НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ

ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра прикладної математики

Звіт

із лабораторної роботи №3

з кредитного модуля «Методи штучного інтелекту»

для студентів

спеціальності 113 «Прикладна математика»

|  |  |
| --- | --- |
| Виконала: | Керівник: |
| студентка групи КМ-81 | Терейковська Л.О. |
| Верзун Поліна |  |

Київ – 2021

# ЗМІСТ

[ЗМІСТ 2](#_Toc57209532)

[ВСТУП 3](#_Toc57209533)

[ХІД РОБОТИ 4](#_Toc57209534)

[Попередня обробка даних 4](#_Toc57209535)

[Навчання мережі. Розпізнавання. 6](#_Toc57209536)

[ВИСНОВКИ 9](#_Toc57209537)

[ДОТАТОК А 10](#_Toc57209538)

# ВСТУП

В даній роботі буде описано процес створення моделі для розпізнавання атаки та portsweep з таблиці KDD99. Як засіб аналізу буде використаний перцептрон структури 54-7-1.

# ХІД РОБОТИ

## Попередня обробка даних

Для роботи з даними було імпортовано модуль pandas.

Наступним кроком буде завантаження таблиці до програми, видалення всіх записів про атаки, що не розглядаються та вибірки певної частини таблиці, а саме 10000 рядків (5000 «нормальних» та всі 5000 «атак»), що можна побачити на рисунку 1.

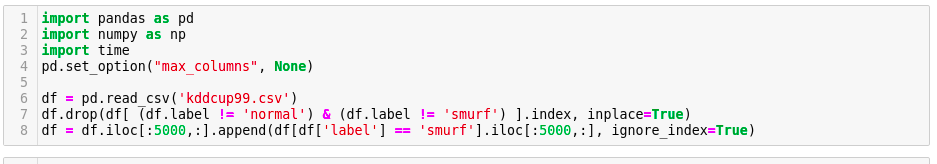


Рисунок 1

Далі необхідно закодувати категоріальні змінні, інакше їх обробка буде неможлива. Для цього зручно користуватись так званими фіктивними змінними, тобто коли кожному класу, присутньому в змінній, що кодується, виділяється окрема змінна. Це дозволить розбити категоріальну змінну на декілька колонок з нулями та одиницями, де певному класу відповідає 1. Очевидно, що в одному рядку серед всіх створених фіктивних змінних може бути лише одна одиниця.

Це не єдиний варіант кодування, але він завжди працює, навідміну кодування, де кожному класу присвоюється номер. В цьому випадку виникає хибне трактування, бо незалежні класи перетворюються на числову пряму і, наприклад, можна зробити висновок, що «клас1»+«клас2»=«клас3», що є абсолютною нісенітницею. Є і інші способи кодування, наприклад замінити назву класу значенням кількості атак шуканого класу. Проте це не можна використати в даному випадку через незбалансованість даних. То ж було використано, найочевидніший працюючий метод.

А щодо вихідної змінної ‘label’, там всього 2 класи, тому закодуємо її, як ‘normal’=0.0001 (щоб уникнути ділення на 0, при підрахунку дельти) та ‘smurf’=1

Наступним кроком є нормалізація даних. Для цього використаємо MinMaxScaler із sklearn.preprocessing, який виконує наступні дії з даними

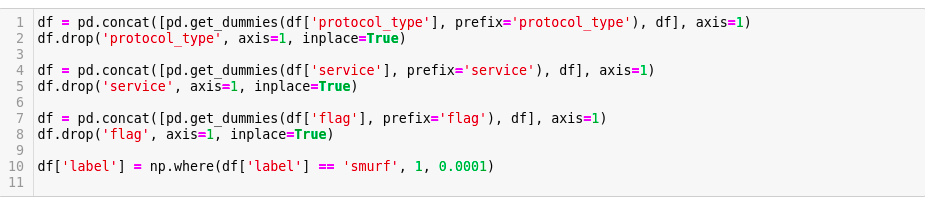


Рисунок 2

Тобто нормалізація відбувається відносно мінімального та максимального значень змінних. Так як нормалізація перетворила змінну ‘label’ = [1, 0.0001] на змінну за значеннями [1, 0], знову явно вкажемо необхідні значення (рис. 3):

