Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему *Автоматизований моніторинг оголошень онлайн-ринку вживаних авто засобами інтелектуального аналізу даних*

Галузь знань *12 – Інформаційні технології*
Шифр і назва галузі знань

Спеціальність *122 – Комп'ютерні науки*
Шифр і назва спеціальності

Освітня програма *Комп'ютерні науки*
Назва освітньої програми

Виконав: *студент 4 курсу, група КН-19-2* (Курс, група виконавця) _____ (Підпис) *І.О. Ціцьвіра* (Ініціали, прізвище)

Керівник: *док. філ., ст. викладач кафедри КН* (Науковий ступінь, посада) _____ (Підпис) *П.М. Радюк* (Ініціали, прізвище)

Нормоконтроль: *к.т.н., доцент кафедри КН* (Науковий ступінь, посада) _____ (Підпис) *Р.О. Багрій* (Ініціали, прізвище)

До захисту допускаю:
Зав. кафедри КН, д.т.н., професор _____ (Підпис) *О.В. Бармак* (Ініціали, прізвище)
05 06 2023 р.

Хмельницький 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет *інформаційних технологій*
Кафедра *комп'ютерних наук*
Освітній ступінь *бакалавр*
Галузь знань *12 – Інформаційні технології*
Спеціальність *122 – Комп'ютерні науки*
Освітня програма *освітньо-професійна програма підготовки бакалавра*

ЗАТВЕРДЖУЮ
Завідувач кафедри *комп'ютерних наук*

(підпис)
д.т.н., професор О.В. Бармак
«06» 03 2023 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

1. Тема кваліфікаційної роботи бакалавра: *«Автоматизований моніторинг оголошень онлайн-ринку вживаних авто засобами інтелектуального аналізу даних»*

2. Завдання видано студену *Ціцьвірі Іміру Олеговичу*
(прізвище, ім'я, по батькові)

3. Керівник роботи *ст. викладач кафедри КН Радюк Павло Михайлович*
(посада, прізвище, ім'я, по батькові)

4. Затверджено наказом університету від «06» 03 2023 р. № 5

5. Дата видачі завдання студенту: «03» 03 2023 р.

6. Зміст пояснювальної записки (перелік задач) та вихідні дані:

Провести аналіз онлайн-платформ для розміщення оголошень онлайн-ринку вживаних авто України; провести аналіз відомих методів та засобів інтелектуального аналізу даних для отримання та оброблення даних з онлайн-платформ; обрати найкращий засіб для розв'язання задачі автоматизованого моніторингу оголошень; реалізувати обраний засіб у вигляді програмного боту; провести експериментальне тестування програмного бота в реальних умовах; вихідними даними є інформація щодо оголошень онлайн-ринку вживаних авто.

7. Календарний план виконання кваліфікаційної роботи бакалавра:

№	Назва етапів (розділів) кваліфікаційної роботи бакалавра	Термін виконання	Примітка
1	*Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи бакалавра з керівником*	*грудень 2022*	виконано
2	*Ознайомлення з предметною областю, формулювання мети та задач дослідження, визначення об'єкта та предмета дослідження*	*січень 2023*	виконано
3	*Робота над розділом 1 – Дослідження відомих методів, засобів та технологій моніторингу оголошень ринку вживаних авто*	*січень 2023*	виконано
4	*Робота над розділом 2 – Проєктування програмного бота для автоматизованого моніторингу оголошень онлайн-ринку вживаних авто*	*березень 2023*	виконано
5	*Робота над розділом 3 – Програмна реалізація програмного боту з використанням обраного способу оброблення даних*	*квітень 2023*	виконано
6	*Оформлення пояснювальної записки згідно вимог*	*травень 2023*	виконано
7	*Підготовка статті до публікації у журналі, попередній захист кваліфікаційної роботи бакалавра*	*травень 2023*	виконано
8	*Захист кваліфікаційної роботи бакалавра на засіданні Екзаменаційної комісії*	*червень 2023*	виконано

