НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

із лабораторної роботи №3

з дисципліни «Системи глибинного навчання»

на тему: “Нейромережеве розпізнавання кібератак”

Варіант 7 – Розпізнавання мережевої кібератаки типу guess\_passwd на базі PNN

|  |  |
| --- | --- |
| Виконав: | Викладач: |
| студент групи КМ-03 | Професор кафедри ПМА |
| Шаповалов Г. Г. | Терейковський І. А. |

Київ — 2023

Зміст

[Теоретичні відомості 3](#_Toc151738651)

[Основна частина 4](#_Toc151738652)

[Перелік посилань 8](#_Toc151738653)

[Додаток А – Код програми 9](#_Toc151738654)

## Теоретичні відомості

Ймовірнісна нейронна мережа (PNN) - це прямопрогонна нейронна мережа, яка широко використовується у задачах класифікації та розпізнавання шаблонів. В алгоритмі PNN батьківська функція розподілу ймовірностей (PDF) кожного класу наближається за допомогою методу ядерних оцінок густини ймовірності, Гаусових функцій.

Операції в PNN організовані в багатошарову прямопрогонну мережу з чотирма шарами:

1. Вхідний шар: Кожен нейрон вхідного шару представляє змінну прогнозування.
2. Шар шаблонів: Цей шар містить один нейрон для кожного випадку в навчальному наборі даних.
3. Шар додавання: Для PNN є один нейрон шаблону для кожної категорії цільової змінної.
4. Вихідний шар: Вихідний шар порівнює зважені голоси для кожної цільової категорії, накопичені в шарі шаблонів, і використовує найбільший голос для прогнозування цільової категорії.

PNN часто використовуються в задачах класифікації. Вони були виведені з Байєсової мережі та статистичного алгоритму.

# Основна частина

На основі датасету NSL-KDD [2] треба навчити та протестувати нейронну мережу PNN.

Назви колонок у датасеті:

'duration','protocol\_type','service','flag','src\_bytes','dst\_bytes','land','wrong\_fragment','urgent','hot',

                'num\_failed\_logins','logged\_in','num\_compromised','root\_shell','su\_attempted','num\_root',

                'num\_file\_creations','num\_shells','num\_access\_files','num\_outbound\_cmds','is\_host\_login','is\_guest\_login',

                'count','srv\_count','serror\_rate','srv\_serror\_rate','rerror\_rate','srv\_rerror\_rate','same\_srv\_rate',

                'diff\_srv\_rate','srv\_diff\_host\_rate','dst\_host\_count','dst\_host\_srv\_count','dst\_host\_same\_srv\_rate',

                'dst\_host\_diff\_srv\_rate','dst\_host\_same\_src\_port\_rate','dst\_host\_srv\_diff\_host\_rate','dst\_host\_serror\_rate',

                'dst\_host\_srv\_serror\_rate','dst\_host\_rerror\_rate','dst\_host\_srv\_rerror\_rate','attack','level'

Колонка - 'attack' відповідає за вид атаки. Якщо normal то атака не підтвердилась.

Оскільки стандартне розподілення на тестові та тренувальні дані має дуже велику різницю в балансі класів:

Кількість 'guess\_passwd' у 'train\_data': 53

Кількість 'guess\_passwd' у 'test\_data' : 1231

Кількість 'normal' у 'train\_data': 67342

Кількість 'normal' у 'test\_data' : 9711

Тому зробимо спільний датасет та розподілимо його так, щоб тренувальна вибірка була 80% від усього датасету. Тепер:

Кількість 'guess\_passwd' у 'train\_data': 1027

Кількість 'guess\_passwd' у 'test\_data' : 257

Кількість 'normal' у 'train\_data': 61642

Кількість 'normal' у 'test\_data' : 15411

Перекодування колонок із категорійних у числові відбувається за допомогою pandas.get\_dummies()

Після кодування з’являються зайві стовпці, тому перевіримо як багато даних, які створюють ці стовпці:

Кількість 'link' у 'train\_data': 1

Кількість 'shell' у 'train\_data': 4

Кількість 'SH' у 'train\_data': 2

Кількість 'RSTOS0' у 'train\_data': 1

Кількість 'remote\_job' у 'test\_data' : 1

Як бачимо, цих даних не багато, порівняно з об’ємами датасету, тому видалимо їх. Тепер кількість колонок у навчальному та тестову датасеті однакова:

