ДОДАТОК А

Графічний матеріал кваліфікаційної роботи

Міністерство освіти та науки України
Харківський національний університет радіоелектроніки

КВАЛІФІКАЦІЙНА РОБОТА

Програмні засоби розпізнавання шкідливого програмного забезпечення з використанням машинного навчання

Виконала:
ст. КІУКІ-21-2
Світлична В.А.

Керівник:
ас. Романюк О.С.

Мета та завдання роботи

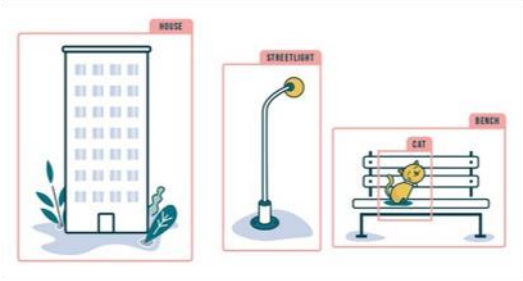
Мета роботи: розробка програмних засобів виявлення шкідливого програмного забезпечення з використанням методів машинного навчання.

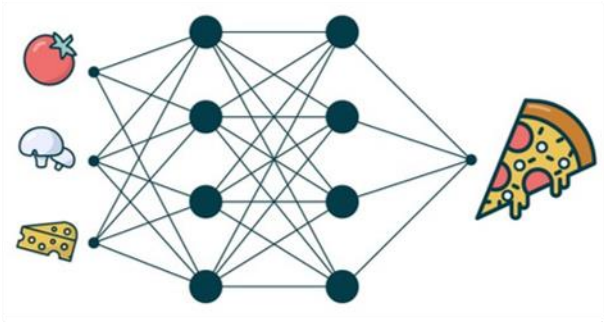
Завдання:

- аналіз існуючих методів машинного навчання;
- аналіз типів шкідливих програм;
- розробка модулів програмних засобів виявлення шкідливого програмного забезпечення на основі методів машинного навчання;
- реалізація в Google Colab.

Машинне навчання

3





Процес рішення завдань за допомогою машинного навчання

4

1	2	3
Визначення зі способами вирішення завдання засобами машинного навчання.	Обрання метрики якості	Отримання даних і виділення ознак
4	**5**	**6**
Визначення з класами моделей	Підготовка даних для навчання і оцінки	Проведення навчання моделей, оцінка результатів

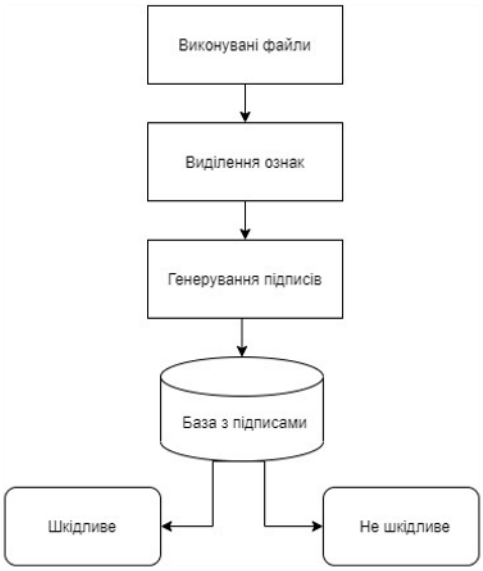
Порівняння традиційного та нового покоління шкідливого ПЗ

