за різними діапазонами ISR виконано підготовку до навчання моделі. Для цього до набору даних було додано стовпець «Color». Цей стовпець відображає діапазон ISR для кожного пристрою, який був визначений на попередньому етапі. Кожен діапазон ISR представлений окремим кольором.

Стовпець «Color» буде використовуватися як цільова змінна при навчанні моделі. Це означає, що модель буде навчатися прогнозувати значення цього стовпця на основі інших характеристик пристрою. Результат додавання стовпця наведено на рис. 9. За допомогою цього стовпця можна визначити рівень безпеки пристрою, що робить це завданням класифікації.

	Device	Type	ISR	CVE	BaseScore	Level	Exploitability	Color
1	Cisco 1000V	switch	2.07	CVE-2020-3508	7.4	HIGH	0.28	yellow
2	Cisco 1100	router	6.64	CVE-2019-12647	7.5	HIGH	0.39	red
12	Cisco 1100 Terminal Services Gateways	unknown	1.82	CVE-2020-3465	6.5	MEDIUM	0.28	green
49	Cisco 1111-4PWE	router	6.71	CVE-2020-3559	8.6	HIGH	0.39	red
51	Cisco 1111-8PLTEEEAWB	router	6.71	CVE-2020-3559	8.6	HIGH	0.39	red

Рис. 9. Перевірка додавання стовпця «Color» до набору даних для використання як цільової змінної при навчанні моделі

Наведене завдання є прикладом багатокласової класифікації, оскільки є більше ніж два можливих класи. Це робить завдання більш складним ніж бінарна класифікація, але також дозволяє отримати детальнішу інформацію про рівень безпеки пристроїв.

III. Навчання моделі для класифікації пристроїв на основі рівня безпеки

У Python існує велика кількість бібліотек для машинного навчання. Бібліотека Scikit-Learn значно спрощує процес створення класифікатора, допомагаючи чітко виділити концепції машинного навчання. В ній реалізовано ці концепції за допомогою зрозумілої, добре задокументованої та надійної бібліотеки [5]. Зазначене робить Scikit-Learn відмінним інструментом для розробки та впровадження моделей машинного навчання.

У системах машинного навчання або нейронних мережах існують входи та виходи. Те, що подається на входи, зазвичай називають ознаками (features). Ознаки по суті є тим самим, що й змінні в науковому експерименті – вони характеризують певний спостережуваний феномен, також їх можна кількісно виміряти. Коли ознаки подаються на входи системи машинного навчання, ця система намагається знайти збіги, помітити закономірність між ознаками. Відповідно на виході генерується результат цієї роботи.

Цей результат зазвичай називають міткою (label), оскільки у виходів є певна позначка, видана їм системою, тобто припущення (прогноз) про те, до якої категорії потрапляє вихід після класифікації.

У контексті машинного навчання класифікація відноситься до навчання з учителем. Такий тип навчання передбачає, що дані, які подаються на входи системи, вже позначені, а важлива частина ознак вже розділена на окремі категорії або класи. Тому мережа вже обізнана, яка частина входів важлива, а яку частину можна самостійно перевірити. Таке завдання може бути виконане за допомогою дерева рішень – одного з типів класифікатора в Scikit-Learn.

При некерованому навчанні в систему подаються непозначені дані, і вона повинна спробувати сама розділити ці дані на категорії. Оскільки розв'язується задача класифікації, спосіб некерованого навчання розглядатися не буде.

Процес навчання моделі – це подача даних для нейромережі, яка в результаті повинна вивести певні шаблони для даних. У процесі використання моделі керованого навчання на вхід подаються ознаки та мітки, а при прогнозуванні на вхід класифікатора подаються лише ознаки.

Дані, які приймає мережа, діляться на дві групи: набір даних для навчання та набір для тестування. Проте, не варто перевіряти мережу на тому ж наборі даних, на яких вона навчалася, оскільки модель вже буде «налаштована» під цей набір.

Алгоритми машинного навчання можуть бути описані як навчання цільової функції f, яка найкращим чином відповідає вхідним змінним X та вихідній змінній Y: Y=f(X).