Колонок у 'filtered\_train\_data': 74

Колонок у 'filtered\_test\_data': 74

Кількість спільних колонок: 74

Створюємо нейронну мережу PNN у якої на виході буде 2 нейрони (атака, не атака). Навчаємо на тренувальних даних. Параметр сігма = 0.1

Тестуємо мережу та дивимось на метрики, щоб зрозуміти наскільки добре вона працює:

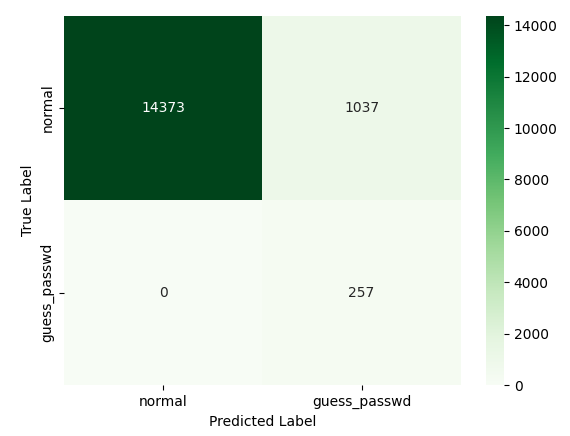
Accuracy (Точність) : 93.38099 %

Precision (Точність): 19.8609 %

Recall (Повнота) : 100.0 %

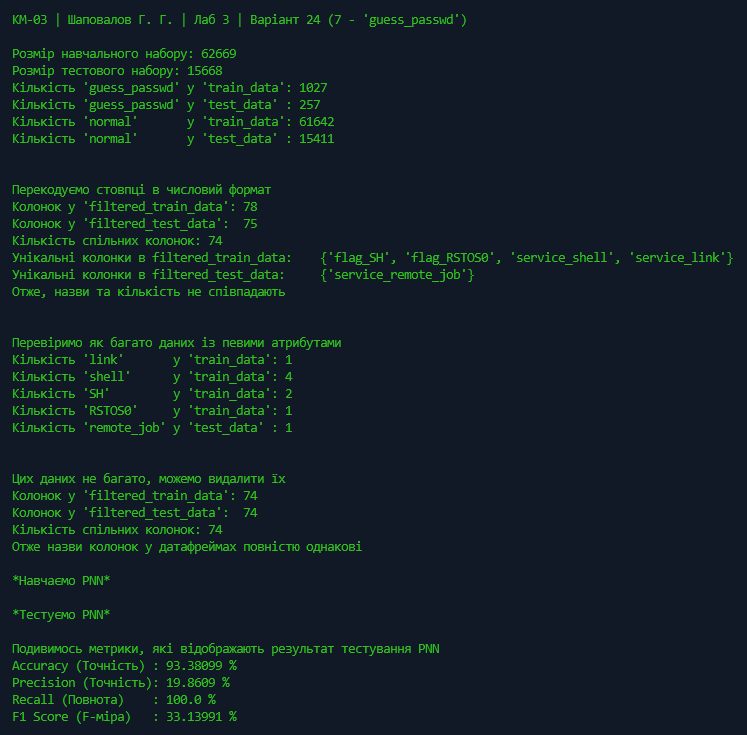
F1 Score (F-міра) : 33.13991 %

Як бачимо, повнота = 100%, тобто всі атаки типу `guess\_passwd` були розпізнані. Побудуємо теплову карту для наглядності.



Як бачимо, PNN помилилась у 1037 випадках, сказавши, що була атака, але атаки не було. Результат непоганий, але його можна покращити кращим підбором параметра сігма, збільшенням навчального набору даних та детальному вивченні кожної колонки в датасеті (можливо є колонки, які не впливають на результат або навпаки ‘заплутують’ модель)

Результат відпрацювання програми:



# Перелік посилань

1. Руденко О.Г. Штучні нейронні мережі. Навч. посіб. / О. Г. Руденко, Є. В. Бодянський.
2. NSL-KDD - <https://www.kaggle.com/datasets/hassan06/nslkdd/data?select=KDDTrain%2B_20Percent.txt>

