ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

КАФ. ЕОМ

МЕТОДИ ЗБІЛЬШЕННЯ ДАНИХ ДЛЯ ПОКРАЩЕННЯ КОНТРОЛЬОВАНОГО НАВЧАННЯ ПРИ ВИЯВЛЕННЯ КІБЕРАТАК

ст. групи СПзм-23-1

Баєв І.С.

Керівник

Знайдюк В.Г.

МЕТА ТА ЗАВДАННЯ РОБОТИ

Мета кваліфікаційної роботи це вивчення та оцінка різних методів доповнення даних для підвищення ефективності моделей контрольованого навчання у виявленні кібератак.

Завдання:

- Провести аналіз методів доповнення даних.
- Запропонувати моделі навчання з учителем
- Застосування кількох методів доповнення даних, включаючи SMOTE та ADASYN.
- Провести порівняльний аналіз використаних методів та моделей

МЕТОДИ ДОПОВНЕННЯ ДАНИХ. SMOTE

- Для збагачення набору даних без втрати цінної інформації також можна використовувати передові методи, такі як генерація синтетичних даних за допомогою таких методів, як SMOTE (техніка синтетичної меншості надмірної вибірки).
- Враховуючи вибірку x_i з класу меншин, SMOTE визначає своїх k-найближчих сусідів у просторі ознак. Нехай x_{nn} позначаємо одного з цих k-найближчих сусідів. Синтетичний зразок x_{new} генерується шляхом інтерполяції між x_i і x_{nn} використовуючи рівняння:

$$x_{new} = x_i + \lambda (x_{nn} - x_i)$$

де λ – випадкове число між 0 та 1. Цей крок інтерполяції створює нову вибірку, яка є лінійною комбінацією вихідної вибірки та її сусіда, таким чином зберігаючи загальний розподіл даних, розширюючи при цьому клас меншості.

3

МЕТОДИ ДОПОВНЕННЯ ДАНИХ. ADASYN

- ADASYN (Адаптивна синтетична вибірка) розширює SMOTE, зосереджуючись більше на генерації синтетичних зразків поруч зі зразками класу меншин, які помилково класифіковані класифікатором.
- Математично ADASYN обчислює кількість синтетичних зразків для генерації для кожного зразка класу меншин x_i використовуючи розподіл щільності:

$$r_i = \frac{\gamma_i}{\sum_{i=1}^{N} \gamma_i}$$

де γ_i – кількість вибірок мажорного класу серед k-найближчих сусідів x_i Кількість синтетичних зразків G_i генерувати для кожного x_i пропорційна r_i.

4

МЕТОДИ ДОПОВНЕННЯ ДАНИХ. ТОМЕК

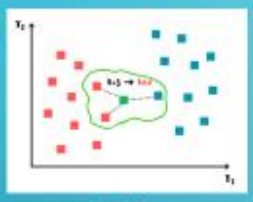
- Доповнення даних за допомогою зв'язків Томека – це метод, який використовується переважно для підвищення продуктивності класифікаторів на незбалансованих наборах даних. Він включає ідентифікацію пар екземплярів, які є найближчими сусідами, але належать до різних класів, та їх видалення для збільшення роздільності класів.
- Між парою екземплярів існує зв'язок Томека x_i і x_j з різних класів, якщо немає екземпляра x_k такий, що $d(x_i,x_k)<d(x_i,x_j)$, де d представляє використовувану метрику відстані, часто евклідову відстань. Математично її можна охарактеризувати так.

$$(y_i \neq y_j) \wedge (\nexists x_k \in S : (d(x_i,x_k) < d(x_i,x_j) \vee d(x_j,x_k) < d(x_j,x_i)))$$

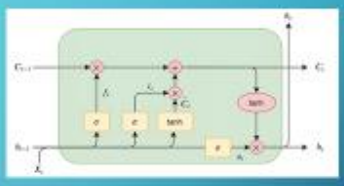
Цей метод особливо ефективний для задач бінарної класифікації та часто використовується як метод очищення даних, а не як метод передискретизації.

