КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота

на здобуття освітнього рівня бакалавра

за спеціальністю 121 Інженерія програмного забезпечення

на тему:

ВЕБ-СКРАПІНГ, АГРЕГАЦІЯ ТА АНАЛІЗ ДАНИХ З САЙТІВ БУКМЕКЕРІВ

Виконав студент 4-го курсу
Ярослав ГАЙДАЙ _____

Науковий керівник:
асистент, кандидат фіз.-мат. наук
Костянтин ЖЕРЕБ _____

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____

Роботу розглянуто і допущено до захисту на засіданні кафедри інтелектуальних програмних систем
« 28 » травня 2021р.,
Протокол № _____

Завідувач кафедри
Олександр ПРОВОТАР _____

Київ – 2021

РЕФЕРАТ

Обсяг роботи: 51 сторінка, 11 ілюстрацій, 3 таблиці, 26 джерел посилань.

АГРЕГАЦІЯ ДАНИХ, АРБІТРАЖНА СТАВКА, АРБІТРАЖНА СТАВКА МАКСИМАЛЬНОГО ПРИБУТКУ, БАНК СТАВКИ, БУКМЕКЕРСЬКА КОМПАНІЯ, ВЕБ-ПАРСИНГ, КОЕФІЦІЄНТ СТАВКИ, МАРЖА, ОПТИМІЗАЦІЯ ЧАСОВИХ ПОКАЗНИКІВ, РЕЗУЛЬТАТ ПОДІЇ, ТЕЛЕГРАМ БОТ, УНІФІКОВАНИЙ СИНТАКСИС.

Об’єктом розроблення програмного засобу є процеси веб-парсингу, агрегації даних та максимізації прибутку клієнтів букмекерських контор. Предметом роботи є програмний засіб для розв’язування задач парсингу, агрегації та максимізації прибутку.

Метою роботи є створення програмного забезпечення для розв’язання задач парсингу, агрегації та максимізації прибутку букмекерських контор та їх клієнтів.

Методи розроблення: веб-парсинг, методи розрахунку прибутку, патерни проектування, об’єктно-орієнoване програмування. Інструменти розроблення: середовище розробки PyCharm IDE Professional Edition 2021.1.1, мова програмування Python, Telegram Bot API, хмарна платформа Heroku, веб-браузер Google Chrome, база даних MySQL, бібліотеки BeautifulSoup, xlsxwriter, фреймворк Selenium.

Результати роботи: виконано загальний огляд роботи букмекерів, побудовано математичну модель, що дозволяє точно встановлювати значення прибутку при відомих коефіцієнтах ставок, розроблено програмний продукт, що використовує вищезгадану модель до даних отриманих за допомогою веб-парсингу та виводить проаналізовану інформацію через телеграм бота, реалізовано власний алгоритм агрегації даних, проведено часову оптимізацію процесів веб-парсингу.

Результати роботи рекомендовано використовувати для отримання максимального прибутку як гравцям, так і букмекерським компаніям.

Сферою застосування є гральний бізнес.

Пропозиції щодо розвитку об’єкта розроблення: пропонується розширення існуючих компонентів шляхом додавання нових дисциплін та букмекерських компаній, створення веб-сервісу для можливості розповсюдження результатів роботи системи через API, також можливе створення веб-додатку, що надавав би альтернативний до телеграм бота інтерфейс користувача для взаємодії з продуктом.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

API – Application Programming Interface;

GIL – Global Interpreter Lock;

IDE – Integrated Development Environment, інтегроване середовище розробки;

АС – арбітражна ставка;

АСМП – арбітражна ставка максимального прибутку.

ВСТУП

Оцінка сучасного стану об’єкта розробки. Інформатизація світу сприяє якнайактивнішому застосуванню веб-парсингу з метою пошуку даних та їх обробки. Це стосується будь-якої компанії, що продає певний продукт на ринку. Відстежуючи ціни на даний товар у конкурентів, така компанія має змогу корегувати свою ціну відповідним чином, створюючи постійну конкуренцію [1].

Іншим застосуванням веб-парсингу на поточний момент є його використання з метою збору інформації для різноманітних баз даних, які, наприклад, можуть використовуватись в системах штучного інтелекту, нейронних мережах, або просто для розміщення даних в зручному та загальнодоступному вигляді.

Агрегація даних багато де йде поруч з веб-парсингом, оскільки при зборі даних з різних джерел часто необхідно згрупувати їх за певними ознаками для подальшого дослідження. Зокрема, агрегація використовується, наприклад, для порівняння цін на будь-які товари або послуги.

Актуальність роботи та підстави для її виконання. При неперервному розширенні ринку послуг букмекерських контор та зростанню їх кількості, проблема парсингу коефіцієнтів, які надають букмекери не тільки залишається ключовою, але й набуває ще більшої ваги в цій галузі.

Так, букмекери мають не тільки постійно відстежувати коефіцієнти один одного, щоб залишатись конкурентноздатними та максимізувати свій прибуток, але й слідкувати за тим, щоб величини, які пропонуються не виходили за межі діапазону отримання вигоди. Тому актуальним є створення засобів ефективного моніторингу та порівняння відповідних значень.

Мета й завдання роботи. Метою роботи є створення програмного засобу для розв’язання задачі парсингу та підтримки актуальності

коефіцієнтів, що пропонуються букмекерами. Для досягнення цієї мети поставлено такі завдання:

– Створити математичну модель, що дозволятиме встановити актуальність та прибутковість коефіцієнтів;

– Дослідити існуючі засоби (аналоги);

– Розробити ефективні алгоритми веб-скрапінгу та агрегації даних;

– Розробити програмне забезпечення, за допомогою якого можна буде дістати необхідну інформацію з сайтів букмекерів;

– Розробити програмне забезпечення для уніфікації отриманих даних, їх співставлення та порівняння;

– Розробити та розгорнути телеграм бота для виводу результатів обробки інформації.

Об’єкт, методи та засоби розроблення. Об’єктом розроблення програмного засобу є процеси веб-парсингу, агрегації даних та максимізації прибутку клієнтів букмекерських контор.

Для створення запропонованого продукту, необхідним є аналіз функціонування букмекерських контор та створення відповідних математичних моделей. А також дослідження існуючих засобів для скрапінгу.

В якості інструменту створення програмного забезпечення було використано PyCharm IDE Professional Edition 2021.1.1 – інтегроване середовище розробки мовою програмування Python. Серед інших засобів розробки: бібліотека BeautifulSoup [2], фреймворк Selenium [3], бібліотека pyTelegramBotAPI [4], база даних MySQL [5], бібліотека xlsxwriter [6] та хмарна платформа Heroku [7].

Можливі сфери застосування. Вихідний програмний продукт може бути застосованим в процесі встановлення коефіцієнтів букмекерської

контори, або ж для аналізу та виявлення найвигідніших варіантів здійснення ставок на ринку.

Взаємозв’язок з іншими роботами. Робота є розширенням власної курсової роботи.

1 ІСТОРІЯ ВИНИКНЕННЯ ТА РОЗВИТКУ БУКМЕКЕРСЬКИХ КОНТОР [8] ТА ПАРСИНГ

1.1 Передумови виникнення букмекерських контор.

Бажання брати участь в суперечках є корінною причиною виникнення спеціальних структур розважального характеру. Біля витоків зародження людства стоїть прагнення здобути фінансову винагороду шляхом вирішення спірних питань. Структурні одиниці, які займаються організацією прийому ставок на різні культурні заходи, називають в сучасному суспільстві букмекерськими конторами. Такі компанії організовують публічні торги, наприклад в спортивних матчах (футбол, баскетбол, хокей). Предметами ставок можуть бути культурні і політичні події. Робота букмекерських контор заснована на аналізі результату заходу і виставленні певного коефіцієнта. Заробляють такі компанії на ставках своїх клієнтів, а прибуток вже закладений в самому коефіцієнті ставки.

1.2 Перші букмекери.

Історія становлення букмекерських контор, як виду бізнесу, почалася ще за часів Римської Імперії. Популярними розвагами того часу були бої за участі гладіаторів та перегони на іподромі (Рис. 1). Саме з появою такого виду розваг і виникли перші ставки на результат поєдинків. Спочатку вони мали словесний характер. Виграшем служили матеріальні цінності або інші блага (нерухомість, сім'я, правління державою).

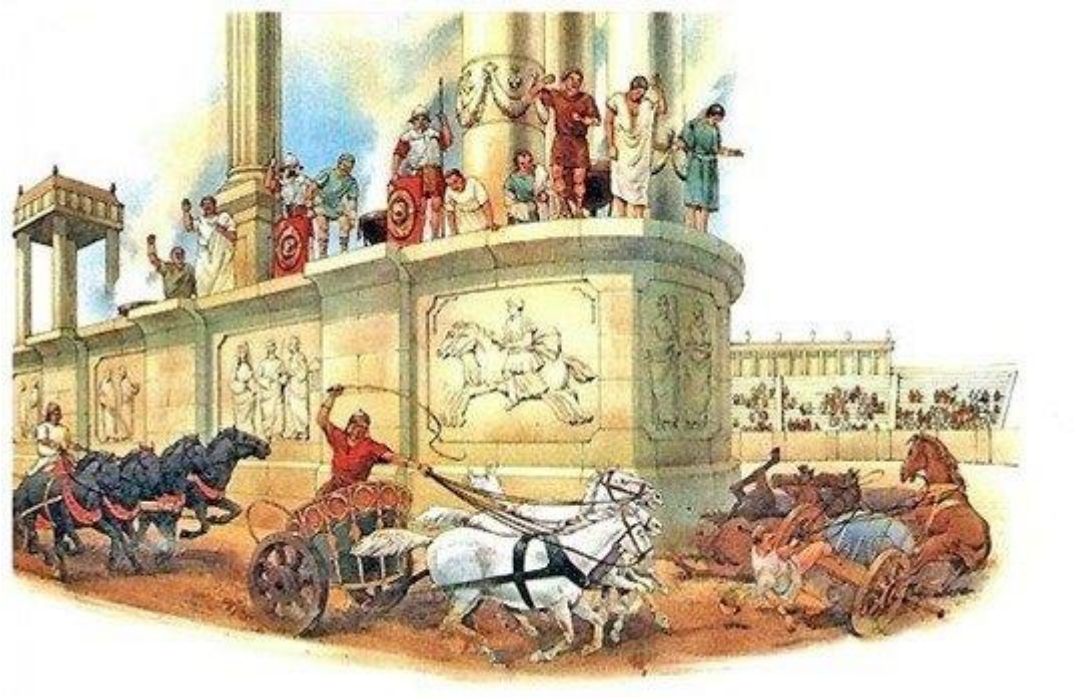

Рисунок 1 – перегони на іподромі в Древньому Римі

Сучасне букмекерство зародилося в XIX столітті, а його основоположником вважається французький крамар П'єр Олер. 1872 року він відкрив перший пункт з прийому ставок на результати кінних перегонів. З того часу основні принципи й правила букмекерської діяльності практично не змінилися. У Британії букмекерство з'явилося в XIX столітті. Ця країна утримує лідерство за числом букмекерських контор і гравців на тоталізаторі.

Широкого розмаху букмекерство набуло у США. У сферу укладення парі на гроші потрапили вже не тільки кінні перегони, але й їх різновид, собачі перегони. В США також винайшли «перегони тарганів».

З появою колективних спортивних ігор, букмекери почали приймати ставки на інші змагання, наприклад, на результати футбольних матчів. У Британії й деяких інших країнах відкрити букмекерську контору та приймати ставки може будь-хто, в інших країнах це право належить державі, подекуди воно трансформувалось в лотереї.

1.3 Букмекерські контори онлайн.

З повсюдним впровадженням глобальної мережі в життя кожної людини, букмекерські контори почали борознити простори інтернету. Розквіт таких контор відбувається в 90-х роках 20 століття. Перша інтернет-ставка сталася в Канаді. Трохи пізніше такі букмекерські компанії завойовують територію Європи. Багатий вибір операторів, систем бонусних нарахувань і предметів публічних суперечок, привертає все більше і більше гравців по всьому світу.

1.4 Наші часи.