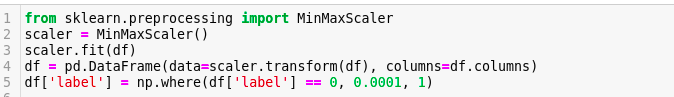


Рисунок 3

Дані готові для аналізу, необхідно лише розділити їх на тестову та тренувальну вибірки. Для цього використаємо train\_test\_split із sklearn.model\_selection. Необхідно зазначити, що частина тестової вибірки – 25%, і вказаний параметр, що зберігає відношення кількості значені шуканої змінної у тестовій та тренувальній змінній, тобто 25% класу 1 точно потрапить до тестової змінної, що видно на рисунку 5, де у тестову вибірку потрапило 260 класи, а у тренувальну 780, що як раз 75% від всіх 1040ка.

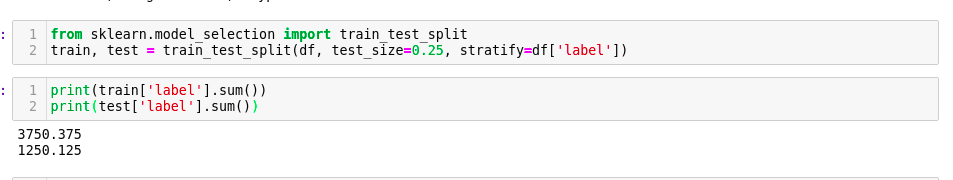
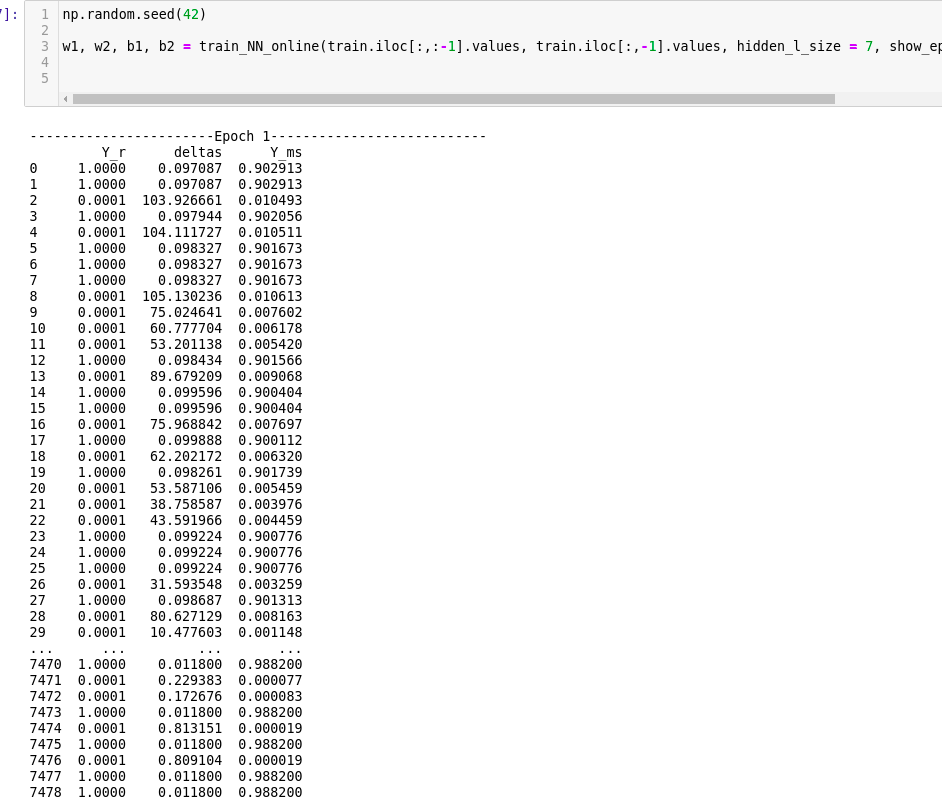


