Тімошин Анатолій Сергійович,
доцент кафедри організації правоохоронних та судових органів Луганського державного університету внутрішніх справ імені Е. О. Дідоренка, кандидат фізико-математичних наук, доцент

ТЕХНОЛОГІЇ ОБРОБКИ ВЕЛИКИХ ДАНИХ У КРИМІНАЛЬНОМУ АНАЛІЗІ

Застосування інформаційних технологій в кримінальному аналізі є невід'ємною частиною аналітичної роботи і передбачає досить широкий спектр задач і інструментів. На даний час є певне уявлення про те, який інструментарій може бути доцільним при вирішенні задач на тому або іншому рівні кримінального аналізу [3, 4, 5]. Між тим, ефективність використання цього інструментарію залежить від меж його можливостей. Зараз дедалі більше говорять про штучний інтелект, нейронні мережі, Big Data. Великі дані, як структуровані масиви даних великого обсягу, обробляються за допомогою спеціальних автоматизованих інструментів для статистики, аналізу, прогнозів та прийняття рішень. Отже, сучасні аналітичні техніки та методи спираються на більш передові інформаційні технології, такі, наприклад, як мова програмування Python або мова R. У свою чергу, застосування більш потужних інструментів аналізу даних змінює зміст задач і завдань, надає більшої можливості для статистичних досліджень, які дозволять простежити ті глибинні процесі в суспільстві, які впливають на масштаб і характер злочинності. Все це обумовлює актуальність дослідження застосування передових технологій.

Почнемо з того, що великий обсяг інформації, який підлягає аналізу, представлено текстовими файлами з структурованими даними. Це типи даних, які мають внутрішню структуру та можуть бути сконструйовані з простих типів даних. Структурованими типами даних можуть бути масиви (рядки, стовпці, матриці), множини (набори неупорядкованих унікальних значень), і, навіть, файли.

Прикладом структурованих даних є табличні формати. Текстовий табличний формат – це структура даних, яка представлена рядками з окремих полів даних (значень), які розділені спеціальними символами (delimiter), зазвичай комами, або, кома з крапкою. До того ж, треба мати опис таблиці – назви та формат полів. Табличні формати як правило зберігаються в текстових файлах txt, або csv. Різниця між цими файлами (якщо там дані структуровані) незначна. Взагалі, можна вважати, що це «чисті» дані, тобто у подальшому їх аналізі в якихось додатках не виникне зайвих труднощів стосовно втрати інформації або їх спотворення.

Між тим, навіть Excel, відкриваючи csv-файл може некоректно «зрозуміти» зміст даних. Більшість казусів пов'язано з тим, що за

замовчуванням рядки з набором цифр перетворюються в числа. Наприклад, якщо у одному полі записані два номера телефону, то Excel перетворить ці номера в одне число. Довгий набір цифр перетворюється до експоненційної форми, що зовсім змінює зміст даних. «Лідируючі» плюси та нулі цілком видаляються. Знак плюс, на початку рядка з цифрами, Excel сприймає як ознаку додатного числа і теж відкидає його.

Щоб уникнути схожих моментів, в Excel передбачено майстер імпорту даних (вкладка Дані > Отримання зовнішніх даних). Для прикладу візьмемо структурований текстовий файл банківських транзакцій, який містить поля Джерело і Призначення в якості номерів рахунків (набір цифр), Дата Час (дата, час), Сума коштів (набір цифр). Після того, як майстер імпорту визначиться зі структурою даних (з роздільниками або фіксованої ширини) та типом символу-роздільника (знак табуляції, крапка з комою, кома, пробіл, інше), потрібно правильно вказати формат або тип даних для кожного поля. Майстер підказує, що загальний формат є найбільш універсальним, тобто числові значення перетворюються в числа, дати – в дати, все інше в текст. У нашому прикладі 1-е та 2-е поле містять цифри, і якщо їх формат залишити за замовчанням як загальний, то в результаті імпорту Excel відкине нуль, який може стояти першим в номері рахунку. Тим самим, ми втрачаємо частину даних. Отже, формат цих полів вибирається текстовим. Це стосується також номерів телефонів, номерів банківських карток, номерів будинків, паролів (коли вони складаються тільки з цифр) і т.п.