5

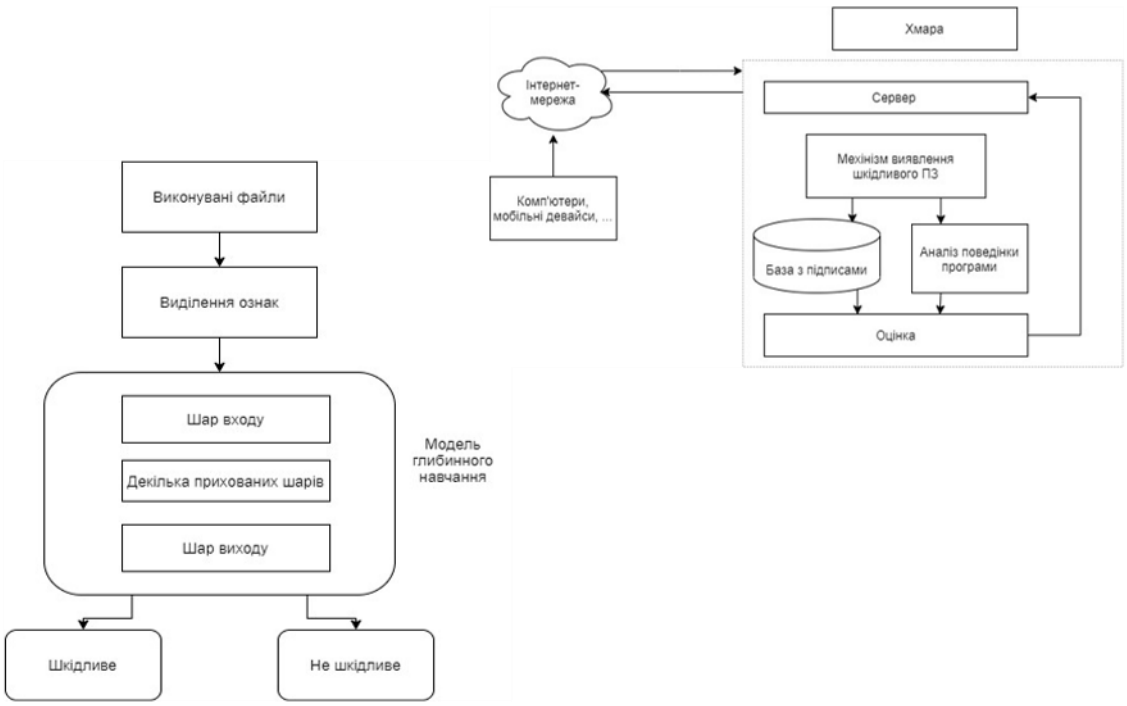
Параметр для порівняння	Традиційне	Нове покоління
Рівень реалізації	простий	складний
Поширення	всі копії однакові	всі копії різні
Джерело поширення	файли .exe розширення	файли з різними розширеннями
Знаходження у системі	тимчасове	постійне
Взаємодія з іншими процесами	декілька процесів (або один)	багато процесів
Приховування	ні	так
Ціль атаки	звичайні комп'ютери	різні девайси(комп'ютери,смартфони, сервери, ...)

6

Підходи на основі сигнатур

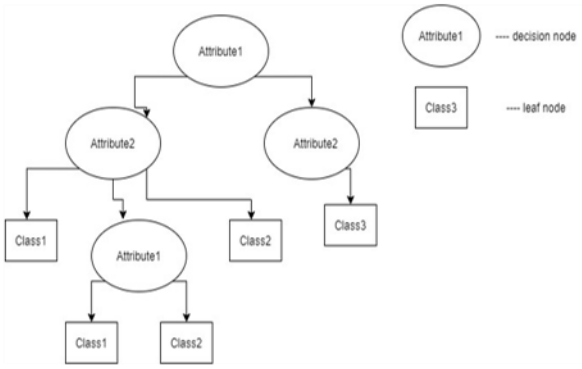


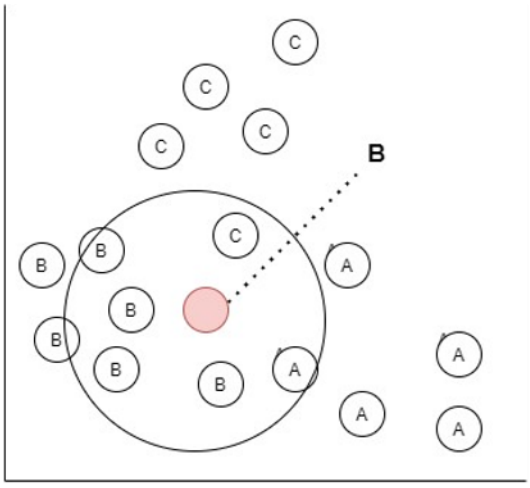
Підходи з використанням глибокого навчання та хмарних обчислень 7



8

Модель дерева рішень та метод k-найближчих сусідів





Етапи процесу впровадження. Характеристики віртуальних машин

9

- ✓ Конфігурація пісочниці;
- ✓ витяг ознак (за допомогою Python 2.7);
- ✓ вибір ознак (за допомогою R);
- ✓ застосування методів машинного навчання (за допомогою R);
- ✓ оцінка результатів.