Гральний вид бізнесу не втрачає свої лідируючі позиції і сьогодні. Ставки рясніють своїм часом екзотичним характером. Тепер можна брати участь не тільки в спортивних суперечках, а й робити ставки на результат інших подій: вибори президентів, музичні конкурси і навіть існування інопланетних цивілізацій.

Враховуючи пристрасть людей до азартних ігор, букмекери отримують великі прибутки. Так, італійський футбол фактично існує на засоби, виручені від спортивних лотерей (які знаходяться у власності держави), причому італійський чемпіонат – один з «найбагатших» у світі.

1.5 Парсинг та області його застосування.

Парсинг – процес синтаксичного розбору даних. На сьогоднішній день парсинг використовується практично неперервно у багатьох сферах. Серед найголовніших областей його застосування можна виділити синтаксичний розбір вихідного коду в мовах програмування [9], побудова індексів в

пошукових системах [10], машинний переклад [11], генерація текстів [12], регулярні вирази [13], формальні граматики [14] та збір інформації з веб-сторінок (веб-парсинг).

Веб-парсинг набуває все більшої популярності зі зростанням кількості товарів, послуг та компаній, що їх надають. Щоб успішно збувати товар потрібно постійно бути в курсі актуальних цін на схожі товари у конкурентів. Тому буквально кожен інтернет-магазин використовує веб-парсинг для автоматичного корегування цін на свої товари та послуги.

Надалі в рамках цієї роботи вважатимемо слова парсинг, веб-парсинг, скрапінг, веб-скрапінг синонімами під якими матимемо на увазі синтаксичний розбір вихідного коду сторінок сайтів та подальший збір необхідної інформації.

1.6 Дослідження існуючих засобів веб-скрапінгу.

Оскільки для програмної реалізації було обрано мову Python, будемо досліджувати існуючі засоби для веб-скрапінгу лише в контексті даної мови програмування. Результати порівняння таких інструментів наведено у таблиці 1.

Таблиця 1 – порівняння існуючих засобів веб-скрапінгу

Засіб/ Метрика	Scrapy [15]	Requests [16]	BeautifulSoup [2]	Selenium [3]
Бібліотека чи фреймворк?	Фреймворк	Бібліотека	Бібліотека	Фреймворк
Задача, яку вирішує	Повноцінне рішення для	Спрощує відправки	Парсер даних	Моделювання поведінки реального

	веб-скрапінгу	HTTP запитів		користувача у веб-браузері
Доступні селектори	JCSS, Xpath	Немає	CSS	CSS, Xpath
Асинхронність	Так	Ні	Ні	Ні
Підтримка Javascript	Так	Ні	Ні	Так
Документація	Чудова	Чудова	Чудова	Гарна
Складність опанування інструменту	Легко	Дуже легко	Дуже легко	Легко
Ідеальний сценарій використання	Розробка великих повторюваних проектів з веб-скрапінгу	Прості проекти без повторюваних задач	Прості проекти без повторюваних задач	Невеликі проекти зі скрапінгу веб-сайтів, що переважно використовують Javascript

Як видно з таблиці, для невеликих проектів зі скрапінгу сайтів, що динамічно наповнюють свої сторінки за допомогою Javascript, ідеально підходить фреймворк Selenium [3]. Також він дозволяє краще «маскуватись» під звичайного користувача, що є необхідною умовою для уникнення одного з основних ризиків проекту – блокування веб-сайтами. Таким чином, можна стверджувати, що Selenium найкраще підходить для реалізації поставлених в роботі завдань. Однак, варто також зазначити, що при збільшенні кількості

підтримуваних програмною реалізацією букмекерів буде збільшуватись і відсоток виконання повторюваних задач при веб-скрапінгу, тому, можливо, стане доцільним перехід на використання фреймворку Scrapy [15] (або ж комбінація підходів).

2 МАТЕМАТИЧНА МОДЕЛЬ БУКМЕКЕРСЬКИХ КОНТОР

Наведена далі модель є розвитком попередньої моделі, що була самостійно побудована в курсовій роботі. Ця модель є розширенням та узагальненням існуючих аналогічних моделей в тому сенсі, що вона підтримує довільну кількість букмекерів та довільну кількість результатів для ставок.

2.1 Прибуток контори.

Букмекери отримують свій прибуток завдяки комісії, що знімається з кожної ставки. Ця комісія називається маржею та має вигляд зменшеного коефіцієнту. Розглянемо найпростіший приклад. Нехай пропонуються ставки на певну подію **A**: подія **A** відбудеться/подія **A** не відбудеться. Таким чином, наприклад, якщо букмекер вважає шанси однаковими, тобто $\boldsymbol{P(A) = P(\overline{A}) = \frac{1}{2}}$, де **P(A)** – ймовірність події A, то коефіцієнти мають бути відповідно: $\boldsymbol{C(A) = \frac{1}{P(A)} = \frac{1}{P(\overline{A})} = C(\overline{A}) = 2}$. Але на практиці, в такому разі їх значення будуть меншими, наприклад $\boldsymbol{C'(A) = C'(\overline{A}) = 1.87}$. Маржу в даному випадку можна отримати підрахувавши ймовірності $\boldsymbol{P'(A) = P'(\overline{A}) = \frac{1}{C'(A)} = \frac{1}{1.87}}$, та обчисливши їх суму: $\boldsymbol{P'(A) + P'(\overline{A}) = \frac{2}{1.87} = \frac{1.87+0.13}{1.87} = 1.07.}$ Таким чином з кожної ставки на подію **A** букмекер заробить в середньому 7%.

Означення 1 Маржею букмекера для події **A** з **M** результатами називається величина $\boldsymbol{M_A = \sum_{i=1}^{M} \frac{1}{C_i} - 1}$, де $\mathbf{C_1, C_2, \ldots, C_M}$ – коефіцієнти даного букмекера на результати події **A**. При цьому $\mathbf{\{C_1, C_2, \ldots, C_n\}}$ називатимемо коефіцієнтами маржі, а для маржі $\boldsymbol{M_A}$ будемо казати, що вона складається з коефіцієнтів $\mathbf{\{C_1, C_2, \ldots, C_n\}}$.

2.2 Поняття відносної маржі.

Отже, як ми з'ясували в попередньому підрозділі, букмекерські контори мають доходи завдяки додатньому значенню маржі. А що ж станеться, якщо комісія виявиться від'ємною? Відповідь досить очевидна, компанія почне приносити збитки а клієнти отримувати прибутки. В постійній підтримці додатного значення маржі і є суть роботи букмекерів. На практиці, ситуації з від'ємною маржею відсутні або ж настільки не часті, що їх помітити майже неможливо.

Введемо поняття відносної маржі.

Означення 2 Відносною маржею для букмекерських контор **B1, B2** та події **A** назвемо маржу, яка складається з коефіцієнтів $\mathbf{\{C_i(A)\}}$ таких що $\boldsymbol{C_i(A) = max(C_{B1_i}(A), C_{B2_i}(A))}$, де $\{\boldsymbol{C_{B1_i}(A)}\}, \{\boldsymbol{C_{B2_i}(A)}\}$ – коефіцієнти на подію **A** в букмекерів **B1** та **B2** відповідно. Позначатимемо її $\mathbf{M_A(B1, B2)}$. При цьому називатимемо її маржею між **B1** та **B2** відносно **A**.

Твердження 1 (про незалежність марж) Відносна маржа між **B1** та **B2** відносно **A** може бути від'ємною, навіть якщо маржі **B1** та **B2** для **A** окремо строго додатні.

Доведення. Для доведення достатньо навести контр-приклад: $\boldsymbol{C_{B1_1}(A) = 2.15, C_{B1_2}(A) = 1.65, C_{B2_1}(A) = 3.45, C_{B2_2}(A) = 1.28}$; $\boldsymbol{M_A(B1, B2) = \frac{1}{3.45} + \frac{1}{1.65} - 1 = 0.89 - 1 = -0.11 < 0}$

Тобто у випадку відносної маржі, її від'ємність означає можливість гравців заробити, при цьому, за твердженням 1, обидві компанії також можуть отримувати прибуток (з гравців, які користуються послугами однієї з контор окремо).

2.3 Узагальнення поняття відносної маржі.

В попередньому підрозділі ми ввели поняття відносної маржі для двох букмекерських контор. Тепер узагальнимо його на довільну зліченну кількість компаній.

Означення 3 Відносною маржею для множини букмекерських контор $\{\boldsymbol{B}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ та події **A** назвемо маржу, яка складається з коефіцієнтів $\{\mathbf{C_i(A)}\}$ таких що $\boldsymbol{C_i(A) = max(C_{B_{1_i}}(A), C_{B_{2_i}}(A), \dots, C_{B_{N_i}}(A))}$, де $\{\boldsymbol{C_{B_{1_i}}(A)}\}, \{\boldsymbol{C_{B_{2_i}}(A)}\}, \dots,$ $\{\boldsymbol{C_{B_N}(A)}\}$ – коефіцієнти на подію **A** в букмекерів $\{\boldsymbol{B}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ відповідно. Позначатимемо її $\mathbf{M_A(\boldsymbol{B_1, B_2,\dots, B_N})}$. При цьому називатимемо її маржею множини $\{\boldsymbol{B}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ відносно **A**.

Зауважимо також, що Твердження 1 залишається справедливою і для N-вимірного випадку, оскільки для її доведення достатньо в попередньо наведений контр-приклад додати довільні коефіцієнти інших N-2 букмекерів, при цьому максимуми відповідних величин не зменшаться, а отже, і відносна маржа залишиться так само від’ємною.

2.4 Прибуток клієнта.

З підрозділу 2.1 ми дізнались про прибуток контори та навчились його визначати. Тепер розглянемо прибуток клієнтів компанії в разі виграшу.

Прибуток (або чистий прибуток) гравця (клієнта) при виграшній його ставці суми **S** на коефіцієнт **C** дорівнюватиме **S * (C – 1)**. При програшній

ставці вважатимемо, що чистий прибуток дорівнює **-S**. Позначатимемо прибуток **pr**.

2.5 Поняття арбітражної ставки.

Як ми вже з'ясували в підрозділі 2.2, якщо відносна маржа є від'ємною, клієнти, користуючись послугами одразу обох компаній, можуть отримати гарантований прибуток. Проілюструємо це на прикладі, $\boldsymbol{C_{B_{1_1}}(A) = 2.15, C_{B_{1_2}}(A) = 1.65, C_{B_{2_1}}(A) = 3.45, C_{B_{2_2}}(A) = 1.28;\ M_A(B_1, B_2) = -0.11.}$ Як бачимо, маржа менша за нуль, тому грамотно підібравши суми ставок, можемо отримати гарантований прибуток. Наприклад, поставивши 100 умовних одиниць на перший результат події **A** в букмекера $\mathbf{B_2}$ та 200 умовних одиниць на другий результат події **A** в букмекера $\mathbf{B_1}$**,** ми в разі перемоги першого результату отримаємо чистий прибуток рівний **100 * (3.45 – 1) – 200 = 45 (у.о.),** а в разі перемоги другого результату чистий прибуток складе **200 * (1.65 - 1) – 100 = 30 (у.о.).** Таким чином, в будь-якому разі подібні дві ставки виявляться прибутковими.

Введемо поняття арбітражної ставки.

Означення 4.1 Арбітражною ставкою (АС) на подію **A**, що має рівно два взаємовиключних результати на множині букмекерів $\{\boldsymbol{B}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ назвемо множину з 2 ставок, що задовольняє наступній умові: при перемозі будь якого з двох результатів сума чистих прибутків з цих ставок буде строго додатною. Суму чистих прибутків множини ставок АС будемо називати її прибутком (або чистим прибутком).

Означення 4.2 Арбітражною ставкою максимального прибутку (АСМП) на подію **A**, що має рівно два взаємовиключних результати, на множині букмекерів $\{\boldsymbol{B}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ назвемо множину з 2 ставок, що задовольняє

наступній умові: при перемозі будь якого з двох результатів сума чистих прибутків з цих ставок буде **рівною** та строго додатною.

Зауваження. Виходячи з вищевведених означень, множина АС є надмножиною АСМП.