# Додаток А – Код програми

import time

import numpy as np

import pandas as pd

import seaborn as sns

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

class PNN:

    def \_\_init\_\_(self, input\_size, output\_size):

        self.input\_size = input\_size

        self.output\_size = output\_size

        self.mean\_vectors = None

        self.sigma = None

        self.weights = None

    def train(self, X, y, sigma=1.0):

        self.mean\_vectors = []

        self.sigma = sigma

        # Обчислюємо середні вектори для кожного класу

        for class\_label in range(self.output\_size):

            class\_samples = X[y == class\_label]

            mean\_vector = np.mean(class\_samples, axis=0)

            self.mean\_vectors.append(mean\_vector)

        self.mean\_vectors = np.array(self.mean\_vectors)

        # Обчислюємо ваги для кожного класу

        self.weights = np.ones(self.output\_size) / self.output\_size

    def predict(self, X):

        predictions = []

        for sample in X:

            probabilities = []

            # Розраховуємо ймовірності для кожного класу

            for class\_label in range(self.output\_size):

                mean\_vector = self.mean\_vectors[class\_label]

                sample = sample.astype(float)

                activation = np.exp(-0.5 \* np.sum((sample - mean\_vector) \*\* 2) / (self.sigma \*\* 2))

                probability = activation \* self.weights[class\_label]

                probabilities.append(probability)

            # Визначаємо клас з найвищою ймовірністю

            predicted\_class = np.argmax(probabilities)

            predictions.append(predicted\_class)

        return np.array(predictions)

def check\_guess\_pass\_normal():

    count\_train\_guess\_pass = (train\_data['attack'] == 'guess\_passwd').sum()

    count\_test\_guess\_pass = (test\_data['attack'] == 'guess\_passwd').sum()

    count\_train\_normal = (train\_data['attack'] == 'normal').sum()

    count\_test\_normal = (test\_data['attack'] == 'normal').sum()

    print(f"Кількість 'guess\_passwd' у 'train\_data': {count\_train\_guess\_pass}")

    print(f"Кількість 'guess\_passwd' у 'test\_data' : {count\_test\_guess\_pass}")

    print(f"Кількість 'normal'       у 'train\_data': {count\_train\_normal}")

    print(f"Кількість 'normal'       у 'test\_data' : {count\_test\_normal}")

if \_\_name\_\_ == '\_\_main\_\_':

    print('\n\nКМ-03 | Шаповалов Г. Г. | Лаб 3 | Варіант 24 (7 - \'guess\_passwd\')')

    start\_timer = time.time()

    train\_data = pd.read\_csv('KDDTrain+.txt')

    test\_data = pd.read\_csv('KDDTest+.txt')

    columns = (['duration','protocol\_type','service','flag','src\_bytes','dst\_bytes','land','wrong\_fragment','urgent','hot',

                'num\_failed\_logins','logged\_in','num\_compromised','root\_shell','su\_attempted','num\_root',

                'num\_file\_creations','num\_shells','num\_access\_files','num\_outbound\_cmds','is\_host\_login','is\_guest\_login',

                'count','srv\_count','serror\_rate','srv\_serror\_rate','rerror\_rate','srv\_rerror\_rate','same\_srv\_rate',

                'diff\_srv\_rate','srv\_diff\_host\_rate','dst\_host\_count','dst\_host\_srv\_count','dst\_host\_same\_srv\_rate',

                'dst\_host\_diff\_srv\_rate','dst\_host\_same\_src\_port\_rate','dst\_host\_srv\_diff\_host\_rate','dst\_host\_serror\_rate',

                'dst\_host\_srv\_serror\_rate','dst\_host\_rerror\_rate','dst\_host\_srv\_rerror\_rate','attack','level'])

    train\_data.columns = columns

    test\_data.columns = columns

    # Поєднання двох датафреймів

    combined\_data = pd.concat([train\_data, test\_data], axis=0)

    # Розділення за типом атаки

    attack\_data = combined\_data[combined\_data['attack'] == 'guess\_passwd']

    normal\_data = combined\_data[combined\_data['attack'] == 'normal']