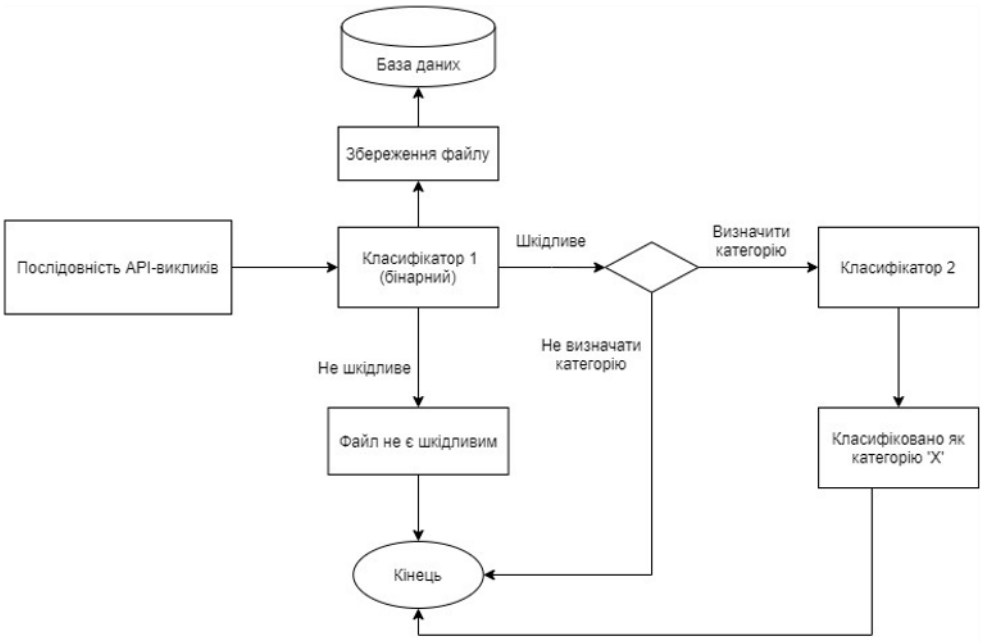
Усі віртуальні машини мають такі характеристики:

- ➢ 1 процесорне ядро 3,2 ГГц;
- ➢ 4 Гб оперативної пам'яті;
- ➢ підключення до інтернету;
- ➢ встановлене програмне забезпечення на всіх віртуальних машинах:

- Windows 7;
- Adobe PDF reader 9.0;
- Adobe Flashplayer 11.7.700.169;
- пакунки, що розповсюджуються Visual Studio 2005 - 2013.
- Java JRE 7
- .NET Framework 4.0

Архітектура розробленого модуля КС

10



Частотне представлення

11

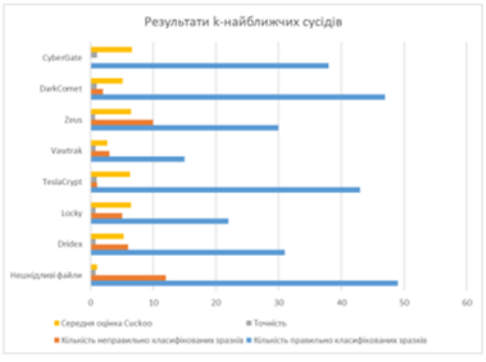
$$API_{bin} = \begin{matrix} \\ S_1 \\ S_2 \\ \vdots \\ S_n \end{matrix} \begin{matrix} \begin{matrix} API_1 & API_2 & \dots & API_n \end{matrix} \\ \begin{bmatrix} 1 & 1 & \cdots & 0 \\ 1 & 0 & \cdots & 1 \\ \vdots & \ddots & \ddots & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix} \end{matrix}$$

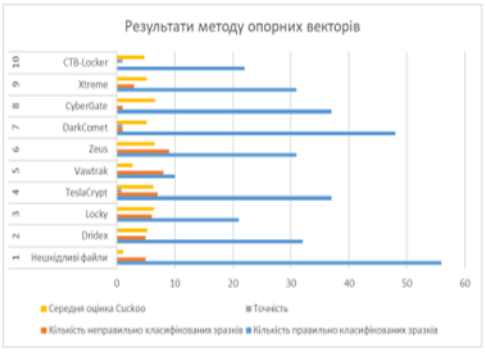
$$API_{freq} \\ = \begin{matrix} \\ S_1 \\ S_2 \\ \vdots \\ S_n \end{matrix} \begin{matrix} \begin{matrix} API_1 & API_2 & \dots & API_n \end{matrix} \\ \begin{bmatrix} 112 & 312 & \cdots & 72 \\ 16 & 23 & \cdots & 315 \\ \vdots & \ddots & \ddots & \vdots \\ 157 & 1 & \cdots & 567 \end{bmatrix} \end{matrix}$$