Твердження 2.1 (про величину ставок в АСМП) Для події **A**, що має рівно два взаємовиключних результати, на множині букмекерів $\{\boldsymbol{B}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ величини ставок з множини АСМП задовольняють наступному співвідношенню: $\boldsymbol{x} * \boldsymbol{c_1} = \boldsymbol{y} * \boldsymbol{c_2}$, де **x** – сума ставки на коефіцієнт $\boldsymbol{c_1}$, **y** – сума ставки на коефіцієнт **c2.**

Доведення. За означенням АСМП маємо:

$$\boldsymbol{x} * (\boldsymbol{c_1} - \boldsymbol{1}) - \boldsymbol{y} = \boldsymbol{y} * (\boldsymbol{c_2} - \boldsymbol{1}) - \boldsymbol{x}$$

$$\boldsymbol{x} * (\boldsymbol{c_1} - \boldsymbol{1}) + \boldsymbol{x} = \boldsymbol{y} * (\boldsymbol{c_2} - \boldsymbol{1}) + \boldsymbol{y}$$

$$\boldsymbol{x} * \boldsymbol{c_1} = \boldsymbol{y} * \boldsymbol{c_2}$$

Означення 5 Назвемо банком арбітражної ставки суму величин ставок її множини.

Твердження 2.2 (про величину ставок в АСМП відносно її банку) Для події **A**, що має рівно два взаємовиключних результати, на множині букмекерів $\{\boldsymbol{B}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$, ставки множини АСМП мають наступний розмір: $\boldsymbol{x} = \boldsymbol{t} * \frac{\boldsymbol{c_2}}{\boldsymbol{c_1}+\boldsymbol{c_2}}, \boldsymbol{y} = \boldsymbol{t} * \frac{\boldsymbol{c_1}}{\boldsymbol{c_1}+\boldsymbol{c_2}}$, де **t** – банк АСМП, **x** – сума ставки на коефіцієнт $\boldsymbol{c_1}$, **y** – сума ставки на коефіцієнт $\boldsymbol{c_2}$.

Доведення. Відповідно до твердження про величину ставок в АСМП, $\boldsymbol{x} * \boldsymbol{c_1} = \boldsymbol{y} * \boldsymbol{c_2}$, а з означення банку АСМП: $\boldsymbol{t} = \boldsymbol{x} + \boldsymbol{y}$, отже можемо записати

$$\boldsymbol{x} * \boldsymbol{c_1} = (\boldsymbol{t} - \boldsymbol{x}) * \boldsymbol{c_2}$$

$$\boldsymbol{x} * (\boldsymbol{c_1} + \boldsymbol{c_2}) = \boldsymbol{t} * \boldsymbol{c_2}$$

$$\boldsymbol{x} = \boldsymbol{t}\frac{\boldsymbol{c_2}}{\boldsymbol{c_1} + \boldsymbol{c_2}}$$

За твердженням про величину ставок в АСМП, $\boldsymbol{y = x * \frac{c_1}{c_2}}$, тому

$$\boldsymbol{y = t\frac{c_1}{c_1 + c_2}}$$

Твердження 2.3 (про величину чистого прибутку для АСМП) Для події **A**, що має рівно два взаємовиключних результати, на множині букмекерів $\{\boldsymbol{B}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ чистий прибуток дорівнює $\boldsymbol{t(\frac{c_1 * c_2}{c_1 + c_2} - 1)}$.

Доведення. Згідно до твердження про величину ставок в АСМП відносно її банку та означення чистого прибутку маємо:

$$\boldsymbol{pr = x(c_1 - 1) - y = t\frac{c_2}{c_1 + c_2}(c_1 - 1) - t\frac{c_1}{c_1 + c_2} = \frac{t}{c_1 + c_2}(c_1 * c_2 - c_1 - c_2) = t(\frac{c_1 * c_2}{c_1 + c_2} - 1)}.$$

Твердження 2.4 (про максимальний прибуток арбітражної ставки) Для події **A**, що має рівно два взаємовиключних результати, на множині букмекерів $\{\boldsymbol{B}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ максимальний прибуток арбітражної ставки співпадає з прибутком АСМП.

Доведення. З означення прибутку: $\boldsymbol{t = x + y}$. Покладемо $\boldsymbol{y = \alpha x}$, де $\boldsymbol{\alpha = \frac{y}{x}}$. Максимізуємо прибуток АС:

$$\boldsymbol{pr = x(c_1 - 1) - y - y(c_2 - 1) + x = x * c_1 - y * c_2 \to max}$$

$$\boldsymbol{pr = x(c_1 - \alpha * c_2) \to max}$$

$$\boldsymbol{\dot{pr} = c_1 - \alpha * c_2 = 0}$$

$$\boldsymbol{\alpha = c_1/c_2}$$

Підставивши значення $\boldsymbol{\alpha}$ отримаємо: $\boldsymbol{x * c_1 = y * c_2}$, що співпадає з формулою величини ставок в АСМП. Отже і максимальний прибуток АС співпадає з прибутком АСМП.

Твердження 2.5 (про необхідні та достатні умови існування АСМП) Для події **A**, що має рівно два взаємовиключних результати, на множині букмекерів $\{\boldsymbol{B}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ для існування арбітражної ставки максимального

прибутку необхідно та достатньо, щоб добуток коефіцієнтів перевищував їх суму.

Доведення. Доведемо необхідність. З твердження про величину чистого прибутку для АСМП: $\boldsymbol{pr = t(\frac{c_1 * c_2}{c_1 + c_2} - 1)}$

А з означення АСМП: $\boldsymbol{pr > 0}$**.** Оскільки $\boldsymbol{t > 0}$ за означенням, маємо:

$$\frac{\boldsymbol{c_1 * c_2}}{\boldsymbol{c_1 + c_2}} - \boldsymbol{1 > 0}$$

$$\boldsymbol{c_1 * c_2 > c_1 + c_2}$$

Доведемо достатність. Нехай маємо $\boldsymbol{c_1 * c_2 > c_1 + c_2,}$ тоді, аналогічно, прибуток АСМП додатний за додатністю обох множників.

2.6 Узагальнення поняття арбітражної ставки.

У попередньому підрозділі ми ввели поняття арбітражної ставки на множині букмекерів $\{\boldsymbol{B}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ для події **A** з рівно двома можливими взаємовиключними результатами. Узагальнимо його на події з довільною кількістю взаємовиключних результатів.

Означення 5.1 Арбітражною ставкою (АС) на подію **A**, що має **M** взаємовиключних результатів на множині букмекерів $\{\boldsymbol{B}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ назвемо множину з **M** ставок, що задовольняє наступній умові: при перемозі будь якого з **M** результатів сума чистих прибутків з цих ставок буде строго додатною. Суму чистих прибутків множини ставок АС так само називатимемо її прибутком (або чистим прибутком).

Означення 5.2 Арбітражною ставкою максимального прибутку (АСМП) на подію **A**, що має **M** взаємовиключних результатів на множині букмекерів $\{\boldsymbol{B}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ назвемо множину з **M** ставок, що задовольняє наступній умові: при перемозі будь якого з **M** результатів сума чистих прибутків з цих ставок буде **рівною** та строго додатною.

Твердження 3.1 (про величину ставок в АСМП) Для події **A**, що має **M** взаємовиключних результати, на множині букмекерів $\{\boldsymbol{B}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ величини ставок з множини АСМП задовольняють наступному співвідношенню: $\boldsymbol{x_1} * \boldsymbol{c_1} = \boldsymbol{x_2} * \boldsymbol{c_2} = \cdots = \boldsymbol{x_M} * \boldsymbol{c_M}$, де $\mathbf{x_1}, \ldots, \mathbf{x_M}$ – суми ставок на коефіцієнти $\mathbf{c_1}$, …, $\mathbf{c_M}$ відповідно.

Доведення. За означенням АСМП маємо: $\boldsymbol{x_1} * (\boldsymbol{c_1} - \mathbf{1}) - \sum_{\boldsymbol{i=2}}^{\boldsymbol{M}} \boldsymbol{x_i} = \boldsymbol{x_2} * (\boldsymbol{c_2} - \mathbf{1}) - \sum_{\boldsymbol{i=1, i\neq 2}}^{\boldsymbol{M}} \boldsymbol{x_i} = \cdots = \boldsymbol{x_M} * (\boldsymbol{c_M} - \mathbf{1}) - \sum_{\boldsymbol{i=1, i\neq M}}^{\boldsymbol{M}} \boldsymbol{x_i}$

Розв'язки цих рівнянь будуть аналогічними до розв'язків з твердження 2.1:

$$\boldsymbol{x_1} * \boldsymbol{c_1} = \boldsymbol{x_2} * \boldsymbol{c_2} = \cdots = \boldsymbol{x_M} * \boldsymbol{c_M}$$

Твердження 3.2 (про величину ставок в АСМП відносно її банку) Для події **A**, що має **M** взаємовиключних результати, на множині букмекерів $\{\boldsymbol{B}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ ставки множини АСМП мають наступний розмір: $\boldsymbol{x_i} = \boldsymbol{t} \,/\, \sum_{\boldsymbol{j=1}}^{\boldsymbol{M}} \frac{\boldsymbol{c_i}}{\boldsymbol{c_j}}$, де **t** – банк АСМП, $\boldsymbol{x_i}$ – сума ставки на коефіцієнт $\boldsymbol{c_i}$.

Доведення. Щоб обчислити величину ставки $\boldsymbol{x_i}$ відносно банку АСМП необхідно розв'язати наступну систему лінійних рівнянь:

$$\begin{cases} \sum_{\boldsymbol{i=1}}^{\boldsymbol{M}} \boldsymbol{x_i} = \boldsymbol{t} \\ \boldsymbol{x_1} * \boldsymbol{c_1} - \boldsymbol{x_2} * \boldsymbol{c_2} = \mathbf{0} \\ \boldsymbol{x_1} * \boldsymbol{c_1} - \boldsymbol{x_3} * \boldsymbol{c_3} = \mathbf{0} \\ \ldots \\ \boldsymbol{x_1} * \boldsymbol{c_1} - \boldsymbol{x_M} * \boldsymbol{c_M} = \mathbf{0} \end{cases} \quad (1)$$

$$\begin{cases} \sum_{\boldsymbol{i=1}}^{\boldsymbol{M}} \boldsymbol{x_i} = \boldsymbol{t} \\ \boldsymbol{x_2} = \boldsymbol{x_1} \dfrac{\boldsymbol{c_1}}{\boldsymbol{c_2}} \\ \boldsymbol{x_3} = \boldsymbol{x_1} \dfrac{\boldsymbol{c_1}}{\boldsymbol{c_3}} \\ \ldots \\ \boldsymbol{x_M} = \boldsymbol{x_1} \dfrac{\boldsymbol{c_1}}{\boldsymbol{c_M}} \end{cases}$$

Підставивши вирази для $\boldsymbol{x_i}$ в перше рівняння отримаємо

$$\boldsymbol{x_1 = t \,/\, (1 + \frac{c_1}{c_2} + \cdots + \frac{c_1}{c_M}) = t \,/ \sum_{j=1}^{M} \frac{c_1}{c_j}}$$

Для решти $\boldsymbol{x_i}$ аналогічно.

Твердження 3.3 (про величину чистого прибутку для АСМП) Для події **A**, що має **M** взаємовиключних результати, на множині букмекерів $\{\boldsymbol{B}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ чистий прибуток дорівнює $\boldsymbol{t}(\frac{1}{\sum_{i=1}^{M}\frac{1}{c_i}} - \boldsymbol{1})$.