Найбільш поширеною задачею в машинному навчанні є прогнозування значень Y для нових значень X. Це називається прогностичним моделюванням, і наша мета – зробити якомога більш точне прогнозування. Це важливий аспект дослідження, оскільки намагаємося класифікувати рівень безпеки пристрою на основі його характеристик. Використовуючи Scikit-Learn, є можливість навчити модель, яка найкращим чином відповідає вхідним даним (характеристикам пристрою) до вихідних даних (рівня безпеки).

Це дозволить нам робити прогнози про рівень безпеки нових пристроїв. Крім того, Scikit-Learn надає доступ до багатьох алгоритмів класифікації.

Отже, в межах основного завдання з передбачення безпеки пристрою для початку виконано підготовку даних. Було визначено числові та категоріальні ознаки у вихідних даних. Числові ознаки містили «BaseScore» та «Exploitability», а категоріальні ознаки включали «Type», «CVE» та «Level». Визначення числових і категоріальних ознак допомагає з'ясувати, які методи обробки даних будуть найефективнішими. Наприклад, числові ознаки можуть потребувати нормалізації, а категоріальні ознаки – кодування.

Були створені обробники для цих ознак: StandardScaler для числових ознак та OneHotEncoder для категоріальних ознак. Обробники для ознак, як-от StandardScaler та OneHotEncoder, використовуються для перетворення даних у формат, який можна використовувати для навчання моделей. StandardScaler нормалізує числові ознаки, що допомагає моделям краще інтерпретувати ці ознаки. OneHotEncoder перетворює категоріальні ознаки на бінарний формат, який можна використовувати в моделях машинного навчання. Ці обробники дозволили нормалізувати числові ознаки та перетворити категоріальні ознаки на формат, який можна використовувати для навчання моделей.

Дані були розділені на навчальну, тестову та валідаційну вибірки. Використання валідаційної вибірки допомогло уникнути перевірки моделі на тих самих даних, на яких вона навчалася.

Після підготовки даних було проведено навчання моделей, використовуючи різні алгоритми класифікації, доступні в Scikit-Learn, а саме лінійну регресію, логістичну регресію, дерева рішень, наївний байєсівський класифікатор та випадковий ліс. Кожна з цих моделей була навчена (рис. 10), а потім оцінена їхня точність на тестових даних. Це дозволило визначити, яка модель найкраще підходить для поставленої задачі.

Для того щоб проаналізувати результати застосування кожної моделі, можна порівняти їхню точність та обрати найкращу модель для розв'язуваної задачі. Також доцільно провести додаткову оптимізацію моделі, використовуючи методи налаштування гіперпараметрів, як-от пошук сітки або випадковий пошук. Для проведення порівняння точність кожної моделі представлено на рис. 11.

```python
# Створюємо і навчаємо модель Random Forest
model_randFor = RandomForestClassifier(random_state=42)
model_randFor.fit(X_train, y_train)
y_pred = model_randFor.predict(X_test)
accuracy_randFor = accuracy_score(y_test, y_pred)
print(f'Точність моделі(Random Forest): {accuracy_randFor}')

# Створюємо і навчаємо модель Decision Tree
model_DesTree = DecisionTreeClassifier(random_state=42)
model_DesTree.fit(X_train, y_train)
y_pred = model_DesTree.predict(X_test)
accuracy_DesTree = accuracy_score(y_test, y_pred)
print(f'Точність моделі(Decision Tree): {accuracy_DesTree}')

# Створюємо і навчаємо модель Logistic Regression
model_LogReg = LogisticRegression(max_iter=1000, random_state=42)
model_LogReg.fit(X_train, y_train)
y_pred = model_LogReg.predict(X_test)
accuracy_LogReg = accuracy_score(y_test, y_pred)
print(f'Точність моделі(Logistic Regression): {accuracy_LogReg}')

# Створюємо і навчаємо модель k-NN
model_knn = KNeighborsClassifier()
model_knn.fit(X_train, y_train)
y_pred = model_knn.predict(X_test)
accuracy_kNN = accuracy_score(y_test, y_pred)
print(f'Точність моделі(k-NN): {accuracy_kNN}')

# Створюємо і навчаємо модель Gradient Boosting
model_gb = GradientBoostingClassifier(random_state=42)
model_gb.fit(X_train, y_train)
y_pred = model_gb.predict(X_test)
accuracy_gb = accuracy_score(y_test, y_pred)
print(f'Точність моделі(Gradient Boosting): {accuracy_gb}')
```