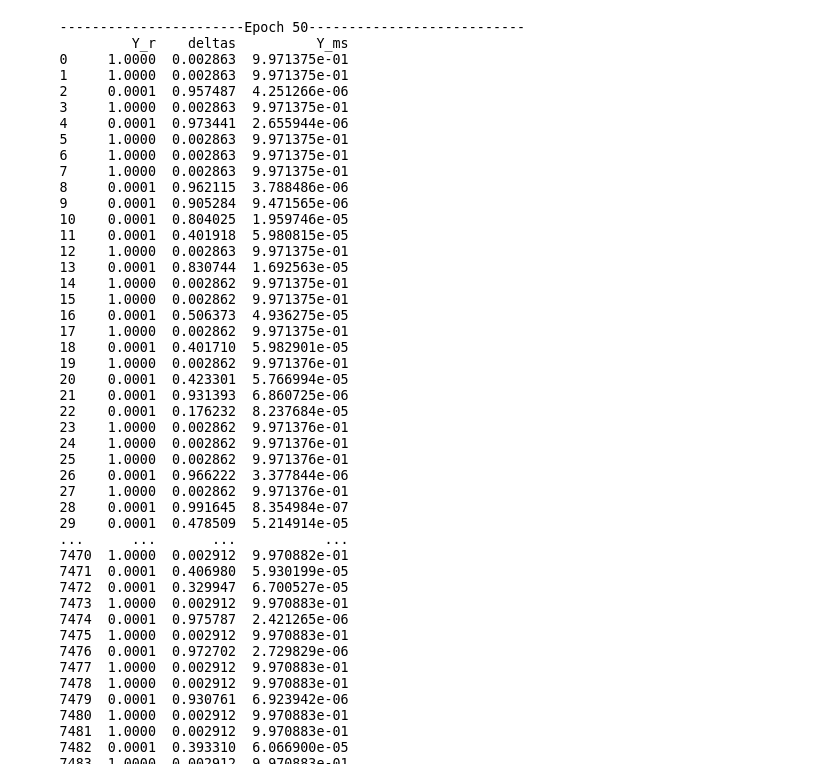
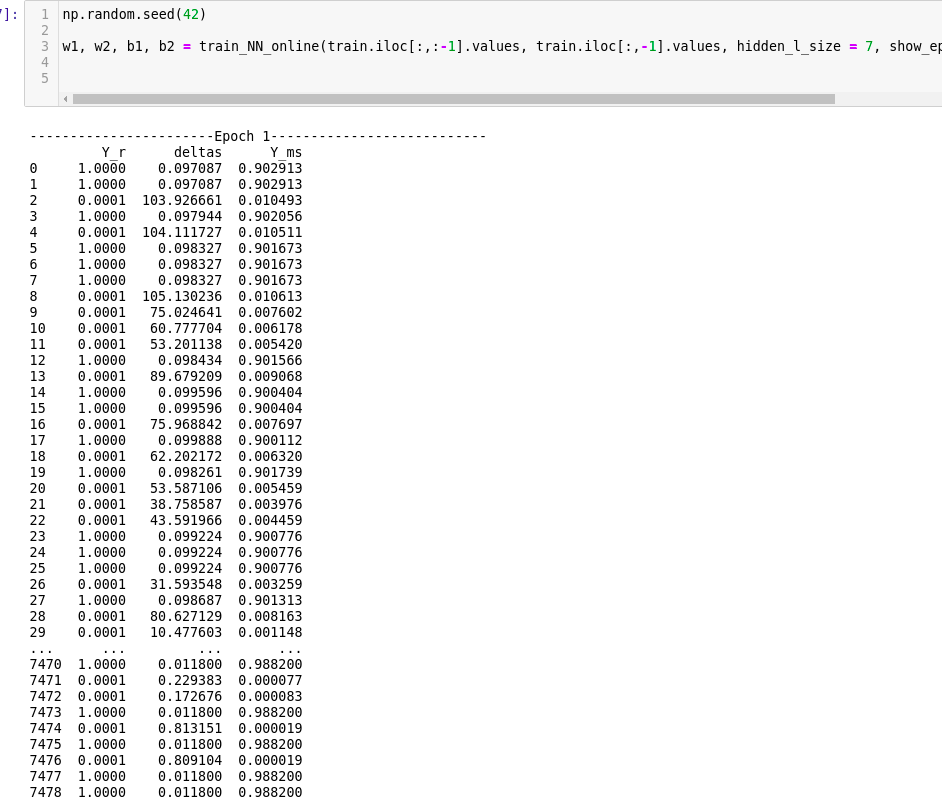
Рисунок 4