    # Розбиття на навчальний та тестовий набір

    train\_attack, test\_attack = train\_test\_split(attack\_data, test\_size=0.2, random\_state=42)

    train\_normal, test\_normal = train\_test\_split(normal\_data, test\_size=0.2, random\_state=42)

    # Об'єднання навчальних та тестових наборів

    train\_data = pd.concat([train\_attack, train\_normal], axis=0)

    test\_data = pd.concat([test\_attack, test\_normal], axis=0)

    # Виведення розмірів навчального та тестового наборів

    print(f'\nРозмір навчального набору: {train\_data.shape[0]}')

    print(f'Розмір тестового набору: {test\_data.shape[0]}')

    ### Перевіримо кількість записів які містять тип атаки guess\_passwd або 'normal'

    check\_guess\_pass\_normal()

    ### Прибираємо з датасету всі записи, які не містять 'normal' або guess\_passwd

    train\_data = train\_data[(train\_data['attack'] == 'guess\_passwd') | (train\_data['attack'] == 'normal')]

    test\_data  = test\_data[ (test\_data['attack']  == 'guess\_passwd') | (test\_data['attack']  == 'normal')]

    ### Перекодуємо категорійні значення

    # Використання one-hot encoding для категорійних змінних

    filtered\_train\_data = pd.get\_dummies(train\_data, drop\_first=True)

    filtered\_test\_data  = pd.get\_dummies(test\_data,  drop\_first=True)

    ### Перевіримо чи співпадають назви колонок та їх кількість

    # Це важливо оскільки ми передаємо ці значення в нейронну мережу

    print(f'\n\nПерекодуємо стовпці в числовий формат')

    print(f'Колонок у \'filtered\_train\_data\': {filtered\_train\_data.shape[1]}')

    print(f'Колонок у \'filtered\_test\_data\':  {filtered\_test\_data.shape[1]}')

    # Отримання назв колонок для обох датафреймів

    columns\_train = set(filtered\_train\_data.columns)

    columns\_test = set(filtered\_test\_data.columns)

    # Знаходження спільних назв колонок

    common\_columns = columns\_train.intersection(columns\_test)

    # Виведення назв та кількості спільних колонок

    print('Кількість спільних колонок:', len(common\_columns))

    # Знаходження унікальних колонок у кожному датафреймі

    unique\_columns\_train = columns\_train.difference(columns\_test)

    unique\_columns\_test = columns\_test.difference(columns\_train)

    # Виведення назв унікальних колонок у кожному датафреймі

    print(f'Унікальні колонки в filtered\_train\_data:    {unique\_columns\_train}')

    print(f'Унікальні колонки в filtered\_test\_data:     {unique\_columns\_test}')

    print(f'Отже, назви та кількість не співпадають')

    ### Подивимось наскільки багато цих даних і чи можемо ми їх позбутися

    a = (train\_data['service'] == 'link').sum()

    b = (train\_data['service'] == 'shell').sum()

    c = (train\_data['flag'] == 'SH').sum()

    d = (train\_data['flag'] == 'RSTOS0').sum()

    e = (test\_data['service'] == 'remote\_job').sum()

    print(f'\n\nПеревіримо як багато даних із певими атрибутами')

    print(f"Кількість 'link'       у 'train\_data': {a}")

    print(f"Кількість 'shell'      у 'train\_data': {b}")

    print(f"Кількість 'SH'         у 'train\_data': {c}")

    print(f"Кількість 'RSTOS0'     у 'train\_data': {d}")

    print(f"Кількість 'remote\_job' у 'test\_data' : {e}")

    ### Видаляємо дані

    # Тренувальний датасет містить більше 67 тис. даних, а тестовий датасет 9 тис.

    # Тому видалення такої кількості даних повинно сильно вплинути на результати досліджень

    # Видалення записів з умовами у train\_data

    print(f'\n\nЦих даних не багато, можемо видалити їх')

    train\_data = train\_data[(train\_data['service'] != 'link') &

                            (train\_data['service'] != 'shell') &

                            (train\_data['flag'] != 'SH') &

                            (train\_data['flag'] != 'RSTOS0')]

    # Видалення записів з умовами у test\_data

    test\_data = test\_data[(test\_data['service'] != 'remote\_job')]