5

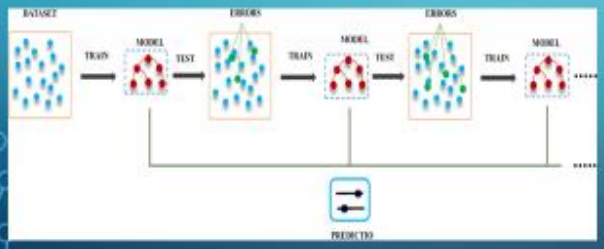
МОДЕЛІ НАВЧАННЯ З УЧИТЕЛЕМ



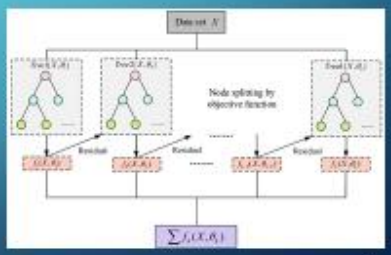
KNN



LSTM



градієнтний бустінг-машина (GBM)



XGBoost (XGB)

6

НАБІР ДАНИХ

Назва змінної	Опис
resp_pkts	Кількість пакетів, надісланих відповідачем під час з'єднання.
service	Тип служби, до якої здійснюється доступ (наприклад, HTTP, FTP).
local_resp	Вказує, чи є відповідач локальним у мережі.
protocol	Мережевий протокол, що використовується в з'єднанні (наприклад, TCP, UDP).
duration	Тривалість з'єднання в секундах.
conn_state	Стан з'єднання (наприклад, встановлено, закрито).
orig_pkts	Кількість пакетів, надісланих ініціатором під час з'єднання.
dest_port	Номер порту призначення з'єднання.
orig_bytes	Кількість байтів, надісланих ініціатором під час з'єднання.
local_orig	Вказує, чи є ініціатор локальним для мережі.
resp_bytes	Кількість байтів, надісланих відповідачем під час з'єднання.
src_port	Номер вихідного порту з'єднання.
techniques_mitre	Методи MITRE ATTACK, пов'язані з кібератакою.

РОЗПОДІЛ TECHNIQUES_MITRE

Розподіл techniques_mitre	Випадки
network_service_discovery	144,279
benign	60,997
reconnaissance_vulnerability_scanning	1581
reconnaissance_wordlist_scanning	715
remote_system_discovery	554
domain_trust_discovery	411
account_discovery_domain	84
reconnaissance_scan_ip_blocks	80
group_policy_discovery	34

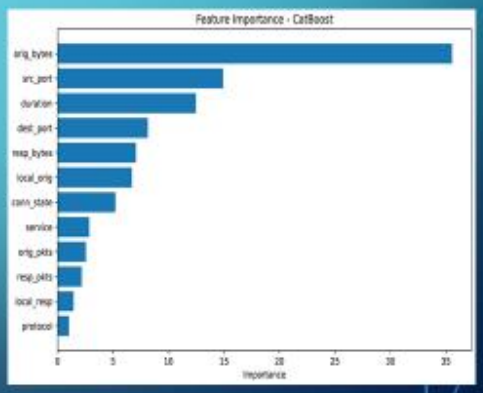
ДОПОВНЕННЯ ДАНИХ

Techniques _MITRE	Original	SMOTE	ADASYN	Borderline-SMOTE	Tomek-Links	SMOTEENN	SMOTE Tomek
Benign	144,279	115,474	115,870	115,474	48,353	107,187	114,001
Account Discovery Domain	60,997	115,474	115,480	115,474	40	114,351	115,334
Domain Trust Discovery	1581	115,474	115,533	115,474	238	113,080	115,030
Group Policy Discovery	715	115,474	115,473	115,474	25	114,503	115,258
Network Service Discovery	554	115,474	115,474	115,474	115,467	115,405	115,470
Reconnaissance Scan IP Blocks	411	115,474	115,478	115,474	62	115,357	115,473
Reconnaissance Vulnerability Scanning	84	115,474	115,751	115,474	1004	111,585	114,734
Reconnaissance Wordlist Scanning	80	115,474	115,475	115,474	577	115,474	115,474
Remote System Discovery	34	115,474	115,475	115,474	431	113,998	115,324

МОДЕЛІ МАШИННОГО НАВЧАННЯ

Значення точності для різних класифікаторів та методів доповнення даних.