Виконавець: *студент 4 курсу, група КН-19-2* (Курс, група виконавця) _____ (Підпис) *І.О. Ціцьвіра* (Ініціали, прізвище)

Керівник: *док. філ., ст. викладач кафедри КН* (Науковий ступінь, посада) _____ (Підпис) *П.М. Радюк* (Ініціали, прізвище)

Анотація

Тема кваліфікаційної роботи бакалавра: *Автоматизований моніторинг оголошень онлайн-ринку вживаних авто засобами інтелектуального аналізу даних*

Виконавець кваліфікаційної роботи бакалавра: *студент групи КН-19-2 Ціцьвіра Імір Олегович*

Керівник кваліфікаційної роботи бакалавра: *доктор філософії, ст. викладач кафедри КН Радюк Павло Михайлович*

Кваліфікаційна робота бакалавра містить:

Пояснювальна записка				Кількість додатків
Сторінок	Рисунків	Таблиць	Джерел інформації	
63	27	8	27	2

Метою кваліфікаційної роботи бакалавра є підвищення ефективності процесу моніторингу оголошень онлайн-ринку вживаних авто України.

Зміст пояснювальної записки (перелік задач) та вихідні дані: провести аналіз онлайн-платформ; провести аналіз відомих методів та засобів інтелектуального аналізу даних для отримання та оброблення даних з онлайн-платформ та обрати найкращий; реалізувати обраний засіб у вигляді програмного боту; провести експериментальне тестування програмного бота в реальних умовах; вихідними даними є інформація щодо оголошень онлайн-ринку вживаних авто.

Досягнення мети кваліфікаційної роботи бакалавра полягає у створенні програмного боту системи обліку миттєвими повідомленнями, використання якого дасть змогу підвищити ефективність процесу моніторингу оголошень онлайн-ринку вживаних авто України.

Ключові слова: ринок вживаних авто, онлайн-платформа, інтелектуальний аналіз даних, оброблення та аналіз даних, програмний бот.

Виконавець: *студент 4 курсу, група КН-19-2* _____ *І.О. Ціцьвіра*

Курс, група виконавця | Підпис | Ініціали, прізвище

Зміст

Перелік скорочень

Скорочення, термін, позначення	Пояснення
API	Application Programming Interface
CSV	Comma – separated values
URL	Uniform Resource Locator
ДТП	Дорожньо – транспортна пригода
ІАД	Інтелектуальний аналіз даних
КН	Комп’ютерні науки
КРБ	Кваліфікаційна робота бакалавра
ЛФП	Лакофарбове покриття
ХНУ	Хмельницький національний університет

Вступ

Кваліфікаційна робота бакалавра присвячена розв’язанню задачі автоматизованого моніторингу оголошень онлайн-ринку вживаних авто засобами ІАД.

Актуальність. В сучасному світі автомобілі стали невід’ємною частиною повсякденного життя, відіграючи ключову роль у транспортній системі та способі життя багатьох людей. В останні роки, з розвитком інтернету та електронної комерції, онлайн-ринок вживаних автомобілів став більш актуальним та доступним для широкого кола користувачів. Однак, такий ринок супроводжується великою кількістю оголошень, що створює складність у аналізі та виборі оптимального автомобіля. Враховуючи це, автоматизація процесів моніторингу та аналізу даних на онлайн-ринку вживаних авто стає все більш актуальною та важливою задачею.

Онлайн-ринок вживаних автомобілів має величезний обсяг даних, який включає тисячі оголошень, що щоденно з’являються та оновлюються. Крім того, автоматизований моніторинг та аналіз даних дозволяє швидко відстежувати зміни на ринку, виявляти нові тенденції та реагувати на них вчасно. Це забезпечує об’єктивність аналізу, оскільки ІАД виключає можливість суб’єктивного сприйняття та похибок, пов’язаних з людським фактором.

Отже, впровадження автоматизованого моніторингу засобами ІАД у сфері онлайн-ринку вживаних автомобілів є важливим кроком для забезпечення більшої ефективності, оперативності та адаптивності до змін.

Об’єкт дослідження – процес моніторингу оголошень онлайн-ринку вживаних авто України.