Рис. 10. Приклад навчання різних моделей, використовуючи бібліотеку Scikit-Learn

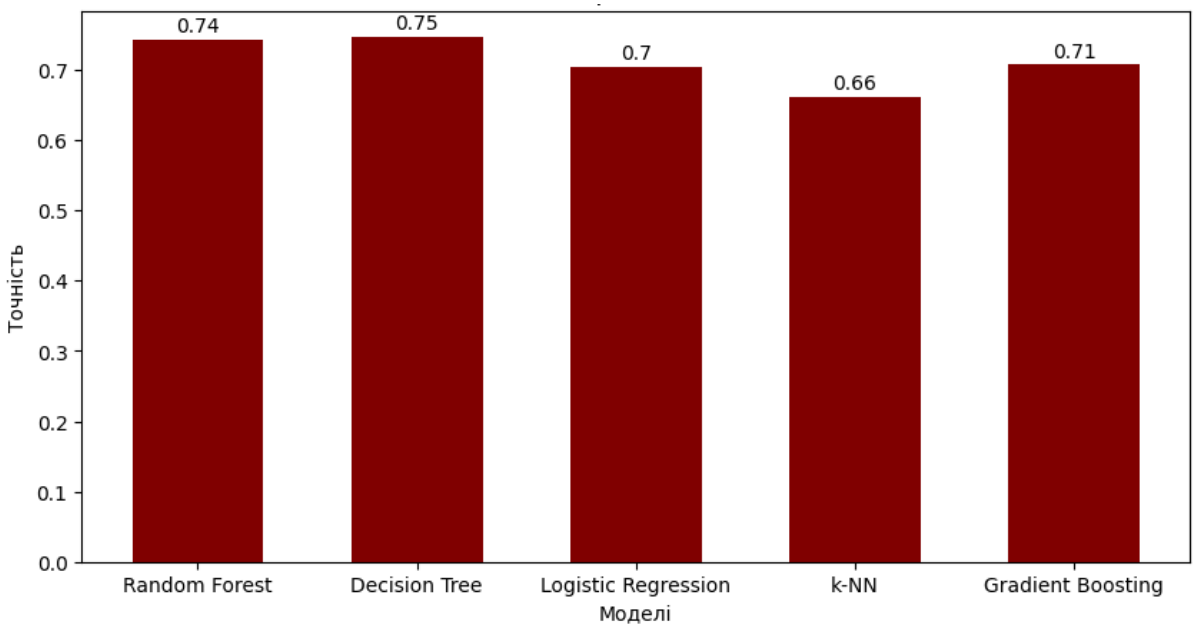

Рис. 11. Точність моделей машинного навчання: Random Forest, Decision Tree, Logistic Regression, k-NN та Gradient Boosting

З рис. 11 видно, що моделі Random Forest та Decision Tree показали найвищу точність у порівнянні з іншими моделями. Це може свідчити про те, що ці моделі були найбільш ефективними для даної задачі класифікації. Logistic Regression, k-NN та Gradient Boosting показали трохи нижчу точність. Це може бути пов'язано з особливостями даних або специфікою задачі. Загалом ці результати підкреслюють важливість проведення ретельного порівняння різних моделей машинного навчання, щоб вибрати найкращу модель для конкретної задачі.

Наступним етапом була побудована матриця помилок (рис. 12) для моделі дерева рішень. Було обрано саме цю модель, бо в неї найкращий результат точності порівняно з іншими моделями. Матриця помилок дозволяє оцінити ефективність моделі, виявляючи не тільки загальну точність, але й специфічні помилки класифікації.

Використовуючи матрицю помилок, було виявлено, що модель дерева рішень найкраще класифікує «жовтий» клас, але має деякі труднощі з «коричневим» та «помаранчевим» класами. Це може свідчити про те, що модель може потребувати додаткового налаштування для покращення її продуктивності на цих конкретних класах.