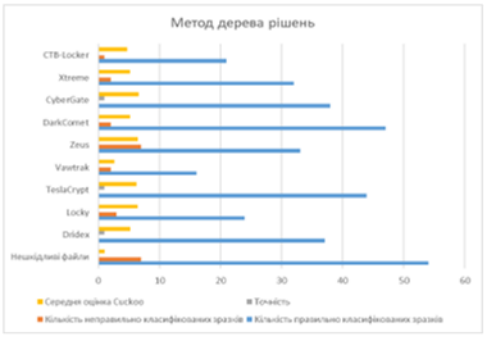
$$ombination \\ \begin{matrix} \\ S_1 \\ = S_2 \\ \vdots \\ S_n \end{matrix} \begin{matrix} \begin{matrix} Pass_1 & \dots & Pass_n & Fail_1 & \dots \end{matrix} \\ \begin{bmatrix} 23 & \cdots & 3 & 224 & \cdots \\ 52 & \cdots & 21 & 224 & \cdots \\ \vdots & \ddots & \ddots & \vdots & \ddots \\ 52 & \cdots & 22 & 210 & \cdots \end{bmatrix} \end{matrix}$$

Результати роботи

12

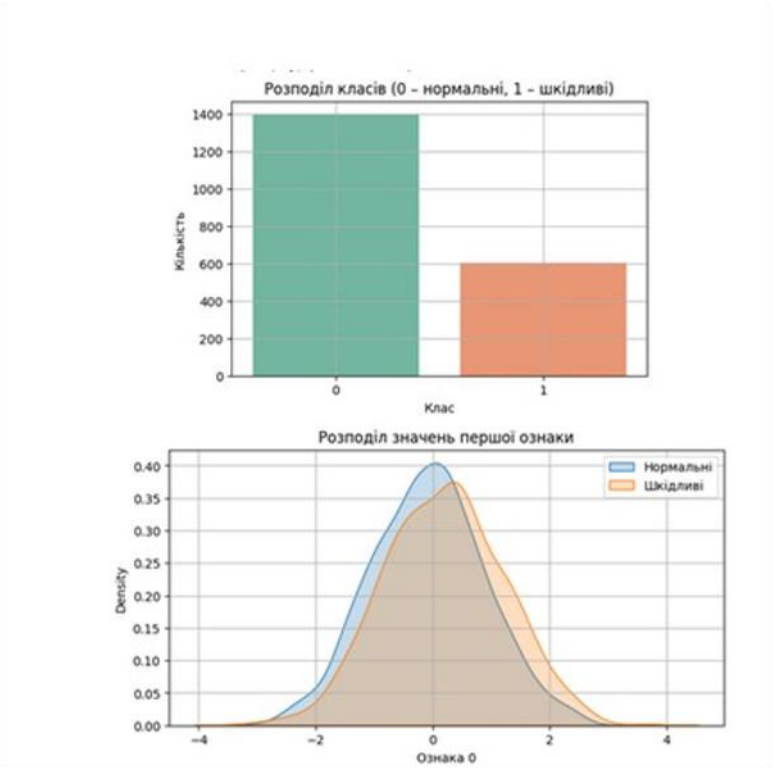






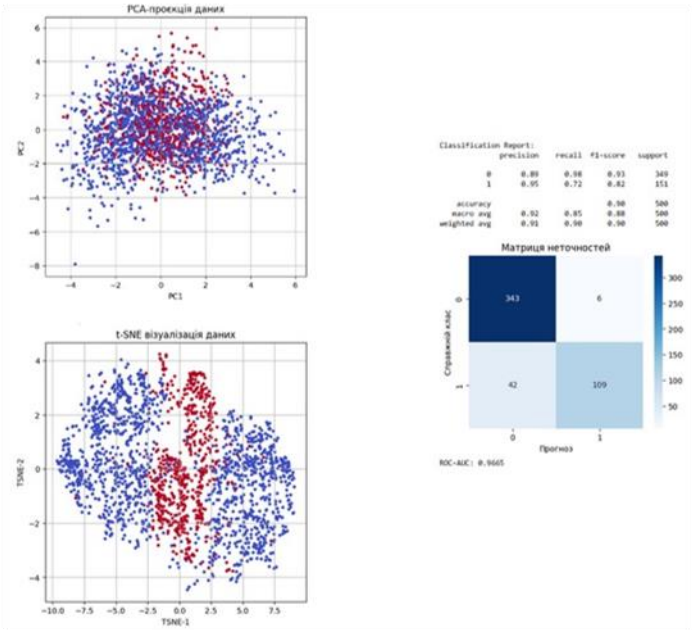
13

Результати роботи



14

Результати роботи



Висновки

15

Обрано та застосовано методи вилучення та представлення ознак, а також оцінено алгоритми машинного навчання. Як метод представлення ознак обрано комбіновану матрицю, що визначає частоту успішних та невдалих викликів API разом із кодами повернення.