Доведення. Згідно до твердження про величину ставок в АСМП відносно її банку та означення чистого прибутку маємо:

$$\boldsymbol{pr = x_1(c_1 - 1) - \sum_{i=2}^{M} x_i = \frac{t}{\sum_{j=1}^{M}\frac{c_1}{c_j}}(c_1 - 1) - \sum_{i=2}^{M}\frac{t}{\sum_{j=1}^{M}\frac{c_i}{c_j}}}$$

$$\boldsymbol{= t\left(\frac{1}{c_1 * \sum_{j=1}^{M}\frac{1}{c_j}}(c_1 - 1) - \sum_{i=2}^{M}\frac{1}{c_i * \sum_{j=1}^{M}\frac{1}{c_j}}\right)}$$

$$\boldsymbol{= t\left(\frac{\prod_{l=2}^{M} c_l}{\prod_{k=1}^{M} c_k * \sum_{j=1}^{M}\frac{1}{c_j}}(c_1 - 1) - \sum_{i=2}^{M}\frac{\prod_{l=2, l\neq i}^{M} c_l}{\prod_{k=1}^{M} c_k * \sum_{j=1}^{M}\frac{1}{c_j}}\right)}$$

$$\boldsymbol{= t\left(\frac{\prod_{i=1}^{M} c_i - \prod_{l=2}^{M} c_l}{\prod_{k=1}^{M} c_k * \sum_{j=1}^{M}\frac{1}{c_j}} - \sum_{i=2}^{M}\frac{\prod_{l=2, l\neq i}^{M} c_l}{\prod_{k=1}^{M} c_k * \sum_{j=1}^{M}\frac{1}{c_j}}\right)}$$

$$\boldsymbol{= t\left(\frac{\prod_{p=1}^{M} c_p - \sum_{i=1}^{M}\prod_{l=1, l\neq i}^{M} c_l}{\prod_{k=1}^{M} c_k * \sum_{j=1}^{M}\frac{1}{c_j}}\right) = t\left(\frac{1 - \sum_{i=1}^{n}\frac{1}{c_i}}{\sum_{j=1}^{M}\frac{1}{c_j}}\right)}$$

$$\boldsymbol{= t\left(\frac{1}{\sum_{j=1}^{M}\frac{1}{c_j}} - 1\right)}$$

Твердження 3.4 (про максимальний прибуток арбітражної ставки) Для події **A**, що має **M** взаємовиключних результати, на множині букмекерів $\{\boldsymbol{B}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ максимальний прибуток арбітражної ставки співпадає з прибутком АСМП.

Доведення. Доведення аналогічне до доведення твердження 2.4, за винятком того, що в даному випадку будемо максимізувати $\boldsymbol{M-1}$ рівняння, в результаті чого знову ж таки отримаємо рівності як в твердженні 3.1

Твердження 3.5 (про необхідні та достатні умови існування АСМП) Для події **A**, що має **M** взаємовиключних результати, на множині букмекерів $\{\boldsymbol{B}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ для існування арбітражної ставки максимального прибутку необхідно та достатньо, щоб виконувалась нерівність $\frac{\mathbf{1}}{\sum_{j=1}^{M}\frac{1}{c_j}} > \mathbf{1}$.

Доведення. Доведення аналогічне до доведення твердження 2.5.

Подавши останню нерівність у еквівалентному вигляді $\sum_{\boldsymbol{j=1}}^{\boldsymbol{M}}\frac{\mathbf{1}}{\boldsymbol{c_j}} - \mathbf{1} < \mathbf{0}$, отримаємо в лівій частині маржу множини букмекерів $\{\boldsymbol{B}\}_{\boldsymbol{i=1}}^{\boldsymbol{N}}$ відносно **A.** Таким чином отримали доведення твердження про залежність прибутку від маржі, що використовувалось раніше в розділі 2.2.

2.7 Порівняння з існуючими моделями.

Термін арбітражна ставка є загальновідомим та досить широковживаним букмекерами та їх клієнтами. Однак, незважаючи на це, жодна з досліджених робіт [17][18][19][20][21] на тему арбітражних ставок, що перебувають у вільному доступі не розглядає їх в контексті більш ніж двох букмекерів. А також абсолютно всі досліджені роботи обмежуються спрощеною моделлю (для подій з двома або трьома результатами). В існуючих моделях використовуються такі терміни як маржа та арбітражна ставка. В даній роботі було наведено визначення, що були зручними для

більш детального опису математичної моделі, тому значення деяких слів може відрізнятись від наданого в іншій літературі (хоча часто слова матимуть однаковий або дуже схожий зміст). Таким чином, можна стверджувати, що всі визначення та твердження є авторськими, а створена модель більш широко розкриває існуючі моделі, що наявні у вільному доступі.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ЇЇ ОСОБЛИВОСТІ

3.1 Особливості програмної реалізації.

Продукт створено на мові Python в інтегрованому середовищі розробки PyCharm IDE Community Edition 2021.1.1 Для веб-парсингу використовуються бібліотека BeautifulSoup [2] та фреймворк Selenium [3]. Робота з веб-браузером в останньому виконується завдяки драйверу браузера Google Chrome – chromedriver. Такий підхід було обрано, оскільки Selenium [3] краще дозволяє емулювати дії звичайного користувача. Для створення телеграм боту використано бібліотеку pyTelegramBotAPI [4], що дозволяє зручно працювати з Telegram Bot API [22]. В якості бази даних для збереження налаштувань користувачів – MySQL [5]. Для роботи з Excel-файлами використано бібліотеку xlsxwriter [6]. Бота розгорнуто на безкоштовному сервері Heroku [7] за допомогою сторонніх білдпаків (third-party buildpacks): heroku-buildpack-google-chrome [23], heroku-buildpack-chromedriver [24].

Інтерфейсом користувача в створеному продукті виступає телеграм бот, який відповідає за ввід та вивід даних. При такому підході програма може бути використана на будь-якій операційній системі та платформі, що підтримується Telegram.

На даний момент програмним продуктом підтримуються наступні букмекери:

- Parimatch;
- 1xBet;
- Favorit Sport;

– Marathon Bet;

– GGBet.

Щодо видів спорту, наразі реалізовано підтримку таких дисциплін та видів спорту:

a) футбол;

b) кіберспорт:

1) «Counter-Strike: Global Offensive»;

2) «Dota 2»;

3) «League of Legends».

3.2 Можливості програмної реалізації.

Опишемо реалізовані команди інтерфейсу користувача:

Вивести АСМП для заданої дисципліни/виду спорту (Рис. 2).

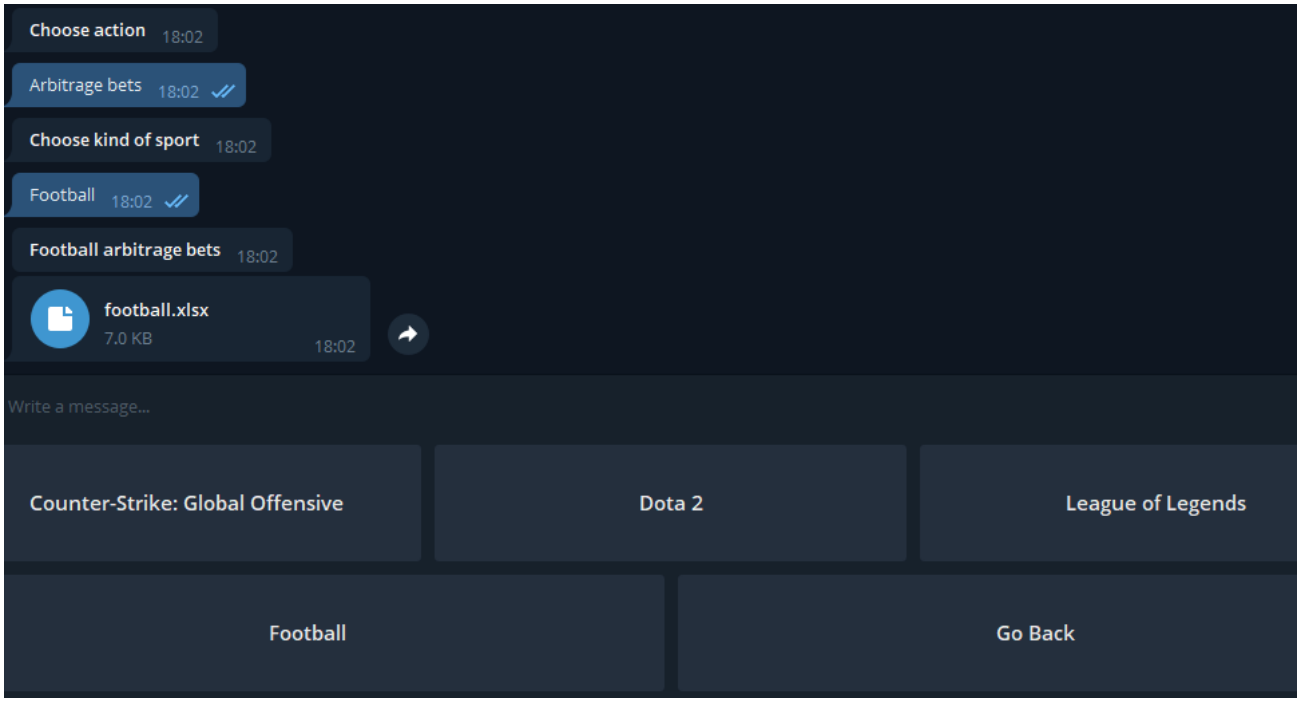

Рисунок 2 – Приклад виведення списку АСМП для виду спорту «Футбол»

Сценарій використання: користувач обирає в меню опцію «Arbitrage Bets», після чого обирає дисципліну/вид спорту. Бот повертає користувачу Excel-файл, що містить АСМП для обраної дисципліни/виду спорту.

Підписатися на дисципліну/вид спорту (Рис. 3).

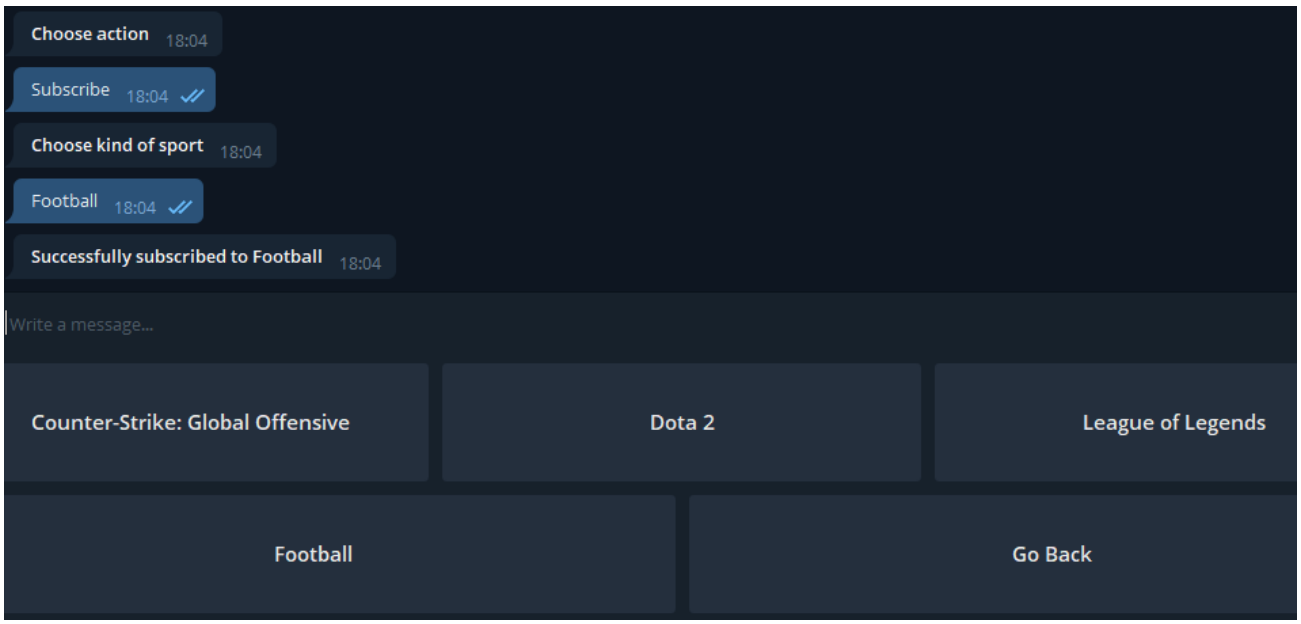

Рисунок 3 – Приклад підписки на вид спорту «Футбол»

Сценарій використання: користувач обирає в меню опцію «Subscribe», після чого обирає дисципліну/вид спорту. Бот повідомляє користувачу про успішність виконання операції.