## Навчання мережі. Розпізнавання.

Отже, дані готові. Наступний етап – створення перцептрону. За основу було взято мережу з третьої частини лабораторної робот №1 та трохи модифіковано до того, розмірності вхідного та прикованого шару динамічні: перший – створюється автоматично, згідно розміру переданих даних, а другий – передається, як параметр при створенні мережі. Вихідний шар завжди розмірності 1.

Запустимо процес навчання мережі. Результати та час виконання приведений на рисунку 5.





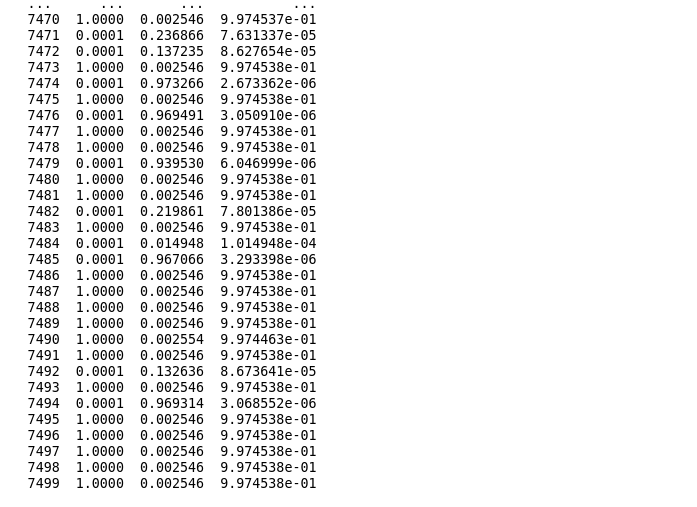


Рисунок 5

Як вино, мережа навчалась протягом 9670,74 секунд, тобто 2 години 42,3 хвилини. Результати схожі на правду, бо була ціль отримати вихідні дані, близько нуля.

І, нарешті останній крок – перевірка модулі на тестових даних. Для цього Випадку доречно використовувати confusion матрицю, яка наочно показує, скільки випадків було детерміновано правильно, а скільки – ні(рис.6):

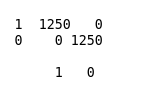


Рисунок 6

Отже, як видно з матриці – 1250 прикладів були віднесені до нормального стану системи в той час, як це було правильним рішенням. В свою чергу 1250 випадки були знайдені небезпечними, що відповідає дійсності. Як видно по нульовим елементам матриці на позиціях (1, 2) та (2, 1) не було жодного хибного твердження.

Повний код програми приведений в додатку А.

Відомості про ноутбук на якому виконувалась робота.