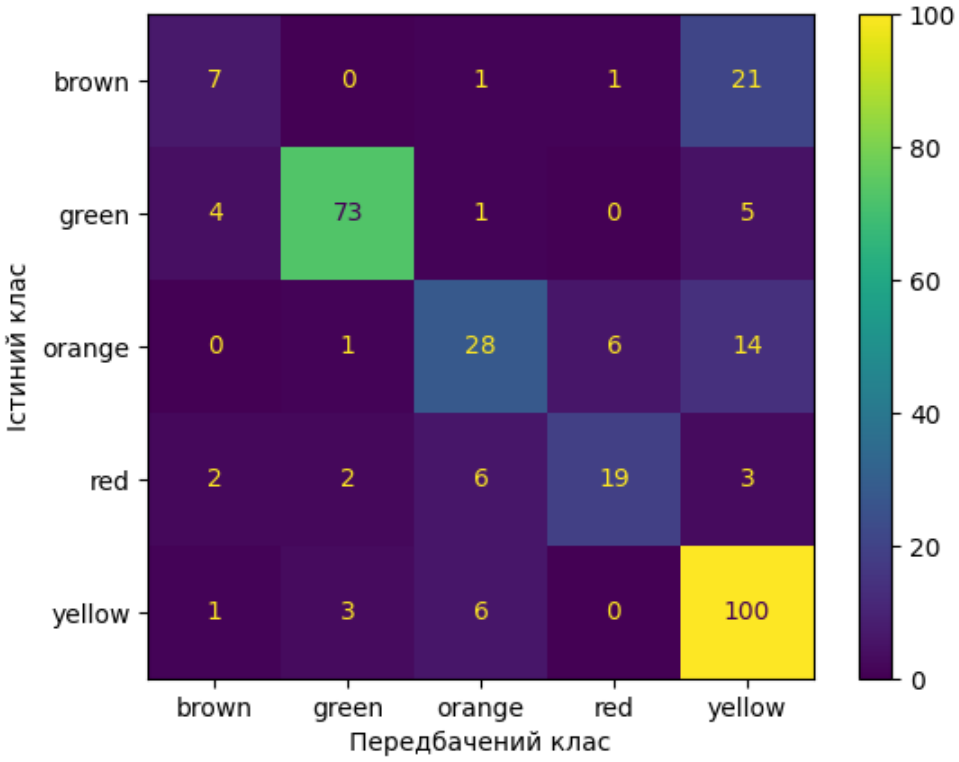

Рис. 12. Матриця помилок для моделі дерева рішень

Осі графіка позначені як «Передбачений клас» та «Істинний клас», і обидві вони варіюються відповідно до п'яти різних класів. Водночас числові значення в комірках представляють кількість прикладів для кожної комбінації істинного і передбаченого класів. Числові значення, розташовані на діагоналі, відображають кількість правильно

класифікованих екземплярів для кожного класу. Вони називаються істинно позитивними передбаченнями. Поза діагоналлю показано помилково класифіковані екземпляри, які називаються «хибно позитивними» і «хибно негативними» передбаченнями.

У даному випадку менша точність моделі для «коричневого» та «помаранчевого» класів може бути пов'язана з недостатнім обсягом даних для цих класів. Тому одним з можливих шляхів покращення цих результатів може бути збір більшої кількості даних для тренування моделей. Збільшення обсягу даних може допомогти моделям краще вивчити шаблони цих класів і, таким чином, покращити їхню точність класифікації.

Останнім етапом перевірки працездатності навченої моделі є перевірка її на валідаційному наборі даних. Цей набір даних не був використаний під час навчання моделі, тому він дозволяє перевірити, наскільки добре модель може узагальнювати своє навчання на нових, невідомих для неї даних. Програмну реалізацію цього завдання представлено на рис. 13.

```
# Зберігаємо модель
joblib.dump(model_DesTree, 'model_DesTree.pkl')

# Завантажуємо модель
loaded_model = joblib.load('model_DesTree.pkl')

test_xls = pd.ExcelFile('/content/drive/MyDrive/validation_set.xlsx')
new_data = pd.read_excel(test_xls )
print('Дані до передбаченням:\n', new_data.head())
# Застосовуємо препроцесор до нових даних
new_data_preprocessed = preprocessor.transform(new_data)

# Використовуємо модель для передбачення на нових даних
predictions = loaded_model.predict(new_data_preprocessed)

# Додаємо передбачення у вихідний DataFrame і зберігаємо його в новий файл Excel
new_data['Color'] = predictions
new_data.to_excel('predicted_data.xlsx', index=False)

files.download('predicted_data.xlsx')
# files.download('model_DesTree.pkl')
print('\n\n Дані після передбачення:\n',new_data.head())
```

Рис. 13. Код для перевірки навченої моделі на валідаційних даних

У даному коді модель Decision Tree, яка була навчена раніше, зберігається за допомогою методу dump і потім завантажується за допомогою методу load. Це дозволяє зберегти стан моделі, щоб її можна було використовувати пізніше без необхідності повторного навчання.