Предмет дослідження – методи, засоби та технології ІАД для отримання та оброблення даних з онлайн-платформ.

Мета кваліфікаційної роботи бакалавра – підвищення ефективності процесу моніторингу оголошень онлайн-ринку вживаних авто України.

Завдання кваліфікаційної роботи бакалавра – для досягнення поставленої мети визначено наступні завдання:

1. Провести аналіз онлайн-платформ для розміщення оголошень онлайн-ринку вживаних авто України.

2. Провести аналіз відомих методів та засобів ІАД для отримання та оброблення даних з онлайн-платформ та обрати найкращий для розв’язання задачі автоматизованого моніторингу оголошень.

3. Реалізувати обраний засіб ІАД у вигляді програмного боту системи обліку миттєвими повідомленнями для розв’язання задачі автоматизованого моніторингу оголошень.

4. Провести експериментальне тестування програмного бота в реальних умов

Розділ 1 Дослідження відомих методів, засобів та технологій моніторингу оголошень ринку вживаних авто

1.1 Аналіз ринку вживаних авто України

Потреба в особистому транспорті завжди була актуальною, оскільки більшість людей хотіли б знайти автомобіль, що відповідає їхнім бажанням і вимогам, а також різним параметрам, таким як ціна, потужність та комфорт. Український ринок автомобілів протягом останніх кількох років показував зростання онлайн-попиту на вживані авто, і незважаючи на негативний вплив війни та карантинних обмежень, він продовжує розвиватися (рисунок 1.1).

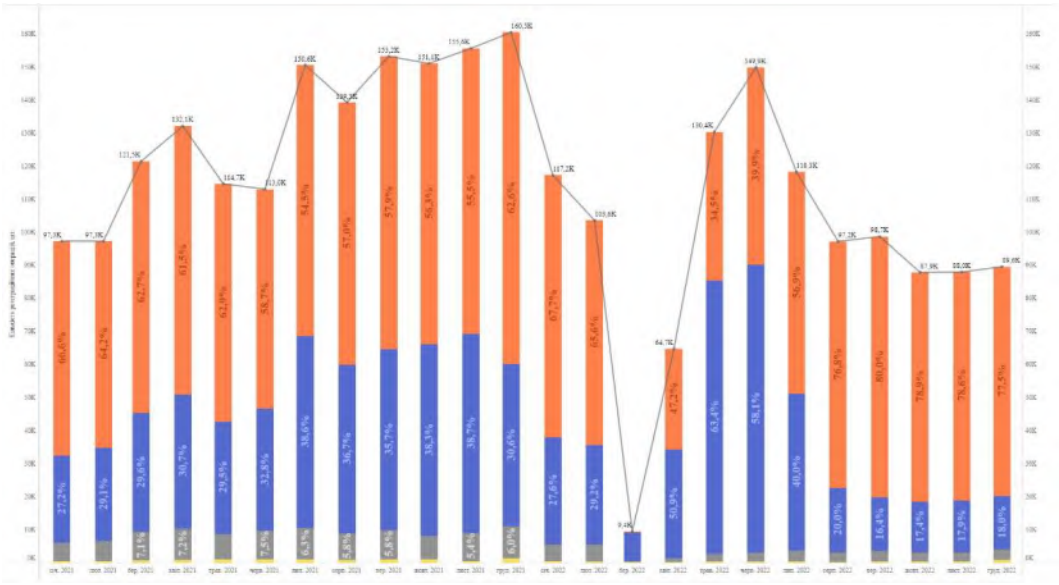


Рисунок 1.1 – Кількість реєстраційних операцій на авторинку України [1]

У світлі цієї тенденції, українці продовжують шукати ідеальний автомобіль, який відповідає їхнім потребам та бажанням. Враховуючи різноманітність параметрів, таких як бренд, модель, тип кузова та двигун, споживачі активно звертаються до онлайн-ресурсів для дослідження та знаходження найкращого варіанту для себе. Ринок автомобілів в Україні продовжує розвиватися, адаптуючись до змінних обставин та забезпечуючи споживачам широкий вибір транспортних засобів [2].