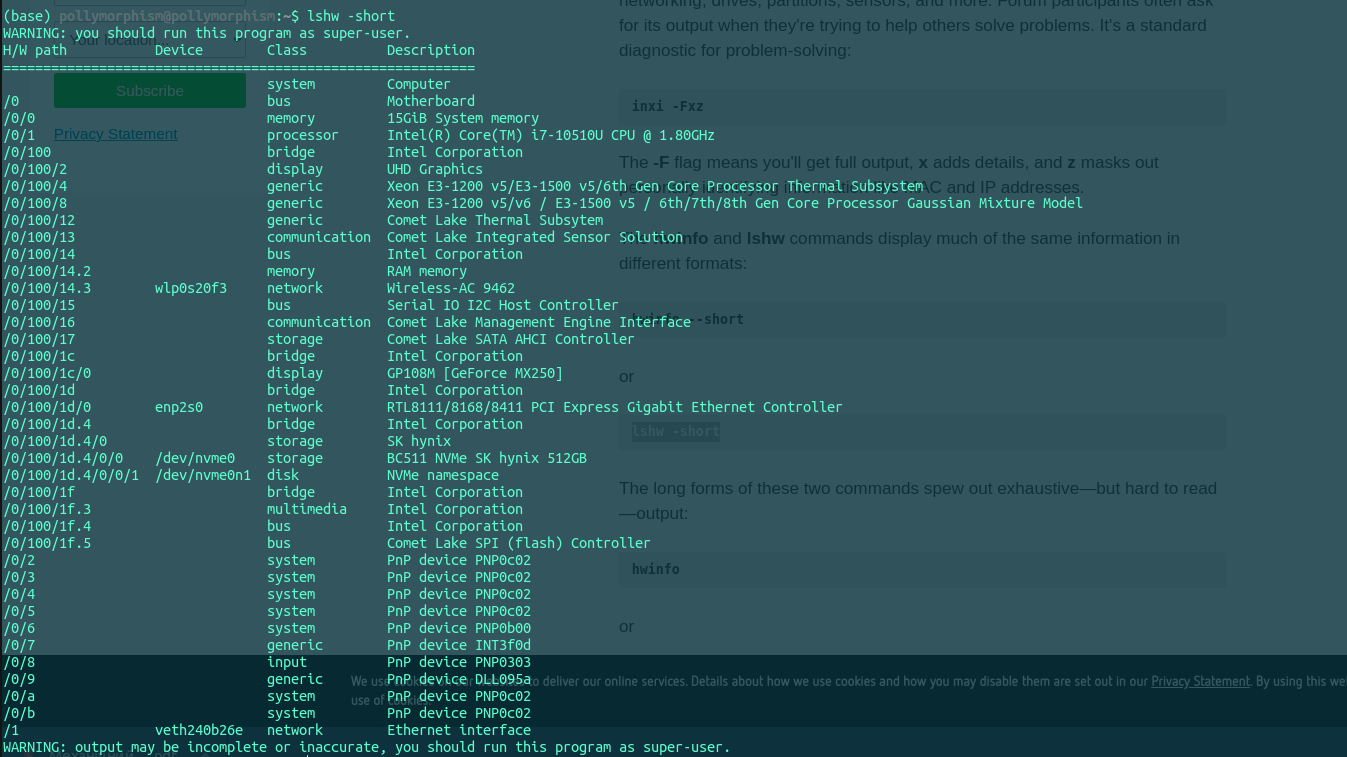


Рисунок 8

# ВИСНОВКИ

В ході даної роботи були напрацьовані навички по попередній обробці даних, їх нормалізації та в цілому роботи за ними. Даний алгоритм не є найбільш оптимізованим, що і показує час його виконання.

В результаті роботи було реалізовано два режими роботи мережі – навчання та розпізнавання, які в кінці вірно розпізнали надані тестові дані без єдиної помилки.

# ДОТАТОК А

#!/usr/bin/env python

# coding: utf-8

# In[10]:

import pandas as pd

import numpy as np

import time

pd.set\_option("max\_columns", None)

df = pd.read\_csv('kddcup99.csv')

df.drop(df[ (df.label != 'normal') & (df.label != 'smurf') ].index, inplace=True)

df = df.iloc[:5000,:].append(df[df['label'] == 'smurf'].iloc[:5000,:], ignore\_index=True)

# In[11]:

df = pd.concat([pd.get\_dummies(df['protocol\_type'], prefix='protocol\_type'), df], axis=1)

df.drop('protocol\_type', axis=1, inplace=True)

df = pd.concat([pd.get\_dummies(df['service'], prefix='service'), df], axis=1)

df.drop('service', axis=1, inplace=True)

df = pd.concat([pd.get\_dummies(df['flag'], prefix='flag'), df], axis=1)

df.drop('flag', axis=1, inplace=True)

df['label'] = np.where(df['label'] == 'smurf', 1, 0.0001)