Відписатися від дисципліни/виду спорту (Рис. 4).

Сценарій використання: користувач обирає в меню опцію «Unsubscribe», після чого обирає дисципліну/вид спорту. Бот повідомляє користувачу про успішність виконання операції.

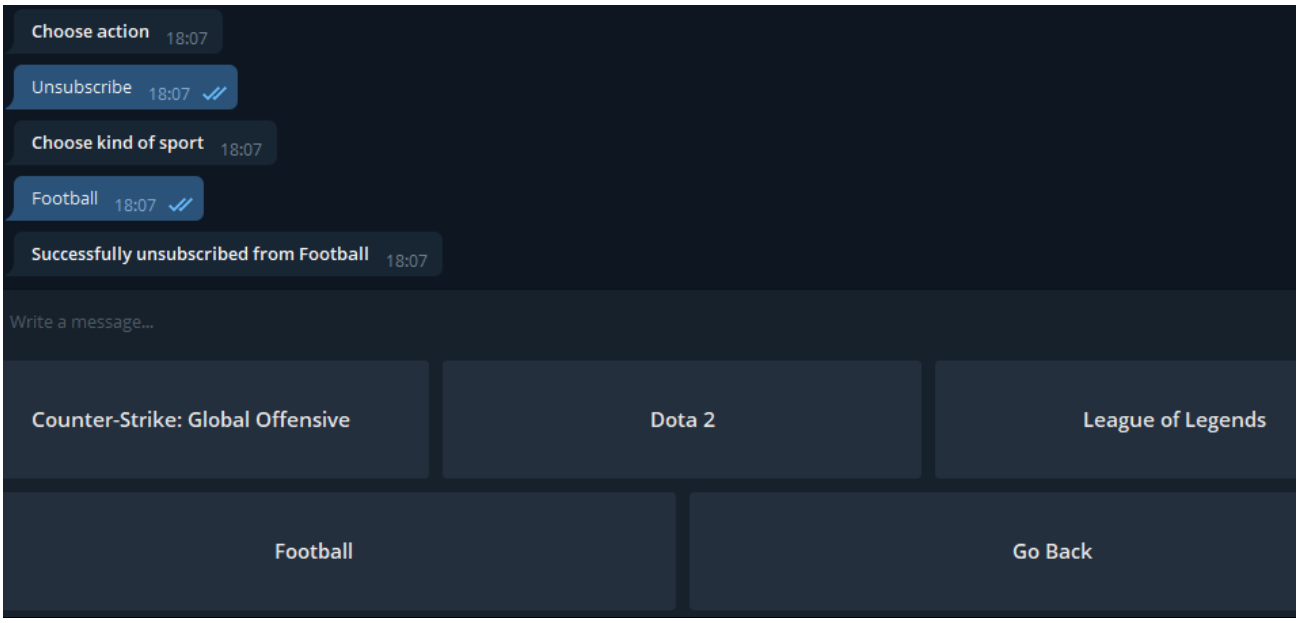

Рисунок 4 – Приклад відписки від виду спорту «Футбол»

Переглянути список підписок (Рис. 5).

Сценарій використання: користувач обирає в меню опцію «Show my subscriptions», після чого обирає дисципліну/вид спорту. Бот повідомляє користувачу про успішність виконання операції.

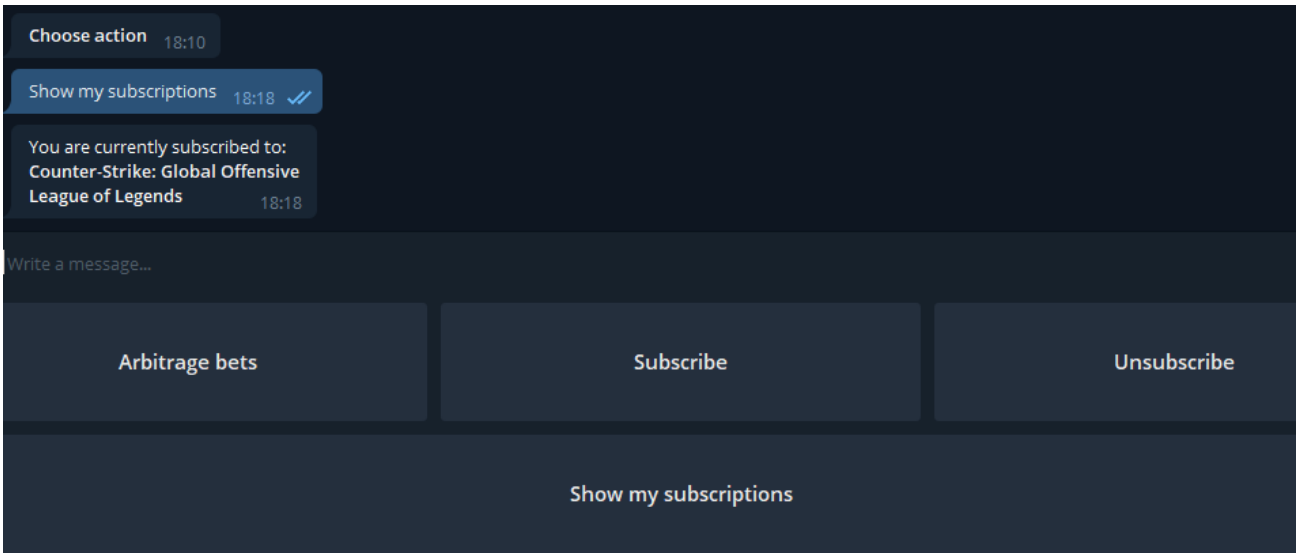

Рисунок 5 – Приклад переглядання списку підписок

Меню телеграм бота складається з двох рівнів: рівень вибору одного з вищеописаних сценаріїв взаємодії та, якщо необхідно, меню вибору

дисципліни/виду спорту. Другий рівень меню також містить кнопку «Go Back», що дозволяє користувачеві повернутися на попередній рівень меню.

3.3 Розгортання системи на сервері Heroku.

Для розгортання системи було використано хмарну платформу Heroku [7], що дозволяє безкоштовно розв'язати цю задачу. Для того, щоб використовувати Selenium [3] разом з Google Chrome та chromedriver на сервері, було встановлено сторонні білдпаки heroku-buildpack-google-chrome [23] та heroku-buildpack-chromedriver [24]. Для налаштування взаємодії з базою даних використано адон (add-on) ClearDB [25] MySQL [5]. Також одним з важливих аспектів є встановлення опції запуску драйвера Google Chrome в режимі «headless», тобто без графічного інтерфейсу, оскільки сервером не передбачено використання останнього.

3.4 Порівняння з аналогами.

На сьогоднішній день існує декілька аналогів реалізованого програмного продукту. Наведемо порівняння з ними в Таблиці 2.

Таблиця 2 – порівняння з аналогами

Сервіс/ Метрика	Продукт роботи	AllBestBets	surebet	Scan-Sport
Платний	Безкоштовний	Частково (обмежений функціонал)	Частково (обмежений функціонал)	Тимчасово безкоштовний
Арбітражні	Так	Так	Так	Так

ставки «прематч»				
Арбітражні ставки «вживу» (live)	Ні	Так	Так	Так
Ставки з перевагою	Ні	Так	Так	Так
Багатовимірні і арбітражні ставки	Так	Так	Ні	Так
Веб-сайт	Ні	Так	Так	Так
Телеграм-бот	Так	Ні	Ні	Ні
Види спорту/ дисципліни	4	40	15	15
Букмекери	5	50+	80	56
Додатковий функціонал	Ні	Блог, форум, служба підтримки, калькулятор ставок	Сортування, фільтри, калькулятор ставок	Новини, служба підтримки, форум, рейтинг букмекерів

Як можна побачити з таблиці, основними недоліками створеного програмного забезпечення в порівнянні з аналогами є відсутність підтримки відстеження арбітражних ставок «вживу» (тобто, для матчів, що відбуваються в поточний момент часу) та відносно невелика кількість

підтримуваних видів спорту та букмекерів, що може прямо впливати на кількість знайдених арбітражних ситуацій. Решта недоліків є незначними, оскільки вони не мають відношення до поставлених в роботі задач. Також можна відмітити, що усі з розглянутих аналогів використовують веб-сайт для взаємодії з користувачами, тому застосунок телеграм боту в створеній системі робить її в певному сенсі унікальною. І в решті решт, варто звернути увагу на те, що ключова складова функціоналу усіх аналогічних продуктів є платною (наприклад, безкоштовно доступні лише арбітражні ставки зі значенням прибутку менше 1% від суми ставки) на відміну від програмної реалізації роботи.

3.5 Практичне застосування.

В результаті практичного використання створеної системи для відстеження арбітражних ставок було підтверджено правильність роботи програмно реалізації. Тобто знайдені системою ставки дійсно можна було знайти на веб-сайтах відповідних букмекерів, а значення коефіцієнтів відповідали тим, що на сайтах. Таким чином було помічено, що на практиці арбітражні ставки рідко тримаються більше декількох годин (тобто, відбувається корегування коефіцієнтів букмекерами), однак бувають і такі випадки, коли час їх життя досягає декількох тижнів. Значення ж прибутку у відсотках від банку ставки частіше за все знаходяться в межах до 3%, однак в деяких випадках можуть сягати і 10%.

4 КОМПОНЕНТИ ТА МОДУЛІ ПРОГРАМИ

Програмна реалізація складається з таких компонентів (Рис. 6):

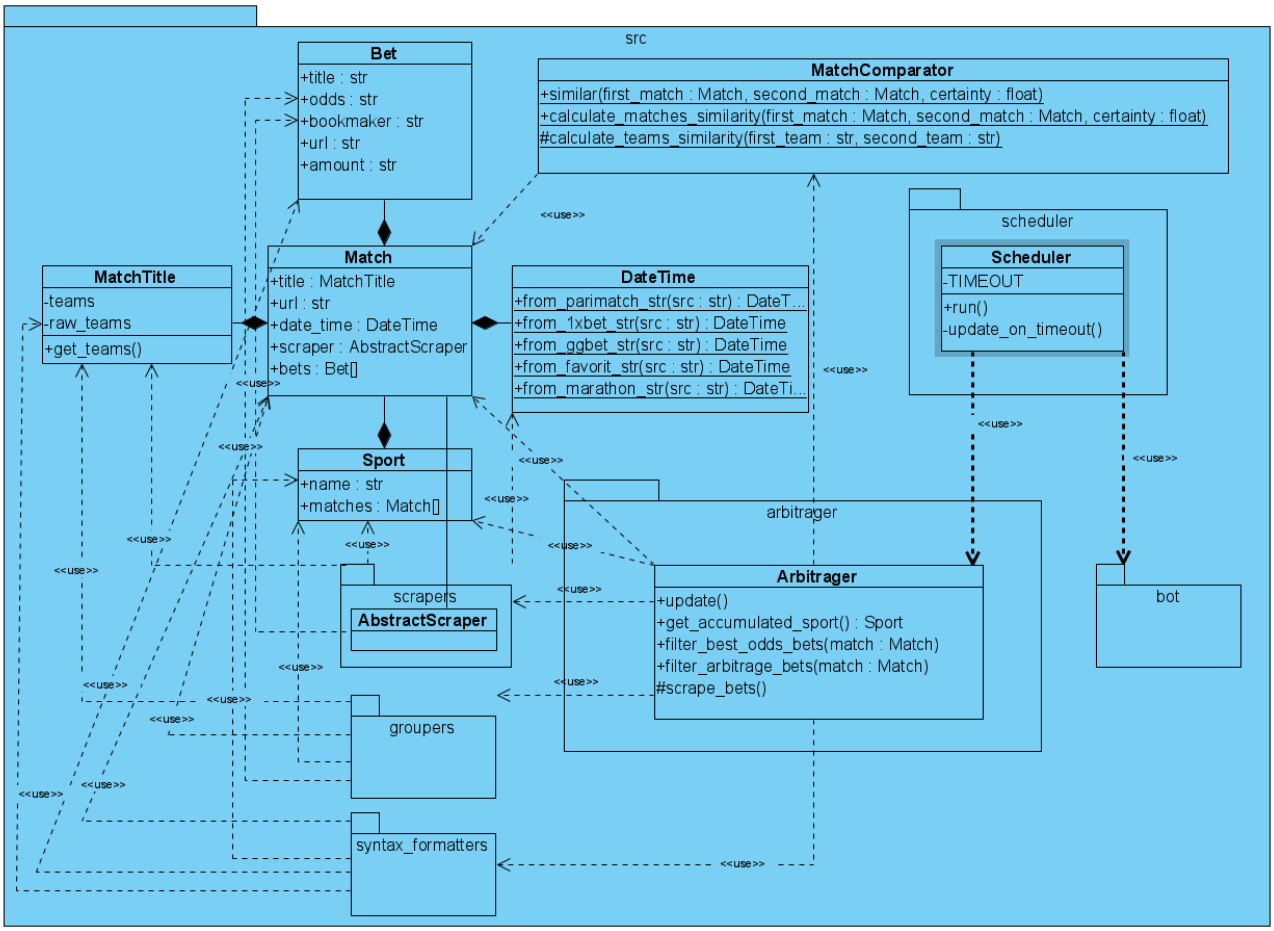

Рисунок 6 – Діаграма класів

4.1 Компоненти renderer та scrapers.