Якщо дані імпортуються в Excel правильно і без втрати, то ми отримаємо заповнений діапазон комірок з однотипними рядками і заголовком. І, подальшу роботу з аналізу даних краще продовжити, перетворивши діапазон в таблицю (вкладка Вставлення > Таблиця). В результаті виконання цієї команди діапазон з даними прийме вид з характерним стилем та кнопками фільтру. Крім того, з'явиться вкладка Конструктор таблиць. На вкладці Конструктор таблиць можна активувати рядок підсумків, який фактично працює з функцією SUBTOTAL. На прикладі таблиці банківських транзакцій, шляхом фільтрації та використання рядка підсумків, можна вирішити багато питань. Наприклад, це – загальна кількість транзакцій або загальна сума коштів для всіх транзакцій, кількість транзакцій з певного рахунку (або певних рахунків), сума коштів для транзакцій з певного рахунку (або певних рахунків), кількість транзакцій або сума коштів за конкретну дату (або певний термін), максимальний (мінімальний) переказ і т.п.

З вкладки Конструктор можна звернутися до команди Підсумувати у зведеній таблиці. Стосовно таблиці банківських транзакцій, за допомогою зведених таблиць [2] можна вирішити такі задачі, як визначення рахунку, з якого (на який) надійшла найбільша

(найменша) сума коштів, одночасно проглянути кількість транзакцій з кожного рахунку і т.п.

Звернемо увагу на те, що поки мова йшла про аналіз лише однієї таблиці, яка розташована на одному листі книги Excel. До то ж, не підкреслювався обсяг інформації, точніше кількість записів в таблиці. Для інших прикладів таблиць полів може бути також значно більше. Все це може сильно ускладнювати аналіз в Excel. Отже, потрібні інструменти більш ефективні та універсальні. В цьому сенсі, ми вже маємо рішення – це інтерпретатор Python [1].

Перед тим, як починати аналіз даних в Python, потрібно також підготувати ці дані. Підготовка даних потребує дотримання певних умов, які представляють собою так званий бест-практикс табличних даних. Наприклад, табличні дані в файлі xlsx повинні задовольняти таким умовам – перший рядок таблиці треба зарезервувати під заголовок; уникати пробіли, замість пробілу краще використовувати знак підкреслення або «горбатий» регістр; віддавати перевагу коротким назвам; уникати використання різних дужок та символів; видалити будь-які коментарі.

Перелічені вище задачі для таблиці банківських транзакцій, які вирішувалися за допомогою рядка підсумків та зведених таблиць, можна вирішити в Python. При цьому, розмір таблиці, кількість полів, або кількість таблиць (на різних листах книги) стає менш важливим при аналізі даних.

Для роботи в Python з даними в Excel користуються бібліотекою openpyex. На початку, звертаються до файлу Excel за допомогою команди:

wb = openpyxl.open («banktransactions.xlsx»).

Тим самим, ми створюємо об’єкт-файл (або, об’єкт-книгу) wb, з яким і працює в подальшому Python.

Бібліотека openpyex дозволяє працювати з листами книги, з комірками листів, з даними в комірках, з набором рядків таблиці, перетворюючи їх в списки (як тип даних в Python). Також, openpyex дозволяє відшукувати в книзі необхідні листи, будувати нові книги, створювати листи в нових книгах та зберігати ці книги у файлах на вашому пристрої. На основі списків створюються певні словники (набір даних, в якому кожному значенню відповідає ключ). Словники допомагають виконувати необхідні сортування та фільтрацію записів.

Розглянемо більш детально техніку аналізу даних табличного формату в Python за допомогою бібліотеки openpyex на прикладі таблиці банківських транзакцій.

Якщо книга wb містить декілька листів, то починаємо з пошуку листа (точніше, його номера), який містить потрібну таблицю. Як правило, пошук може бути або за ім’ям листа, або за вмістом певної комірки. Список листів можна отримати за допомогою атрибуту об’єкта-книги, а саме це – wb.sheetnames. Вміст комірки (наприклад, A1) може бути отримано за допомогою конструкції wb[i][‘A1’].value.

Ім’я листа під номером «і» отримуємо так: wb[i].title. За допомогою оператора циклу (проходячи елементи списку листів) та умовного оператора знаходимо номер потрібного листа (нехай, це «і»). Нарешті, створюємо об’єкт-лист: ws = wb[i].

Зауважимо, що кількість записів таблиці, як правило, значно перевищує кількість унікальних рахунків (відправників, або отримувачів). Отже, наступний крок, це створення словника, який би зібрав окремо для кожного рахунку-відправника (це буде ключ) всі рахунки-отримувачі, дату і суму перерахованих коштів (це буде значенням ключа). Для початку створюється порожній словник: pryznachdata = {}. Далі, в циклі проходимо усі рядки таблиці, починаючи з другого (перший рядок зарезервовано під заголовок), і для кожного унікального рахунку-відправника (змінна pryz) формуємо список з відповідних йому рядків:

for row in ws.iter_rows(min_row=2, min_col=1, max_row=ws.max_row, max_col = ws.max_column):

…pryz = row[0].value

…pryzdata = [cell.value for cell in row]

……if pryz not in pryznachdata:

……pryznachdata[pryz] = []

……pryznachdata[priz].append(pryzdata)

Отриманий словник можна умовно записати так:

{rachunok_1: [[row_11], …, [row_1k1]], …, rachunok_m: [[row_m1], …, [row_mkm]]},

де m – число унікальних рахунків-відправників.