Кожна людина має індивідуальні потреби та вимоги до авто, але є критерії важливі переважній більшості:

– Марка авто – кожна марка має свою репутацію, історію та характеристики. Марка також може впливати на вартість, надійність та доступність запчастин.

– Модель авто – визначає дизайн, технічні характеристики, розміри та функціональність автомобіля.

– Рік випуску – вказує на вік автомобіля, що впливає на його зношеність, технічний стан та потенційну вартість на вторинному ринку.

– Пробіг – відображає загальний рух автомобіля та може свідчити про його зношеність, можливі проблеми з двигуном або ходовою частиною.

– Тип кузова – визначає практичність та естетичні аспекти автомобіля.

– Регіон – дозволяє з’ясувати історію автомобіля, враховуючи можливі регіональні різниці в умовах експлуатації та погодних умов.

– Кількість власників – чим менше власників, тим більш ймовірно, що автомобіль був доглянутий та відповідно обслуговувався.

– Об’єм двигуна – впливає на потужність, економічність та екологічність автомобіля.

– Модель двигуна – вказує на технічні характеристики, надійність та доступність запчастин для ремонту.

– Стан ЛФП – важливий для визначення зовнішнього стану автомобіля, можливих пошкоджень, корозії або потреби в додаткових ремонтних роботах.

– Технічний стан – в цілому описує надійність автомобіля, наявність неполадок, потребу в ремонті та потенційні витрати на утримання.

– Тип палива – впливає на економічність, екологічність та доступність палива.

– Тип коробки передач – може впливати на комфорт керування автомобілем, вартість обслуговування та енергоефективність.

– Колір – відіграє значну роль у зовнішньому вигляді автомобіля та впливає на його естетичність.

– Кондиціонер – важливий для комфорту водія та пасажирів.

– Скло підйомники – забезпечують зручність управління вікнами та підвищують комфорт користування автомобілем.

– Матеріали салону – впливають на комфорт, естетику та довговічність салону автомобіля. Вибір між тканиною, шкірою або іншими матеріалами відображає вимоги покупця до комфорту та дизайну.

– Підсилювач керма – забезпечує легкість керування автомобілем, особливо при низьких швидкостях або при паркуванні, полегшуючи маневрування та підвищуючи комфорт водія.

– Участь в ДТП – ця інформація важлива для оцінки стану автомобіля та можливих прихованих пошкоджень.

– Ціна – одна з ключових характеристик при виборі автомобіля, яка відображає фінансові можливості покупця та його бажання витратити певну суму на придбання авто.

Ринок вживаних авто в Україні показував тенденцію до зростання в останні роки [3]. Однією з причин є зниження імпортних зборів на вживані автомобілі, що сприяло збільшенню кількості доступних моделей та марок. Крім того, економічні обставини змушують українців шукати доступніші варіанти транспортних засобів, у тому числі вживані автомобілі.

Ціни на вживані автомобілі можуть суттєво відрізнятися залежно від марки, моделі, року випуску, стану автомобіля та інших факторів. Однак, середня вартість вживаного автомобіля в Україні може коливатися від 2 000 до 11 000 доларів США (рисунок 1.2).

Рисунок 1.2 – Цінова структура пропозиції на авторинку України [4]

Україна вважаєть країною з досить низьким рівнем автомобілізації, але попри це та враховуючи зростання ринку вживаних авто в Топ країн за рівнем автомобілізації продемонстровано на рисунку 1.3.

Рисунок 1.3 – Топ країн за рівнем автомобілізації [4]

З огляду на стан авто-ринку можуть з’явитися нові можливості для ринку вживаних автомобілів, такі як підвищення стандартів якості та розвиток сервісів післяпродажного обслуговування. Зокрема, це може стосуватися створення онлайн-платформ, які дозволять покупцям легко порівнювати моделі та ціни, отримувати рекомендації щодо покупки, а також забезпечити безпечні та прозорі умови угод [5, 6].