renderer (Рис. 7) – емулює роботу веб-браузера Google Chrome, використовуючи фреймворк Selenium [3] та драйвер chromedriver. Необхідність реалізації компонента зумовлена тим, що часто код сторінок веб-сайтів формується динамічно (наприклад, за допомогою мови

програмування JavaScript) і для того, щоб отримати код сторінки таким, яким його бачить користувач потрібно виконати її рендеринг (як це роблять веб-браузери). Саме цю задачу і вирішує компонент renderer. Також він реалізує часто використовувані при парсингу методи, такі як отримання вихідного коду сторінки та навігація по сторінці за допомогою натискання на веб-елементи.

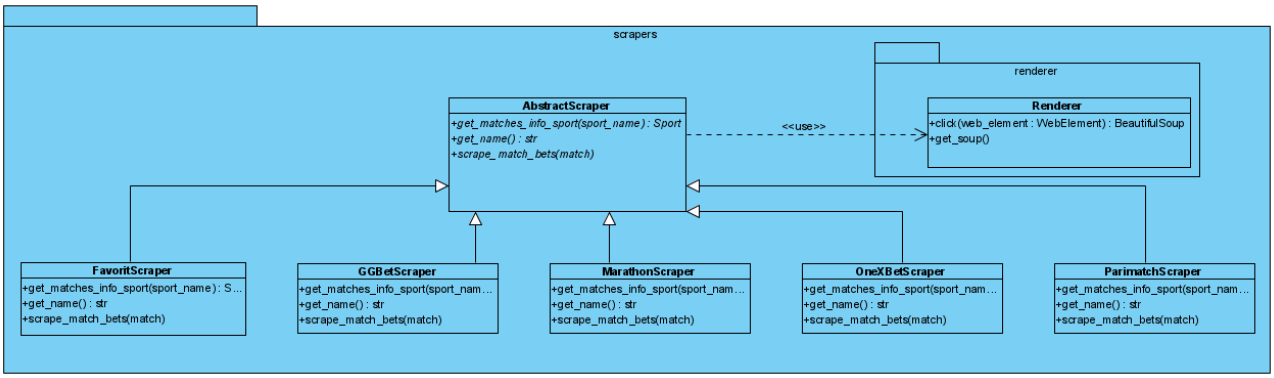

Рисунок 7 – Діаграма класів компоненту scrapers

scrapers (Рис. 7) – відповідає за веб-скрапінг інформації з сайтів букмекерів, зокрема, назв та часу проведення матчів, посилань на матчі, назв та коефіцієнтів ставок. Для кожного букмекера реалізовано свій алгоритм парсингу веб-сторінок, оскільки інтерфейс сайтів є унікальним та потребує особистого підходу. Інтерфейс AbstractScraper описує спільні для всіх скраперів методи. До інтерфейсу застосовано патерн Singleton за допомогою механізму метакласів мови Python (такий варіант реалізації найкраще відповідає принципам ООП та дозволяє уникнути зайвого дублювання коду та надмірного його ускладнення). Таким чином відбувається контроль над створенням екземплярів кожного з класів, що реалізує цей інтерфейс. Зі спільних методів для кожного парсеру можна відмітити get_matches_info_sport(sport_name) та scrape_match_bets(match). Перший з яких проводить початковий збір інформації (такої як посилання на матчі, назви та час матчів) для заданої дисципліни або виду спорту. Другий же для заданого матчу збирає назви та коефіцієнти ставок. Парсинг відбувається з

використанням компоненту renderer та за допомогою функціоналу фреймворку Selenium [3] та бібліотеки BeautifulSoup [2].

4.2 Компонент syntax_formatters.

syntax_formatters – відповідає за уніфікацію синтаксису використовуваного букмекерами для найменування ставок. Форматування синтаксису та встановлення певного «компромісного» варіанта є дуже важливим етапом, оскільки завдяки цьому потім стає можливим агрегація зібраної інформації.

Цей компонент має найскладнішу з усіх компонентів ієрархію класів (Рис. 8). Так само як і в випадку scrapers, спільні риси для всіх форматувальників описуються інтерфейсом AbstractSyntaxFormatter. Але на цьому винесення спільних рис не завершується, адже дисципліни в рамках одного спорту (наприклад, кіберспортивні дисципліни «Counter-Strike: Global Offensive», «Dota 2» та «League of Legends») також мають між собою багато спільного, тому для кожного виду спорту створюється свій абстрактний базовий клас, що реалізує інтерфейс AbstractSyntaxFormatter (для кіберспортивних дисциплін цей базовий клас – AbstractEsportsSyntaxFormatter). У випадку, якщо спорт немає додаткових дисциплін (наприклад, футбол), конкретні реалізації форматувальників для кожного букмекера успадковуються безпосередньо від абстрактного базового класу цього спорту (для футболу – AbstractEsportsSyntaxFormatter). Якщо ж присутній розділ на дисципліни, то від останнього успадковуються інтерфейси для конкретних дисциплін (у випадку кіберспорту такими інтерфейсами є AbstractCSGOSyntaxFormatter, AbstractDotaSyntaxFormatter та AbstractLoLSyntaxFormatter). А їх уже реалізують конкретні класи-форматувальники для кожного букмекера. Але і на цьому не завершується

проектування архітектури компоненту, адже класи форматувальники в рамках різних дисциплін/видів спорту, також можуть мати спільні риси, що визначаються букмекером форматуванням синтаксису якого займається клас. Тому для кожного букмекера створено свій інтерфейс форматувальника в який винесено такі риси, а для конкретних реалізацій застосовується множинне наслідування (від інтерфейса виду спорту/категорії та від інтерфейса букмекера).

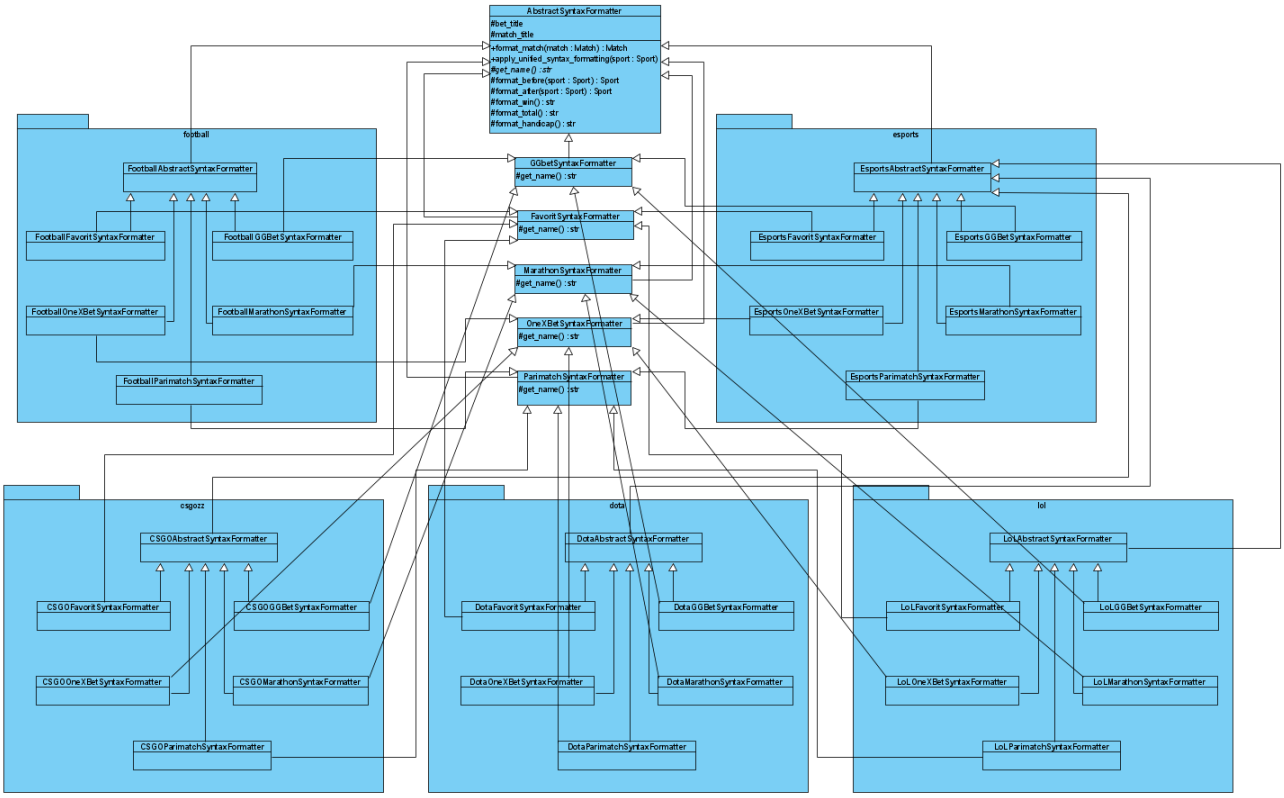

Рисунок 8 – Діаграма класів компоненту syntax_formatters

Як і в випадку scrapers до класу AbstractSyntaxFormatter застосовано патерн Singleton (якраз для такої багаторівневої ієрархії класів і проявляються усі переваги реалізації цього патерна через метаклас). Також використано патерн проектування Template Method. Таким чином інтерфейс AbstractSyntaxFormatter задає порядок в якому відбувається форматування назв ставок, а підкласи в рамках ієрархії вже самі визначають чи потрібне форматування того чи іншого елемента назви ставки, чи можна довіритися

його реалізації з «верхнього» рівня. Такий варіант дозволяє зменшити дублювання коду, що надзвичайно важливо для такої великої кількості класів.

Форматування відбувається за допомогою використання регулярних виразів [26] та рядкових функцій;

4.3 Компонент groupers.

groupers (Рис. 9) – розв'язує задачу агрегації ставок та матчів.

Розв’язання задачі агрегації ставок означає, що в рамках певного матчу ставки мають бути розбиті на групи таким чином, щоб в межах кожної групи знаходилися лише ставки певної АСМП. Наприклад, для футбольного матчу Динамо – Шахтар одна з таких груп буде: «Динамо виграє», «Нічия», «Шахтар виграє».