Classifier	SMOTE	ADASYN	Borderline SMOTE	Tomek Links	SMOTEENN	SMOTE Tomek
Naïve Bayes	0.497	0.453	0.602	0.718	0.668	0.659
KNN	0.824	0.978	0.981	0.992	0.993	0.990
XGB	0.838	0.925	0.942	0.993	0.981	0.977
GBM	0.842	0.940	0.953	0.989	0.989	0.984
RF	0.833	0.985	0.985	0.994	0.998	0.996
Logistic	0.738	0.741	0.847	0.807	0.860	0.851
RNN	0.759	0.823	0.888	0.979	0.647	0.797
LSTM	0.819	0.875	0.916	0.982	0.945	0.944



Важливість ознак 10

МОДЕЛІ ТА ЇХ ПАРАМЕТРИ

У таблиці наведено параметри, що використовуються кожною моделлю. Для GBM ключові гіперпараметри включають кількість оцінок, швидкість навчання та максимальну глибину. Їх налаштування включає наступне:

- n_estimators: більше число зазвичай збільшує складність моделі. Перехресна перевірка допомагає знайти оптимальний баланс, щоб уникнути перенавчання;
- learning_rate: контролює внесок кожного дерева. Низькі значення зазвичай вимагають більшої кількості дерев;
- max_depth: обмежує глибину окремих дерев для контролю перенавчання.

Для KNN основним гіперпараметром є кількість сусідів.

_neighbors: Невелике число може призвести до шумних прогнозів, тоді як велике число може згладити прогноз, але ігнорувати локальні закономірності. Пошук по сітці з перехресною перевіркою зазвичай використовується для визначення оптимального значення.

Ключові гіперпараметри включають максимальну кількість ітерацій та ваги класів:

- max_iter: забезпечує збіжність. Вищі значення дозволяють розв'язувачу виконати більше ітерацій для збіжності, що особливо корисно для складних наборів даних;
- class_weight: балансує набір даних, коригуючи ваги обернено пропорційно частотам класів. Це особливо важливо для незбалансованих наборів даних.

Model	Parameters
GBM	n_estimators = 100, learning_rate = 0.1, max_depth = 3
KNN	n_neighbors = 5
Logistic Regression	max_iter = 10,000, class_weight = 'balanced'
LSTM	optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy']
GaussianNB	none
Random Forest	n_estimators = 100
RNN	optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy']
XGB	n_estimators = 100, learning_rate = 0.1, max_depth = 3,
	eval_metric = 'mlogloss'

ВИСНОВКИ

В ході кваліфікаційної роботи було вивченно та оціненно різні методи доповнення даних для підвищення ефективності моделей контрольованого навчання у виявленні кібератак.

Вирішенні наступні завдання:

- Проведено аналіз методів доповнення даних.
- Запропоновано моделі навчання з учителем
- Застосовано декілька методів доповнення даних, включаючи SMOTE та ADASYN.
- Проведено порівняльний аналіз використаних методів та моделей

Апробація результатів відбувалося на IX Міжнародної студентської наукової конференції Теоретичне та практичне застосування результатів сучасної науки. м.Умань, 13 червня, 2025 рік / ГО «Молодіжна наукова ліга. Тези доповіді «Методи збільшення даних для покращення контрольованого навчання при виявлення кібератак»