Ця репрезентація обрана через відображення фактичної поведінки файлу та об'єднання інформації про різноманітні системні зміни, включаючи модифікації реєстру та файлів, на відміну від інших методів.

Порівняльний аналіз існуючих підходів до виявлення шкідливого програмного забезпечення виявив обмеженість традиційних методів на основі сигнатур у протидії сучасним загрозам. Шкідливе програмне забезпечення нового покоління, що використовує методи обфускації, поліморфізму та метаморфізму, потребує принципово нових підходів до виявлення та класифікації.

Експериментальне дослідження п'яти алгоритмів машинного навчання на наборі даних із 2140 файлів продемонструвало значні відмінності в ефективності різних методів. Алгоритм випадкового лісу показав найвищу точність, досягнувши 95,69% для багатокласової класифікації та 96,8% для бінарної класифікації, що перевищує результати інших досліджуваних методів.

ДОДАТОК Б

Лістинг коду

```
# Програмні засоби розпізнавання шкідливого програмного
забезпечення з використанням машинного навчання

!pip install seaborn scikit-learn --quiet

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,
confusion_matrix, roc_auc_score
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.decomposition import PCA
from sklearn.manifold import TSNE

# Генерація або завантаження прикладового датасету
from sklearn.datasets import make_classification
X, y = make_classification(n_samples=2000, n_features=20,
n_informative=15,
                           n_redundant=5, n_classes=2,
weights=[0.7, 0.3], random_state=42)

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

# Візуалізація розподілу класів
plt.figure(figsize=(6,4))
sns.countplot(x=y, palette='Set2')
plt.title("Розподіл класів (0 – нормальні, 1 – шкідливі)")
plt.xlabel("Клас")
plt.ylabel("Кількість")
plt.grid(True)
plt.show()

# Розподіл ознак (приклад для першої ознаки)
plt.figure(figsize=(8, 4))
sns.kdeplot(X_scaled[y == 0][:, 0], label='Нормальні',
fill=True)
sns.kdeplot(X_scaled[y == 1][:, 0], label='Шкідливі', fill=True)
plt.title("Розподіл значень першої ознаки")
plt.xlabel("Ознака 0")
plt.legend()
plt.grid(True)
```

```
plt.show()

# Зменшення розмірності для візуалізації
X_pca = PCA(n_components=2).fit_transform(X_scaled)
plt.figure(figsize=(6,6))
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap='coolwarm',
s=10)
plt.title("PCA-проєкція даних")
plt.xlabel("PC1")
plt.ylabel("PC2")
plt.grid(True)
plt.show()

# t-SNE для глибшої візуалізації
X_tsne = TSNE(n_components=2, perplexity=30, n_iter=300,
random_state=42).fit_transform(X_scaled)
plt.figure(figsize=(6,6))
plt.scatter(X_tsne[:, 0], X_tsne[:, 1], c=y, cmap='coolwarm',
s=10)
plt.title("t-SNE візуалізація даних")
plt.xlabel("TSNE-1")
plt.ylabel("TSNE-2")
plt.grid(True)
plt.show()

# Поділ на train/test
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.25, stratify=y, random_state=42)

# Побудова моделі
model = RandomForestClassifier(n_estimators=100,
random_state=42)
model.fit(X_train, y_train)

# Прогноз
y_pred = model.predict(X_test)
y_prob = model.predict_proba(X_test)[:, 1]

# Оцінка
print("Classification Report:")
print(classification_report(y_test, y_pred))
# Матриця неточностей
plt.figure(figsize=(5,4))
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True,
fmt='d', cmap='Blues')
plt.title("Матриця неточностей")
plt.xlabel("Прогноз")
plt.ylabel("Справжній клас")
plt.show()

# ROC-AUC
roc_auc = roc_auc_score(y_test, y_prob)
print(f"ROC-AUC: {roc_auc:.4f}")
```