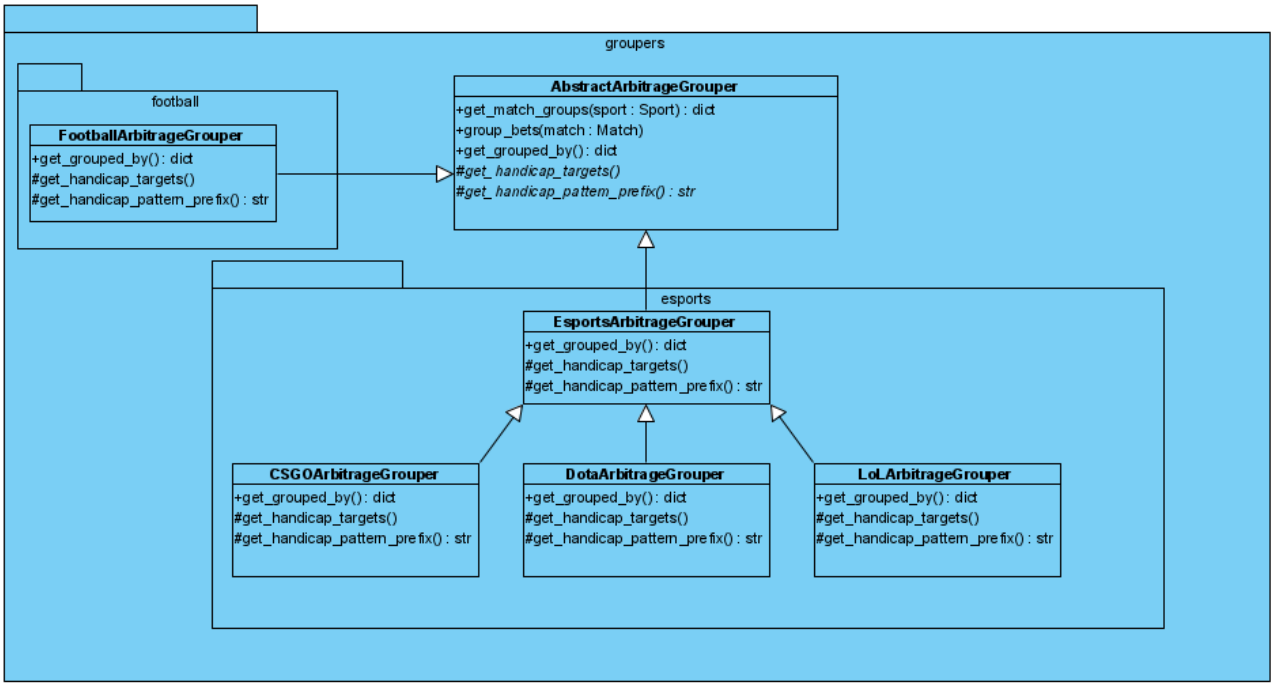

Рисунок 9 – Діаграма класів компоненту groupers

Задача агрегації матчів детально розглядається в наступному підрозділі.

4.4 Агрегація матчів та використані алгоритми.

Розв’язання задачі агрегації матчів означає, що матчі необхідно згрупувати таким чином, щоб в рамках однієї групи знаходилися матчі, що відповідають одному і тому ж реальному матчу. Наприклад, одна із груп може бути такою: «Динамо – Шахтар (Букмекер 1)», «Динамо – Шахтар (Букмекер 2)» і т. д.

Варто також зазначити, що букмекери можуть мати різні найменування для однієї і тієї ж команди (наприклад, Dynamo Kyiv, FC Dynamo Kyiv, FC Dynamo Kiev) тому поставлена задача аж ніяк не є тривіальною. Для її вирішення було реалізовано алгоритм порівняння матчів за назвою (назва включає дві команди, що беруть участь у матчі), часом проведення матчу та букмекером. Значення, що повертає алгоритм відповідає схожості двох матчів, де 1 означає, що матчі вважаються ідентичними, 0 – абсолютно різними.

Для початку введемо алгоритм порівняння назв команд. Нехай L_{min} – довжина назви коротшої з двох команд, L_{max} – довжина назви довшої команда. Знайдемо для двох назв всі спільні підрядки довжини не менше трьох символів (починаючи з найбільших до найменших). Нехай L_{sub}– їх сумарна довжина. Тоді схожість назв команд дорівнюватиме: $S_t = \frac{L_{sub}}{L_{min}} + \frac{L_{sub}-L_{max}}{10L_{max}}$. Перший доданок визначає частку літер коротшої назви, що співпала з літерами довшої назви та набуває значень від 0 (не знайдено спільних підрядків довжини не менше трьох) до 1 (менша назва є комбінацією підрядків більшої назви). Другий же доданок набуває значень від -0.1 до 0 та визначає частку літер довшої назви, що не співпала з літерами меншої назви. Щоб краще зрозуміти сенс другого доданку розглянемо

порівняння назв «Orlean», «FC Orlean» та порівняння назв «Orlean», «FC New Orlean». Легко помітити, що в обох випадках менша з двох назв є підрядком більшої. Тому перший доданок має значення $\frac{L_{sub}}{L_{min}} = 1$. Але доданок $\frac{L_{sub}-L_{max}}{10L_{max}}$ в першому випадку набуває значення $\frac{L_{sub}-L_{max}}{10L_{max}} = \frac{6-9}{10*9} = \frac{-3}{90} = -\frac{1}{30}$, а в другому: $\frac{L_{sub}-L_{max}}{10L_{max}} = \frac{6-13}{10*13} = -\frac{7}{130}$. А отже, результуюче значення S_t виявиться трохи більшим в першому випадку, що зіграло би вирішальну роль, якби ми хотіли підібрати найбільш схожу назву до «Orlean» серед «FC Orlean» та «FC New Orlean».

Тепер можемо переходити до визначення схожості двох матчів. Нехай Δt – різниця в годинах між часом проведення матчів. Введемо часовий коефіцієнт $C_t = \begin{cases} 1, \text{якщо } \Delta t < 1 \\ \frac{1}{\Delta t}, \text{інакше} \end{cases}$. Серед всіх можливих попарних групувань команд з різних матчів (всього $n!$ групувань, якщо в кожному матчі бере участь n команд) виберемо таке, при якому сумарна схожість команд є максимальною. Тоді схожість матчів визначатиметься наступним чином: $S_m = C_t \frac{S_{max}}{n}$, де n – кількість команд, що бере участь у кожному матчі, S_{max} – максимальна сумарна схожість команд. Таким чином схожість матчів прямо пропорційна середній схожості їх команд та обернено пропорційна різниці в часі між ними, при чому різниця в межах однієї години вважається допустимою та не впливає на результат (це зумовлено тим, що матчі можуть переноситись на невеликий час, наприклад, через погодні умови або організаційні моменти);

4.5 Компоненти arbitrager, bot та scheduler.

arbitrager – керує процесами скрапінгу, форматування та агрегації (компонентами scrapers, syntax_formatters, groupers відповідно); займається пошуком арбітражних ставок. Фактично, цей компонент використовує математичну модель з розділу 2 для аналізу агрегованої та уніфікованої інформації. Знайдені арбітражні ставки компонент записує за допомогою модуля xlsxwriter [6] в Excel-файли (Рис. 10). Такий варіант подання інформації зручний для використання та можливого подальшого аналізу користувачами;

	Match	Time	Profit	Bet	Odds	Bet amount	Bookmaker	URL
2	ex-ethereal - lyngby vikings	2020-12-28 20:30:00	3.71%	ex-ethereal handicap -1.5 maps	2.36	0.4394	marathon	https://www.mara
3				lyngby vikings handicap +1.5 maps	1.85	0.5606	ggbet	https://gg.bet/en
4			4.10%	ex-ethereal handicap -6.5 rounds	1.81	0.5751	favorit	https://old.favorit
5				lyngby vikings handicap +6.5 rounds	2.45	0.4249	ggbet	https://gg.bet/en
6								
7								

Рисунок 10 – Приклад вигляду Excel-файлу з арбітражними ставками

bot (Рис. 11) – відповідає за роботу телеграм боту для комунікації з користувачами. Використовується реалізація багаторівневого меню на основі структури даних «дерево» (вузлами виступають опції меню). Для збереження інформації про користувачів, як наприклад, їх підписки на дисципліни/види спорту використовується база даних MySQL [5]. Таким чином, кожного разу коли користувач проводить операцію над підписками відбувається відповідна взаємодія з базою даних, що дозволяє запам'ятовувати налаштування для різних користувачів. Компонент bot нічого не знає про інші компоненти системи, окрім місцезнаходження Excel-файлів, які він повертає за запитом користувача;

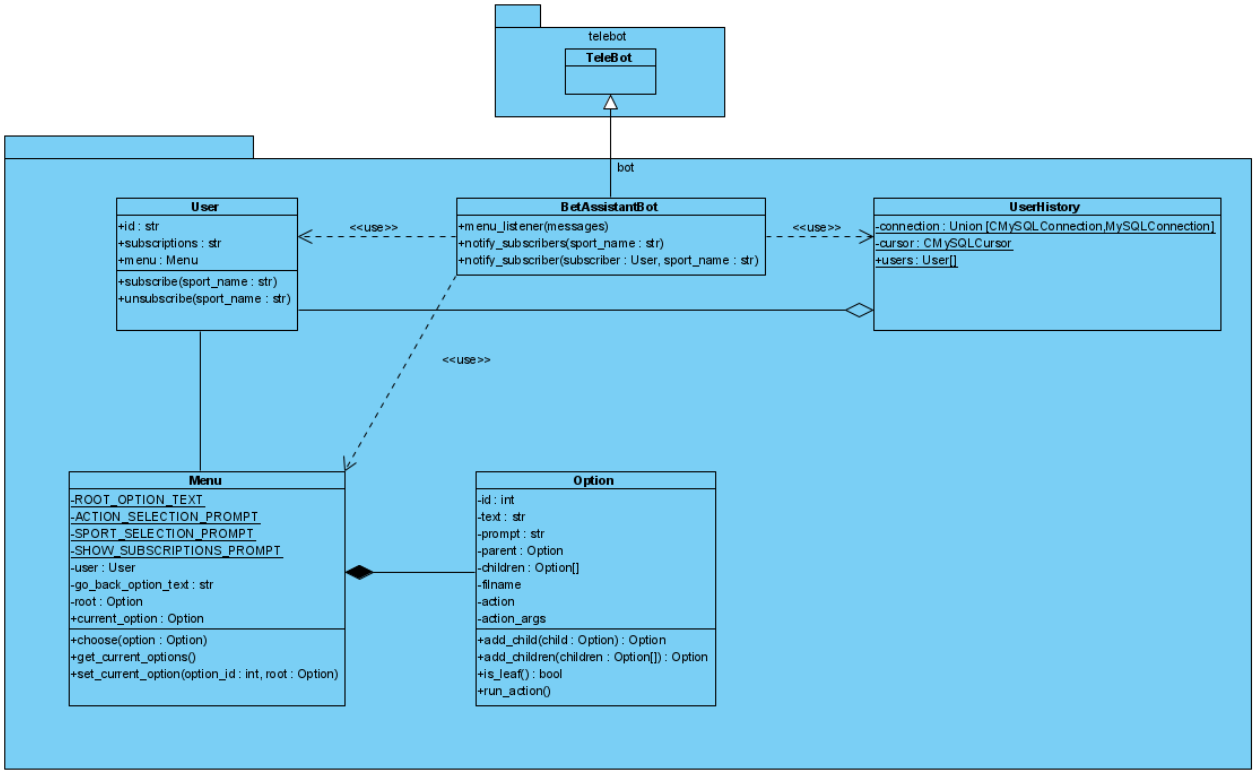

Рисунок 11 – Діаграма класів компоненту bot

scheduler – відповідає за паралельний запуск компонентів bot та arbitrager. Задає частоту оновлення інформації компонента arbitrager, тобто, наприклад, кожні дві години весь процес пошуку арбітражних ситуацій запускається спочатку та в результаті оновлюється вміст Excel-файлів.

4.6 Оптимізація часових показників та актуальності інформації.

Кількість матчів розміщених на сайтах букмекерів для таких популярних видів спорту як футбол часто складає тисячі, що вже казати про сотні можливих варіантів ставок, що пропонуються на них букмекерами. Одразу постає питання оптимізації часу роботи системи. Але ще більш важливим є питання актуальності даних, що повертаються системою, оскільки однією з найважливіших характеристик інформації є її своєчасність. Це особливо важливо для програмних продуктів такого типу,

адже частота зміни коефіцієнтів прямо впливає на достовірність даних на момент часу коли вони дійдуть до користувача системи.

Саме тому в системі дії виконуються в наступному порядку:

- збір мінімально необхідної для агрегації матчів інформації про матчі;
- агрегація матчів;
- для кожної групи матчів збір решти даних (назви та коефіцієнти ставок тощо);
- агрегація ставок;
- пошук арбітражних ситуацій.