З урахуванням усього вищезгаданого, зростання ринку вживаних авто в Україні є прогнозовним [7], можна очікувати його подальшого розвитку та розширення. Використання інформаційних технологій та онлайн-платформ може допомогти забезпечити більш прозорий та надійний процес купівлі-продажу. На фоні цього розробка системи автоматизованого моніторингу онлайн-ринку вживаних авто засобами ІАД набуває особливої актуальності.

1.2 Огляд онлайн-платформ для розміщення оголошень онлайн-ринку вживаних авто

Існує ряд онлайн-платформ, для розміщення оголошень онлайн-ринку вживаних авто. Найпопулярнішими в Україні є AUTO.RIA, RST та OLX [8], що можемо побачити на рисунку 1.4.

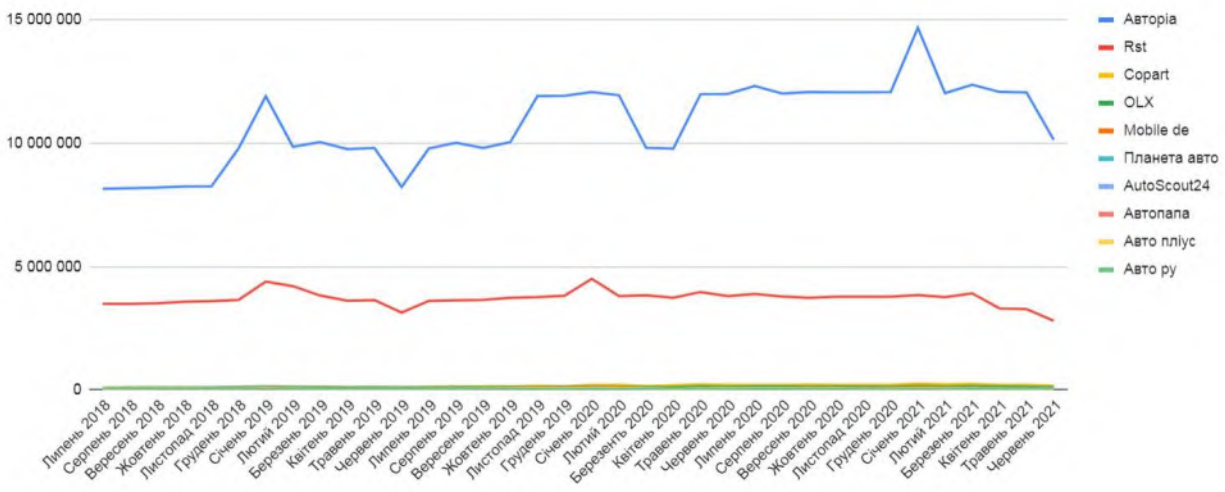


Рисунок 1.4 – Динаміка онлайн-попиту на авто агрегатори [9]

AUTO.RIA [10] це веб-сайт, який спеціалізується на продажі та купівлі автомобілів в Україні. На цьому сайті можна знайти широкий асортимент автомобілів, включаючи нові та б/у машини різних марок та моделей. AVTO.RIA також пропонує послуги фінансування, страхування та сервісу автомобілів. Даний сервіс має переваги ти недоліки (таблиця 1.1)

Таблиця 1.1 – Переваги та недоліки сайту AUTO.RIA

Переваги	**Недоліки**
Зручний пошук. Фільтри та опції сортування дозволяють знайти автомобіль, що відповідає потребам користувача.	Незручність спілкування між користувачами. Значні недоліки в системі спілкування між покупцями та продавцями
Інформативність. Сайт містить корисну інформацію, таку як огляди автомобілів, новини, поради.	Відсутність персональних рекомендацій. Сайт не пропонує автоматичних рекомендацій на основі інтересів користувача.
Сервіс оцінки автомобілів. Можливість отримати орієнтовну вартість автомобіля.	Недостовірність деяких оголошень. Можливість зустрічі з нечесними або шахрайськими оголошеннями.
Мобільний додаток. Доступ до сайту та його функцій на смартфонах через додатки для Android та iOS.	Високий рівень конкуренції для продавців. Через велику кількість оголошень, може бути важко привернути увагу до конкретного авто.
Сервіс порівняння авто. Користувачі можуть порівнювати різні моделі автомобілів для кращого вибору.	Не всі оголошення містять повну інформацію про авто. Деякі продавці можуть надати недостатньо деталей про автомобіль.