    ### Перевіримо чи зараз співпадають назви та кількість колонок

    # Розділення 'attack' від інших змінних

    attacks\_train = train\_data[['attack']]

    attacks\_test = test\_data[['attack']]

    # Видалення 'attack' з оригінального набору даних

    train\_data = train\_data.drop(columns=['attack'])

    test\_data = test\_data.drop(columns=['attack'])

    # Перекодування за допомогою pd.get\_dummies() для інших змінних

    filtered\_train\_data = pd.get\_dummies(train\_data, drop\_first=True)

    filtered\_test\_data = pd.get\_dummies(test\_data,  drop\_first=True)

    # Додавання нового стовпця 'attack\_guess\_passwd'

    filtered\_train\_data['attack\_guess\_passwd'] = attacks\_train['attack'].map(lambda x: 1.0 if x == 'guess\_passwd' else 0.0)

    filtered\_test\_data['attack\_guess\_passwd'] = attacks\_test['attack'].map(lambda x: 1.0 if x == 'guess\_passwd' else 0.0)

    print(f'Колонок у \'filtered\_train\_data\': {filtered\_train\_data.shape[1]}')

    print(f'Колонок у \'filtered\_test\_data\':  {filtered\_test\_data.shape[1]}')

    # Отримання назв колонок для обох датафреймів

    columns\_train = set(filtered\_train\_data.columns)

    columns\_test = set(filtered\_test\_data.columns)

    # Знаходження спільних назв колонок

    common\_columns = columns\_train.intersection(columns\_test)

    # Виведення назв та кількості спільних колонок

    print('Кількість спільних колонок:', len(common\_columns))

    print('Отже назви колонок у датафреймах повністю однакові')

    ### Для коректної роботи НМ треба нормалізувати дані

    scaler = MinMaxScaler()

    new\_columns = filtered\_train\_data.columns

    filtered\_train\_data[new\_columns] = scaler.fit\_transform(filtered\_train\_data[new\_columns])

    filtered\_test\_data[new\_columns] = scaler.transform(filtered\_test\_data[new\_columns])

    ### Розділимо датафрейми `filtered\_train\_data` та `filtered\_test\_data` на X\_train, X\_test, y\_train, y\_test

    X\_train = filtered\_train\_data.drop(columns=['attack\_guess\_passwd'])

    y\_train = filtered\_train\_data['attack\_guess\_passwd']

    X\_test = filtered\_test\_data.drop(columns=['attack\_guess\_passwd'])

    y\_test = filtered\_test\_data['attack\_guess\_passwd']

    print(f'\n\*Навчаємо PNN\*')

    ### Ініціалізуємо модель PNN

    input\_size = train\_data.shape[1]

    output\_size = 2  # розмірність вихідного шару (два класи: guess\_passwd та normal)

    pnn = PNN(input\_size, output\_size)

    ### Навчання PNN

    pnn.train(X\_train, y\_train, sigma=0.1)

    ### Використання PNN

    print(f'\n\*Тестуємо PNN\*')

    predictions = pnn.predict(X\_test.values)

    ### Знайдемо метрики, щоб зрозуміти наскільки хороших результатів було досягнуто

    y\_true = y\_test

    accuracy = accuracy\_score(y\_true, predictions)

    precision = precision\_score(y\_true, predictions)

    recall = recall\_score(y\_true, predictions)

    f1 = f1\_score(y\_true, predictions)

    print(f'\nПодивимось метрики, які відображають результат тестування PNN')

    print(f"Accuracy (Точність) : {round(accuracy \* 100, 5)} %")

    print(f"Precision (Точність): {round(precision \* 100, 5)} %")

    print(f"Recall (Повнота)    : {round(recall \* 100, 5)} %")

    print(f"F1 Score (F-міра)   : {round(f1 \* 100, 5)} %")

    ### Побудуємо теплову карту з результатами дослідження

    cm = confusion\_matrix(y\_true, predictions)

    sns.heatmap(cm, annot=True, fmt="d", cmap="Greens",

                xticklabels=['normal', 'guess\_passwd'], yticklabels=['normal', 'guess\_passwd'])

    plt.title('Теплова карта прогнозування кібератаки типу \'guess\_passwd\'\n')

    plt.xlabel("Predicted Label")

    plt.ylabel("True Label")

    plt.show()