Таким чином, спершу відбувається агрегація матчів, що дозволяє мінімізувати проміжок між парсингом одного і того ж матча на сайтах різних букмекерів.

4.7 Особливості оптимізації часових показників та результати вимірів.

Тепер, коли важливішу з двох вищеописаних проблем розв’язано, можна переходити до оптимізації часових показників парсингу. Далі наведемо час парсингу 20 матчів (обрано саме таку кількість матчів, бо її достатньо, щоб підрахувати середнє значення, а сам процес скрапінгу виконується за прийнятний час) для різних дисциплін та видів спорту в секундах (Таблиця 3, послідовна реалізація):

Таблиця 3 – результати вимірів

Реалізація/ Дисципліна	Послідовна	Паралельна	Оптимізована	Прискорення
«Counter-Strike: Global Offensive»	522 сек	250 сек	51 сек	2,08/10,23
«Dota 2»	513 сек	240 сек	50 сек	2,14/10,26
«League of Legends»	591 сек	269 сек	59 сек	2,20/10,01
Футбол	1514 сек	953 сек	152 сек	1,59/9,96

Як бачимо в середньому один футбольний матч обробляється за 75 секунд, що є абсолютно неприйнятним показником, оскільки за умови необхідності обробки тисячі таких матчів загальний час складе 75 000 секунд або, приблизно, 21 годину. Щоб зменшити час виконання скрапінгу можемо виконувати парсинг паралельно для кожного сайту, адже це абсолютно різні задачі. Запропоновану паралелизацію було виконано за допомогою використання 5 потоків (по одному для кожного букмекера) на етапі с) виконання програми. Конкретна реалізація використовує пакет стандартної бібліотеки concurrent та модуль futures. Застосувавши описану оптимізацію отримаємо наступні показники, наведені в Таблиці 3 (паралельна реалізація).

Як бачимо, отримали помітні покращення в продуктивності, але менші ніж можна було б очікувати. Серед причин можна виділити дві.

По перше, реалізація використовуваного в проекті та самого популярного інтерпретатора Python – CPython. Проблема в тому, що ця реалізація вирішує проблему синхронізації потоків за допомогою Global Interpreter Lock (GIL, можна розглядати його як примітив синхронізації mutex, тільки от, який застосовується глобально до всього інтерпретатора). Таким чином, може виникнути враження, що приросту в продуктивності

взагалі не варто було очікувати, але це не зовсім так. Справа в тім, що, дійсно, одночасно може виконуватись лише один потік, але задача, яку ми розв’язуємо опирається не тільки на швидкодію процесора комп’ютера, а й на взаємодію з різними веб-ресурсами. Відправка даних на веб-сервери та отримання з них відповіді займають значну частину часу необхідного на парсинг. Тому виконуючи скрапінг різних сайтів паралельно ми, якраз, і можемо зменшити цей час. Тобто коли один з парсерів, наприклад, перебуває в очікуванні відповіді від веб-сервера інтерпретатор Python передає управління потоку іншого парсера. Також, варто зазначити, що як альтернативу до багатопотокового програмування можна було б використати багатопроцесне, але на практиці отримані прискорення виявились незначними, що може бути зумовлено тим, що процеси потребують більше ресурсів для комунікації між собою, а частка задач, що залежать від вводу/виводу (запит клієнта/відповідь сервера) досить значна.

Але якби на отримані часові показники впливала тільки вищеописана проблема, ми б отримали пропорційне зменшення в часі виконання скрапінгу. Другою проблемою є «вузьке місце», тобто певні сайти потребують більше часу на взаємодію з ними, ніж інші, а оскільки паралельний парсинг відбувається в рамках групи матчів, то кожного разу потоки мають дочекатися «відстаючих». Щоб розв’язати цю проблему треба зрозуміти різницю між такими інструментами, як BeautifulSoup [2] та Selenium [3]. Перший з них будує дерево синтаксичного розбору вихідного коду сторінки і на цьому взаємодія з веб-сторінкою завершується. Далі всі такі дії, як пошук веб-елемента або «витягнення» його тексту відбуваються використовуючи побудовану структуру даних. Другий же інструмент є більш потужним та постійно взаємодіє з браузером, оновлюючи представлення дерева сторінки, що сповільнює виконання аналогічних розглянутим операціям. Звичайно, використання Selenium [3] не уникнути

на етапі рендерингу сторінки, або при необхідності натиснути на певний веб-елемент, але звузивши його використання до лише подібних цілей, можна досягти доволі значного приросту в продуктивності парсингу. Застосувавши таку оптимізацію до «вузьких місць», отримаємо час парсингу для 20 футбольних матчів рівний 152 секундам (Таблиця 3, оптимізована реалізація), що означає, що в порівнянні з початковими вимірами нам вдалося досягти десятикратного прискорення. А час потрібний на скрапінг тисячі футбольних матчів складе тепер 90 хвилин (саме тому, до речі, запуск повторного пошуку арбітражних ставок компонентом scheduler відбувається кожні 2 години).

ВИСНОВКИ

В результаті роботи було проаналізовано роботу букмекерських компаній, побудовано математичну модель для грального бізнесу, що дозволяє встановити можливість прибутку в тій чи іншій ситуації, його величину та максимізувати цю величину. Дану модель було реалізовано в програмному забезпеченні, за допомогою якого результати можна швидко, оперативно та автоматично отримувати в Telegram від бота у зручному Excel-форматі. Було реалізовано та застосовано в системі авторський алгоритм агрегації даних. Бота розгорнуто на безкоштовному сервері хмарної платформи Heroku, що дозволяє отримати необхідну інформацію в будь-який момент часу. Було реалізовано збереження налаштувань користувачів використовуючи базу даних MySQL [5].

Отримані результати повністю відповідають сучасному рівню технічних та наукових знань. Для розробки було використано найпопулярніші та досить потужні інструменти. На даний момент не існує точних аналогів створеного програмного забезпечення (хоча є схожі платні продукти, що не використовують телеграм-ботів, наприклад, AllBestBets), а побудована математична модель є розширенням існуючої [17][18][19][20][21] для M-вимірного випадку, всі визначення, твердження та доведення в ній є оригінальними.

Результати роботи пропонується використовувати в сфері грального бізнесу як самим букмекерським компаніям, так і їх клієнтам. На практиці ситуації з від'ємною відносною маржею з'являються кожного дня і тримаються від декількох хвилин до декількох тижнів.

В створеного програмного продукту надзвичайно великі перспективи розвитку, як, наприклад, збільшення кількості опрацьовуваних букмекерів, та дисциплін, на які пропонуються ставки. Також можливе створення компоненту, що відповідав би за моделювання користувацької активності

при веб-скрапінгу, створення веб-сервісу для можливості розповсюдження результатів роботи системи через API, також можливе створення веб-додатку, що надавав би альтернативний до телеграм бота інтерфейс користувача для взаємодії з продуктом. Іншими можливостями для розширення є опрацювання ставок на події, що відбуваються в режимі реального часу, побудова графіків зміни коефіцієнтів та їх аналіз.

Математична модель наведена в роботі має достатню наукову значимість для проведення подальших її досліджень, в той час як науково-технічна значимість роботи полягає у ефективному (парсинг сайтів часто потребує великої кількості часу) та непомітному (сайти мають захист від атак та з часом починають сповільнювати чи блокувати запити) веб-парсингу і також потребує створення нових алгоритмів та покращення існуючих, а також у точній агрегації даних, яка є ключовим елементом успішної роботи подібних систем. Що ж стосується соціально-економічної складової, масове використання розробленого або схожих продуктів потягне за собою балансування коефіцієнтів, що пропонуються букмекерськими конторами та підтриманню додатної маржі множини всіх існуючих букмекерів відносно будь-якої події.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Кингснорт С. Стратегия цифрового маркетинга / С. Кингснорт — М.: Олимп-Бизнес, 2019. — 416 с. — ISBN 978-5-6040010-2-8.
2. Crummy. BeautifulSoup Documentation [Електронний ресурс]. — Режим доступу до ресурсу: https://www.crummy.com/software/beautifulsoup/bs4/doc/.
3. Selenium. Selenium Documentation [Електронний ресурс]. — Режим доступу до ресурсу: https://selenium-python.readthedocs.io/.
4. pyTelegramBotAPI source code [Електронний ресурс]. — Режим доступу до ресурсу: https://github.com/eternnoir/pyTelegramBotAPI/tree/master/telebot.
5. MySQL. MySQL Documentation [Електронний ресурс]. — Режим доступу до ресурсу: https://dev.mysql.com/doc/.
6. Xlsxwriter. Xlsxwriter Documentation [Електронний ресурс]. — Режим доступу до ресурсу: https://xlsxwriter.readthedocs.io/.
7. Heroku. Heroku Documentation [Електронний ресурс]. — Режим доступу до ресурсу: https://devcenter.heroku.com/categories/reference.
8. История букмекерских контор [Електронний ресурс]. — Режим доступу до ресурсу: https://historygames.ru/azartnyie-igryi/istoriya-bukmekerskih-kontor.html.
9. Compilers: principles, techniques, and tools / Aho A.V., Sethi R., Ullman J.D. — Boston: Addison-Wesley Longman Publishing, 1986. — 796 p.
10. Dynamic Inverted Indexes for a Distributed Full-Text Retrieval System MultiText Project Technical Report / Charles L. A. C., Gordon V. C. — Waterloo: University of Waterloo Press, 1995. — 13 p.

11. Кулагина О. С. О современном состоянии машинного перевода // Математические вопросы кибернетики — М.: Наука, 1991. — вып. 3. — С. 5–50. — ISBN 5-02-014323-5.
12. Building natural language generation systems / Dale Robert, Reiter Ehud — Cambridge, U.K.: Cambridge University Press, 2000. — 250 p. — ISBN 978-0-521-02451-8.
13. Aycock J. JIT Compilation Techniques / J. Aycock — Calgary: University of Calgary Press, 2003. — 98 p.
14. Гладкий А. В. Формальные грамматики и языки / А. В. Гладкий — М.: Наука, 1973. — 368 с.
15. Scrapy documentation [Електронний ресурс]. — Режим доступу до ресурсу: https://docs.scrapy.org/en/latest/.
16. Python requests documentation [Електронний ресурс]. — Режим доступу до ресурсу: https://docs.python-requests.org/en/master/.
17. bettingexpert Academy. What Is An Arbitrage Bet [Електронний ресурс]. — Режим доступу до ресурсу: https://www.bettingexpert.com/academy/advanced-betting-theory/arbitrage-betting.
18. Фаворит. Что такое арбитражный беттинг [Електронний ресурс]. — Режим доступу до ресурсу: https://favoritnr1.com/39509-chto-takoe-arbitrazhnyiy-betting.html.
19. Avery J. S. Arbitrage in the European Soccer Betting Market [Електронний ресурс]. — Режим доступу до ресурсу: https://economics.yale.edu/sites/default/files/files/Undergraduate/Nominated%20Senior%20Essays/2015-16/Schwartz_Avery_SeniorEssay%202016.pdf.
20. Surebet. Surebets [Електронний ресурс]. — Режим доступу до ресурсу: https://en.surebet.com/site/faq/surebets.

21. Metaratings. Математика букмекерских вилок [Електронний ресурс]. — Режим доступу до ресурсу: https://meta-ratings.com.ua/vilki/matematika-bukmekerskikh-vilok/.
22. Telegram. Telegram Bot API Documentation [Електронний ресурс]. — Режим доступу до ресурсу: https://core.telegram.org/bots/api.
23. heroku-buildpack-google-chrome source code [Електронний ресурс]. — Режим доступу до ресурсу: https://github.com/heroku/heroku-buildpack-google-chrome.
24. heroku-buildpack-chromedriver source code [Електронний ресурс]. — Режим доступу до ресурсу: https://github.com/heroku/heroku-buildpack-chromedriver.
25. Heroku. ClearDB MySQL [Електронний ресурс]. — Режим доступу до ресурсу: https://devcenter.heroku.com/categories/reference.
26. Python regular expressions documentation [Електронний ресурс]. — Режим доступу до ресурсу: https://docs.python.org/3/library/re.html.