Сайт AUTO.RIA є одним з найбільших і найпопулярніших в Україні в галузі продажу автомобілів. Він надає користувачам можливість швидко та зручно знайти потрібний автомобіль за допомогою розширеної системи пошуку, яка включає такі параметри, як марка, модель, рік випуску, тип кузова, об’єм двигуна, тип палива, колір, пробіг та багато іншого.

Крім того, на сайті можна знайти новини автомобільного ринку, огляди нових моделей, відгуки власників автомобілів, а також корисні поради щодо покупки, експлуатації та обслуговування автомобіля.

Rst.ua (також відомий як RST) [11] – це український інтернет-ресурс, що спеціалізується на продажу та оренді автомобілів, мотоциклів, комерційних транспортних засобів та запчастин до них. Rst.ua є однією з провідних онлайн-платформ в Україні для пошуку транспортних засобів. Загалом Rst.ua є дуже корисною інтернет-платформою для тих, хто шукає автомобіль, мотоцикл або запчастину до них в Україні, і має значну кількість користувачів, що свідчить про її популярність та надійність. Rst.ua – це зручний та надійний ресурс для пошуку транспортних засобів та запчастин до них в Україні, які має переваги ти недоліки (таблиця 1.2).

Таблиця 1.2 – Переваги та недоліки сайту RST.ua

Переваги	**Недоліки**
Великий вибір автомобілів. Ресурс пропонує велику кількість оголошень про продаж автомобілів.	Реклама. Сайт може містити рекламу, яка відволікає користувачів.
Розширені функції пошуку. Користувачам доступні детальні фільтри для пошуку потрібного авто.	Недостовірність деяких оголошень. Можливість зустрічі з нечесними або шахрайськими оголошеннями.
Інформативність. Сайт надає корисну інформацію про авто, включаючи деталі, фото та контактну інформацію продавця.	Відсутність належного перекладу. Сайт працює переважно українською мовою, що може утруднювати використання користувачами, які не володіють цією мовою.
Можливість безпосереднього контакту з продавцем. Користувачі можуть безпосередньо зв’язатися з продавцями для отримання додаткової інформації або підтвердження даних.	Нестабільність роботи сайту. Користувачі можуть зіткнутися з проблемами доступу до сайту або його функцій.

OLX.Auto [12] – це розділ на платформі OLX, який спеціалізується на продажу транспортних засобів, таких як автомобілі, мотоцикли, велосипеди,

скутери, а також автозапчастин і аксесуарів для них. На OLX.Auto користувачі можуть розміщувати оголошення про продаж або купівлю транспортних засобів, описуючи основні характеристики, такі як марка, модель, рік випуску, пробіг та інші параметри. OLX.Auto також пропонує різні фільтри для пошуку, щоб допомогти користувачам знайти транспортний засіб за вказаними параметрами, такими як рік випуску, ціновий діапазон, тип кузова тощо. OLX.Auto є популярним сервісом для продажу транспортних засобів в Україні та деяких інших країнах, де присутня платформа OLX. Але попри все, сайт має свої переваги ти недоліки (таблиця 1.3)

Таблиця 1.3 – Переваги та недоліки сайту OLX

Переваги	**Недоліки**
Функція повідомлень, яка дозволяє покупцям і продавцям спілкуватися безпосередньо на платформі.	Сайт може містити рекламу, яка відволікає користувачів.
Можливість залишати відгуки та оцінки продавця, що забезпечує додаткову інформацію для майбутніх покупців.	Відгуки та оцінки можуть бути не завжди об’єктивними або корисними.
Постійне оновлення бази оголошень, завдяки чому можна знайти свіжі пропозиції.	Оголошення можуть швидко зникати, що може бути незручним для покупців.
Можливість продавати і купувати не тільки автомобілі, а й інші товари та послуги.	Якість фотографій і детальність опису товару може сильно варіюватися між оголошеннями.
Можливість створювати списки обраних оголошень для подальшого порівняння або відслідковування.	Не всі продавці дотримуються правил платформи, що може призвести до неприємних ситуацій.