12

ДОДАТОК Б
ПРОГРАМНА РЕАЛІЗАЦІЯ ЗАСТОСУНКУ

```
import tkinter as tk
from tkinter import filedialog, messagebox, ttk
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import classification_report,
confusion_matrix
import os

class CyberAttackApp:
    def __init__(self, root):
        self.root = root
        self.root.title("Аналіз кіберзагроз")
        self.root.geometry("700x500")

        self.label = tk.Label(root, text="Оберіть CSV-файл з
даними")
        self.label.pack(pady=10)

        self.button_load = tk.Button(root, text="Завантажити
файл", command=self.load_file)
        self.button_load.pack(pady=5)

        self.model_label = tk.Label(root, text="Оберіть модель
машинного навчання")
        self.model_label.pack(pady=5)

        self.model_combo = ttk.Combobox(root, values=["Random
Forest", "Logistic Regression", "SVM", "Decision Tree"])
        self.model_combo.current(0)
        self.model_combo.pack(pady=5)

        self.button_run = tk.Button(root, text="Запустити
класифікацію", command=self.run_model)
        self.button_run.pack(pady=5)

        self.button_report = tk.Button(root, text="Показати
матрицю похибки", command=self.show_confusion_matrix)
        self.button_report.pack(pady=5)
```

```
        self.button_plot = tk.Button(root, text="Побудувати
графіки", command=self.show_plots)
        self.button_plot.pack(pady=5)

        self.result_text = tk.Text(root, height=10, width=85)
        self.result_text.pack(pady=10)

        self.df = None
        self.model = None
        self.X_test = None
        self.y_test = None
        self.y_pred = None

    def load_file(self):
        file_path = filedialog.askopenfilename(filetypes=[("CSV
Files", "*.csv")])
        if file_path:
            self.df = pd.read_csv(file_path)
            messagebox.showinfo("Успіх", f"Файл завантажено:
{os.path.basename(file_path)}")

    def preprocess_data(self):
        df = self.df.copy()
        if 'techniques_mitre' not in df.columns:
            messagebox.showerror("Помилка", "У наборі даних має
бути колонка 'techniques_mitre'")
            return None, None

        le_dict = {}
        for col in df.select_dtypes(include=['object']).columns:
            le = LabelEncoder()
            df[col] = le.fit_transform(df[col].astype(str))
            le_dict[col] = le

        X = df.drop('techniques_mitre', axis=1)
        y = df['techniques_mitre']

        scaler = StandardScaler()
        X_scaled = scaler.fit_transform(X)

        return train_test_split(X_scaled, y, test_size=0.3,
random_state=42)

    def run_model(self):
        if self.df is None:
            messagebox.showerror("Помилка", "Будь ласка,
завантажте файл")
            return

        X_train, X_test, y_train, y_test =
self.preprocess_data()
        if X_train is None:
            return
```

```
        selected_model = self.model_combo.get()
        if selected_model == "Random Forest":
            self.model = RandomForestClassifier(random_state=42)
        elif selected_model == "Logistic Regression":
            self.model = LogisticRegression(max_iter=1000)
        elif selected_model == "SVM":
            self.model = SVC()
        elif selected_model == "Decision Tree":
            self.model = DecisionTreeClassifier(random_state=42)

        self.model.fit(X_train, y_train)
        self.y_pred = self.model.predict(X_test)

        self.X_test = X_test
        self.y_test = y_test

        report = classification_report(y_test, self.y_pred)
        self.result_text.delete('1.0', tk.END)
        self.result_text.insert(tk.END, report)

    def show_confusion_matrix(self):
        if self.y_test is None or self.y_pred is None:
            messagebox.showerror("Помилка", "Спочатку запустіть
класифікацію")
            return

        cm = confusion_matrix(self.y_test, self.y_pred)
        plt.figure(figsize=(10, 6))
        sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')
        plt.title('Матриця похибки')
        plt.xlabel('Передбачено')
        plt.ylabel('Фактично')
        plt.tight_layout()
        plt.show()

    def show_plots(self):
        if self.df is None:
            messagebox.showerror("Помилка", "Спочатку завантажте
дані")
            return

        plt.figure(figsize=(10, 6))
        sns.countplot(x='techniques_mitre', data=self.df)
        plt.title('Кількість записів по класах атак')
        plt.xticks(rotation=45)
        plt.tight_layout()
        plt.show()

        plt.figure(figsize=(10, 6))
        sns.histplot(self.df['duration'], bins=50, kde=True)
        plt.title('Розподіл тривалості сесій')
        plt.xlabel('Тривалість')
```

```
        plt.tight_layout()
        plt.show()

if __name__ == '__main__':
    root = tk.Tk()
    app = CyberAttackApp(root)
    root.mainloop()
```