Якщо ми намагаємося знайти, з якого рахунку надійшла найбільша (найменша) сума коштів, то для вирішення цієї задачі створюється словник на основі словника pryznachdata. А саме,

dicpryzsum = {}

for pryz in pryznachdata:

…dicpryzsum[pryz] = 0

…for i in range(len(pryznachdata[pryz])):

……..dicprizsum[pryz] = dicprizsum[pryz] + pryznachdata[pryz][i] [3]

…print («Відправник –», pryz, «перерахував»:, dicpryzsum[pryz])

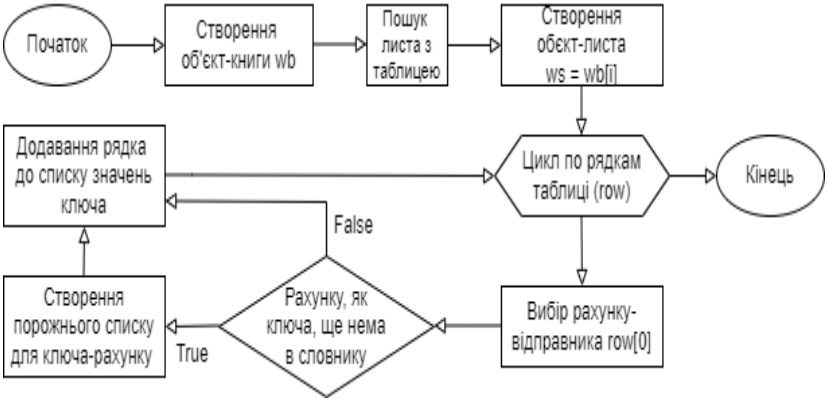
Зовнішній цикл «пробігає» ключі основного словника pryznachdata, а внутрішній – по кожному рядку значення ключа, звертаючись тільки до четвертого поля з перерахованими сумами коштів і кожного разу накопичуючи цю суму. Елементи рядка в Python рахуються з 0, тому звертаємось до поля pryznachdata[priz][i] [3].

В результаті виконання приведеного блоку буде створено словник, який умовно можна записати так: {rachunok_1: suma_1, …, rachunok_m: suma_m}. Для більшої наочності можна створити інший словник sortSuma, отриманий шляхом сортування словника dicpryzsum по значенню:

sortSuma = dict(sorted(dicpryzsum.items(), reverse = True, key=lambda x: x[1])).

В словнику sortSuma елементи будуть розташовані в порядку спадання сум.

Нарешті, приведемо загальну схему початку роботи з таблицею та побудови основного словника:



Треба відмітити, що структура таблиці трафіку з'єднань схожа на структуру таблиці банківських транзакцій. І, тому, схема побудови основного словника практично повністю може бути застосована для аналізу трафіку з'єднань.

Викладена технологія роботи в Python з великими таблицями може бути успішно застосована правоохоронцями-аналітиками для аналізу різних структурованих даних, вирішуючи задачі порівняння кількісних характеристик об'єктів.

Список використаних джерел

1. Костюченко А.О. Основи програмування мовою Python: навчальний посібник. Ч.: ФОП Баликіна С.М., 2020. 180 с

2. Нелюбов В. О., Куруца О. С. Основи інформатики. Microsoft Excel 2016: навчальний посібник. Ужгород : ДВНЗ «УжНУ», 2018. 58 с.

3. Основи кримінального аналізу : посіб. з елементами тренінгу / Користін О. Є., С. В. Албул, А. В. Холостенко та ін. – Одеса : ОДУВС, 2016. 112 с

4. Тактичний кримінальний аналіз: теорія та практика; навчальний посібник / О.Є. Користін, Н.П. Свиридюк, О.М. Цільмак, О.М. Заєць, К.Ю. Ісмайлов, В.А. Некрасов; МВС України, ДНДІ, ОДУВС. Одеса : РВВ ОДУВС, 2019. 216 с

5. Федчак І. А. Основи кримінального аналізу : навчальний посібник. Львів : Львівський державний університет внутрішніх справ, 2021. 288 с