Новий набір даних завантажується з файлу Excel, а потім передбачення генеруються за допомогою завантаженої моделі. Ці передбачення потім додаються до вихідного набору даних і зберігаються в новому файлі Excel. Результат передбачення можна побачити на рис. 14.

Дані до передбаченням:

	Device	Type	ISR	CVE	BaseScore	Level	Exploitability
0	Cisco Catalyst 3750E-48PD-EF	unknown	3.82	CVE-2017-3881	9.8	CRITICAL	0.39
1	Cisco Meraki MR53E	unknown	5.54	CVE-2020-26140	6.5	MEDIUM	0.28
2	Cisco Nexus 9364D-GX2A	switch	1.82	CVE-2023-20089	6.5	MEDIUM	0.28
3	Cisco C6800-8p10g-xl	switch	0.54	CVE-2019-1649	6.7	MEDIUM	0.08
4	Cisco Webex Wireless Phone 860	unknown	1.82	CVE-2020-26141	6.5	MEDIUM	0.28

Дані після передбачення:

	Device	Type	ISR	CVE	BaseScore	Level	Exploitability	Color
0	Cisco Catalyst 3750E-48PD-EF	unknown	3.82	CVE-2017-3881	9.8	CRITICAL	0.39	yellow
1	Cisco Meraki MR53E	unknown	5.54	CVE-2020-26140	6.5	MEDIUM	0.28	orange
2	Cisco Nexus 9364D-GX2A	switch	1.82	CVE-2023-20089	6.5	MEDIUM	0.28	green
3	Cisco C6800-8p10g-xl	switch	0.54	CVE-2019-1649	6.7	MEDIUM	0.08	green
4	Cisco Webex Wireless Phone 860	unknown	1.82	CVE-2020-26141	6.5	MEDIUM	0.28	green

Рис. 14. Результат роботи моделі машинного навчання на валідаційному наборі даних

З рис. 14 видно, що після передбачення у даних з'явився новий стовпець з кольором. Цей стовбець і є результатом роботи моделі МН. Перші 5 пристроїв були класифіковані правильно. Результат передбачення для всіх пристроїв показано на рис. 15.

```
Всього пристроїв: 269
Кількість правильно передбачених пристроїв: 210
```

Рис. 15. Результат роботи моделі машинного навчання на валідаційному наборі даних

Результати на рис. 15 демонструють, що модель здатна класифікувати нові дані з високою точністю, бо з 269 пристроїв правильно передбачено було 210, що свідчить про її ефективність. Однак, як і з будь-якою моделлю машинного навчання, важливо пам'ятати, що точність передбачення може залежати від якості та репрезентативності вхідних даних. Тому, хоча ці результати є обнадійливими, завжди варто проводити додаткову валідацію моделі з використанням нових даних для перевірки її ефективності.

Висновки

Дана стаття присвячена рішенню завдання класифікації безпеки пристроїв за допомогою машинного навчання з використанням числових і категоріальних ознак для передбачення рівня безпеки мережних пристроїв. У роботі проведено порівняння різних моделей машинного навчання, включаючи лінійну регресію, логістичну регресію, дерева рішень та випадковий ліс, з метою визначення найкращої моделі для поставленої задачі. Також використано матрицю помилок для оцінки ефективності моделі, щодозволило виявити специфічні помилки класифікації та визначити, які класи модель класифікує найкраще. Точність моделі може бути покращена шляхом збору більшої кількості даних для тренування моделей. Це допоможе моделям краще вивчити шаблони цих класів і, таким чином, покращити їхню точність класифікації. Перевірено працездатність моделі на валідаційному наборі даних, який не був використаний під час навчання моделі. Це дозволило виявити, наскільки добре модель може узагальнювати своє навчання на нових даних.

Результати роботи підтвердили доцільність інтеграції результатів класифікації мережних пристроїв на основі машинного навчання в процеси оптимізації мереж, що дозволить використовувати передбачувану інформацію про рівень їхньої безпеки [12 – 14]. У процесі дослідження встановлено, що моделі Random Forest та Decision Tree показали найвищу точність прогнозування стану безпеки мережних пристроїв у порівнянні з іншими моделями – Logistic Regression, k-NN та Gradient Boosting. Проведено перевірку працездатності навченої моделі на валідаційному наборі даних. Модель Decision Tree правильно передбачила стан приблизно 78% мережних пристроїв щодо рівня їхньої безпеки.