# In[12]:

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

scaler.fit(df)

df = pd.DataFrame(data=scaler.transform(df), columns=df.columns)

df['label'] = np.where(df['label'] == 0, 0.0001, 1)

# In[13]:

df['label']

# In[14]:

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

train, test = train\_test\_split(df, test\_size=0.25, stratify=df['label'])

# In[15]:

print(train['label'].sum())

print(test['label'].sum())

# In[16]:

# In[27]:

def sigmoid(x):

return 1 / (1 + np.exp(-x))

def initialize\_parameters(n\_x, n\_h, n\_y):

np.random.seed(42)

W1 = np.random.randn(n\_h, n\_x)\*0.01

b1 = np.random.randn(n\_h, 1)\*0.01

W2 = np.random.randn(n\_y, n\_h)\*0.01

b2 = np.random.randn(n\_y, 1)\*0.01

return W1, b1, W2, b2

def train\_NN(x, y\_r, DELTA, b1, b2, w1, w2, rand\_state=42):

np.random.seed(42)

for i in range(1000):

y2 = sigmoid(w1.dot(x) + b1)

y\_m = float(sigmoid(w2.dot(y2) + b2))

delta = float( np.absolute((y\_r - y\_m)/y\_r))

if delta <= DELTA:

return w1, w2, b1, b2, delta, y\_m

else:

q3 = float( y\_m \* (1 - y\_m) \* (y\_r - y\_m) )

d2 = q3 \* y2.reshape(1,-1)

w2 = w2 + d2

q2 = y2 \* (1 - y2) \* (q3 \* w2.reshape(-1, 1))

d = q2 \* x.reshape(1, -1)

w1 = w1 + d

return w1, w2, b1, b2, delta, y\_m

return output

def train\_NN\_online(X, Y, DELTA = 0.1,

hidden\_l\_size = 3,

show\_epochs\_stat=False

):

all\_time = time.time()

W1, b1, W2, b2 = initialize\_parameters(X.shape[1], hidden\_l\_size, 1)

for epoch in range(1, 101):

deltas, y\_ms = [], []

for x, y in zip(X, Y):

W1, W2, b1, b2, delta, y\_m = train\_NN(x.reshape(-1, 1),

y,

DELTA,

b1=b1, b2=b2,

w1=W1, w2=W2

)

deltas.append(delta)

y\_ms.append(y\_m)

if ((epoch == 1) or (epoch % 50 == 0)) and show\_epochs\_stat:

print(f'\n-----------------------Epoch {epoch}---------------------------')

print(pd.DataFrame({

'Y\_r':Y,

'deltas': deltas,

'Y\_ms':y\_ms

}))

print('----------------------------------------------------------')

print(f'counted for {time.time() - all\_time}')

return W1, W2, b1, b2

def predict(x, w1, w2, b1, b2):

y1 = sigmoid(w1.dot(x) + b1)

output = sigmoid(w2.dot(y1) + b2)

return float(output)

def predict\_online(X, w1, w2, b1, b2):

y\_pred = []

for x in X:

y\_pred.append( predict(x.reshape(-1, 1), w1, w2, b1, b2) )

return y\_pred

# In[17]:

np.random.seed(42)

w1, w2, b1, b2 = train\_NN\_online(train.iloc[:,:-1].values, train.iloc[:,-1].values, hidden\_l\_size = 7, show\_epochs\_stat=True)

# In[18]:

pred = predict\_online(test.iloc[:,:-1].values, w1, w2, b1, b2)

res = pd.DataFrame(data={

'real': test.iloc[:,-1].values,

'pred': pred

})

res[res['pred'] > 1e-2]

# In[19]:

from sklearn.metrics import confusion\_matrix

res['pred'] = np.where(res['pred'] > 1e-2, 1, 0)

res['real'] = np.where(res['pred'] > 1e-2, 1, 0)

a = confusion\_matrix(res['real'], res['pred'])

out = f"""

1 {a[0][0]: >3} {a[0][1]: >3}

0 {a[1][0]: >3} {a[1][1]: >3}

1 0

"""

print(out)

# In[ ]: