Data\_loader.py

**Імпорт бібліотек**

import pandas as pd

* Імпортується бібліотека pandas для роботи з табличними даними (наприклад, файлами .csv).
* pd — це скорочення (аліас) для швидшого виклику функцій із бібліотеки.

from src.config import DATA\_PATH

* Імпортується змінна DATA\_PATH з файлу config.py, який знаходиться в папці src.
* Змінна DATA\_PATH містить шлях до файлу з даними.

Оголошення функції

def load\_data(filepath=DATA\_PATH):

df = pd.read\_csv(filepath)

return df

* Оголошується функція load\_data.
* Параметр filepath приймає шлях до файлу. За замовчуванням він дорівнює DATA\_PATH.
* Всередині функції:
  + pd.read\_csv(filepath) відкриває та зчитує вміст CSV-файлу у змінну df (DataFrame — табличний формат даних).
  + return df повертає таблицю з даними.

**Data\_cleaner.py**

## Імпорт бібліотеки

import pandas as pd

* Імпортується бібліотека **pandas**, яка використовується для роботи з табличними даними.

## Оголошення функції

def clean\_data(df, numeric\_columns):

df = df.dropna()

for col in numeric\_columns:

df[col] = pd.to\_numeric(df[col], errors='coerce')

df = df.dropna()

return df

* Оголошується функція **clean\_data**.
* Параметри функції:
  + df — таблиця даних (DataFrame), яку потрібно очистити.
  + numeric\_columns — список назв колонок, які повинні бути числовими.

### Що виконує функція:

1. df = df.dropna()
   * Видаляє всі рядки з пропущеними значеннями (NaN) у будь-яких стовпцях.
2. for col in numeric\_columns:
   * Проходить циклом по кожному стовпцю, який має бути числовим.
3. df[col] = pd.to\_numeric(df[col], errors='coerce')
   * Пробує перетворити дані у стовпці в числовий тип.
   * Якщо дані не можна перетворити (наприклад, текст замість числа), вони замінюються на NaN (errors='coerce').
4. df = df.dropna()
   * Після перетворення ще раз видаляються всі рядки з пропущеними значеннями.
5. return df
   * Функція повертає очищену таблицю.

## Загальна суть

* Ця функція:
  + Видаляє пропущені дані.
  + Переводить вказані стовпці у числовий формат.
  + Ще раз видаляє рядки, де перетворення в числовий тип не вдалося.
* Результат — повністю очищена таблиця з правильними типами даних.

**Dataset.py**

# Пояснення коду

## Імпорт бібліотек

import pandas as pd

import numpy as np

import random

from datetime import datetime, timedelta

* Імпортуються бібліотеки для роботи:
  + pandas — обробка таблиць.
  + numpy — числові обчислення.
  + random — генерація випадкових чисел.
  + datetime, timedelta — робота з датами та часом.

## Встановлення генераторів випадкових чисел

np.random.seed(42)

random.seed(82)

* Встановлюються фіксовані **seed** значення для відтворюваності результатів генерації випадкових чисел.

## Основні параметри

n\_rows = 230000

* Задається кількість рядків у майбутньому датасеті: **230 000**.

## Визначення можливих категорій

protocols = [...]

services = [...]

states = [...]

* Оголошуються можливі значення для колонок:
  + **protocols** — мережеві протоколи (TCP, UDP тощо).
  + **services** — сервіси або служби (HTTP, DNS, FTP тощо).
  + **states** — стани з'єднань.

## Генерація часових міток

base\_time = datetime.now() - timedelta(days=30)

start\_times = [...]

* Генерується базова дата — початок місяця тому від поточного моменту.
* Створюються випадкові дати і час початку з'єднань для кожного рядка датасету.

## Створення основного датасету

df = pd.DataFrame({...})

* Створюється таблиця **DataFrame**, де кожна колонка заповнюється:
  + Випадковими категоріальними значеннями (proto, service, state).
  + Випадковими або експоненційно розподіленими числовими даними (dur, rate, sload, sinpkt тощо).
  + Логічним прапорцем атаки is\_attack (0 — нормальний трафік, 1 — атака) з імовірністю 10%.

## Додавання часових міток завершення

df['start\_time'] = start\_times

df['end\_time'] = [st + timedelta(seconds=dur) for st, dur in zip(start\_times, df['dur'])]

* Додаються стовпці start\_time і end\_time для кожного з'єднання.

## Округлення числових значень

for col in [...]:

df[col] = df[col].round(6)

* Округлюються дробові числові колонки до 6 знаків після коми для підвищення точності й реалістичності даних.

## Збереження датасету

output\_path = "data/dataset1.csv"

df.to\_csv(output\_path, index=False)

* Таблиця зберігається у файл dataset1.csv у папку data, без індексації рядків.

## Виведення результату

print(...)

* Виводиться інформація про успішне створення датасету і список усіх колонок.