Список літератури

1. *Майба, М. А., Митцева, О. С.* (2023), "Використання штучного інтелекту для покращення наукової продуктивності", Наука та освіта в дослідженнях молодих учених : матеріали IV Міжнар. наук.-практ. конф. для студ., аспірантів, докторантів, молод. учених, Харків. нац. пед. ун-т ім. Г. С. Сковороди, 18 травня, С. 167–168. URL: https://openarchive.nure.ua/handle/document/23974.

2. *Guo, Z.* (2022). Deep Reinforcement Learning-Based Traffic Engineering in SD-WANs. In: Bringing Machine Learning to Software-Defined Networks. SpringerBriefs in Computer Science. Springer, Singapore, P. 7–22. **DOI**: https://doi.org/10.1007/978-981-19-4874-9_2.

3. *Cicioğlu, M., Çalhan, A.* (2023), "MLaR: machine-learning-assisted centralized link-state routing in software-defined-based wireless networks", Neural Computing and Applications, No. 35(7), P. 5409–5420. **DOI**: https://doi.org/10.1007/s00521-022-07993-w.

4. *Musumeci, F., Rottondi, C., Nag, A., Macaluso, I., Zibar, D., Ruffini, M., Tornatore, M.* (2019), "An overview on application of machine learning techniques in optical networks", IEEE Communications Surveys & Tutorials, No. 21(2), P. 1383–1408. **DOI**: https://doi.org/10.1109/COMST.2018.2880039.

5. *Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., Gao, M., Hou, H., Wang, C.* (2018), "Machine learning and deep learning methods for cybersecurity", IEEE Access, No. 6, P. 35365–35381. **DOI**: https://doi.org/0.1109/ACCESS.2018.2836950.

6. *Luong, N. C., Hoang, D. T., Gong, S., Niyato, D., Wang, P., Liang, Y. C., Kim, D. I.* (2019), "Applications of deep reinforcement learning in communications and networking: A survey", IEEE Communications Surveys & Tutorials, No. 21(4), P. 3133–3174. **DOI**: https://doi.org/10.1109/COMST.2019.2916583.

7. *Arulkumaran, K., Deisenroth, M. P., Brundage, M., Bharath, A. A.* (2017), "Deep Reinforcement Learning: A Brief Survey", IEEE Signal Processing Magazine, No. 34(6), P. 26–38. **DOI**: https://doi.org/10.1109/MSP.2017.2743240.

8. *Fadlullah, Z. M., Tang, F., Mao, B., Kato, N., Akashi, O., Inoue, T., Mizutani, K.* (2017), "State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems", IEEE Communications Surveys & Tutorials, No. 19(4), P. 2432–2455. **DOI**: https://doi.org/10.1109/COMST.2017.2707140.

9. Scikit-Learn. Supervised learning. **URL**: https://scikit-learn.org/stable/supervised_learning.html#supervised-learning.

10. CVE API. NATIONAL VULNERABILITY DATABASE. **URL**: https://nvd.nist.gov/developers/vulnerabilities.

11. *Лемешко, О.В., Єременко, О.С., Євдокименко, М.О., Шаповалова, А.С., Слейман, Б.* (2022), Моделювання та оптимізація процесів безпечної та відмовостійкої маршрутизації в телекомунікаційних мережах: Монографія. Х.: ХНУРЕ, 198 с. **DOI**: https://doi.org/10.30837/978-966-659-378-1.

12. *Лемешко, О. В., Єременко, О. С., Невзорова, О. С.* (2020), Потокові моделі та методи маршрутизації в інфокомунікаційних мережах: відмовостійкість, безпека, масштабованість, Харків: ХНУРЕ, 308 с. DOI**:** https://doi.org/10.30837/978-966-659-282-1.

13. Технології інформаційної безпеки в децентралізованих розподілених мережах : монографія / За загальною редакцією Р. Олійникова, О. Кузнецова та О. Лемешка. Харків: Видавництво «Форт», 2021. 300 с.

14. Scikit-learn: machine learning in Python – scikit-learn 1.3.2 documentation. **URL**: https://scikit-learn.org/stable/.