При виборі оптимального джерела даних для автоматизованого моніторингу онлайн-ринку авто, важливо враховувати специфіку та індивідуальні характеристики різних ресурсів. AUTO.RIA, наприклад, представляє собою високоспеціалізований ресурс із розширеними функціями пошуку, який може бути дуже корисним для детального аналізу автомобільного ринку [13]. Тим часом, OLX та RST.ua являють собою більш загальні платформи оголошень.

1.3 Аналіз відомих методів та засобів ІАД для отримання та оброблення даних з онлайн-платформ

ІАД визначається як систематичний процес ідентифікації, інтерпретації, узагальнення та вилучення корисної інформації з первинних даних, що має на меті підтримати прийняття обґрунтованих рішень [14]. Цей аналіз створений на засадах використання різноманітних методів та технологій, які включають статистичний аналіз [15], машинне навчання, аналіз тексту [16], візуалізацію даних [5] та інші техніки для забезпечення високоякісного аналізу даних та виявлення кореляцій, закономірностей та відмінностей в розглядуваних даних.

ІАД призначений для автоматизації процесу обробки даних, що дозволяє проводити прогнозування, виявляти аномалії, класифікувати об’єкти та генерувати рекомендації [17]. Його застосування є різнорідним і охоплює широкий спектр галузей, включаючи, але не обмежуючись такими як маркетинг, фінанси, медицину, наукові дослідження, спортивну аналітику та багато інших.

Серед ключових етапів ІАД виділяють збір даних. Джерела даних можуть варіюватись від баз даних до файлів, онлайн-платформ, API, тощо.

У контексті збору даних одним з широко використовуваних методів є веб-скрапінг [18]. Він включає процес збору даних з веб-сайтів та їх збереження в необхідному форматі, як-от табличні структури, бази даних або текстові файли. Веб-скрапінг стає важливим інструментом, особливо коли дані не можна отримати через API. Варто зазначити, що розробка та використання веб-скраперів вимагає спеціалізованих інструментів [19].

Процес підготовки та попередньої обробки даних є важливим етапом в ІАД, який включає в себе очищення, трансформацію та інтеграцію зібраних даних для їх подальшого аналізу. Цей етап може включати такі операції, як видалення або заповнення відсутніх значень, об’єднання даних з різних джерел, кодування категоріальних змінних та нормалізація даних.

На наступному етапі відбувається власне аналіз даних, використовуючи обрані методи та моделі. Це може включати навчання моделей машинного

навчання, використання статистичних методів для виявлення закономірностей та залежностей, аналіз текстових даних для виявлення тем та сентиментів, використання методів кластеризації для групування схожих об’єктів, а також візуалізацію даних для наглядного представлення результатів аналізу.

Останнім етапом процесу ІАД є інтерпретація результатів аналізу та їх використання для підтримки прийняття рішень. Це може включати формулювання висновків на основі отриманих результатів, створення прогнозів, виявлення аномалій та виявлення можливих проблем та можливостей.

Використовуючи різні методи та інструменти ІАД, можуть виконувати складний аналіз даних, включаючи класифікацію, регресію, кластеризацію, зменшення розмірності та вибір моделі, для знаходження шаблонів, тенденцій, аномалій та взаємозв’язків в своїх даних.

Основні задачі, які можуть виникати у процесі аналізу даних [20]:

– Класифікація – визначення категорії об’єкта на основі його ознак.

– Регресія – передбачення числового значення на основі ознак об’єкта.

– Кластеризація – групування об’єктів за схожістю, без попередньої інформації про їх категорії.

– Зменшення розмірності даних вибір найбільш інформативних ознак або перетворення даних для спрощення аналізу.

– Оброблення даних – очищення, масштабування, кодування категорійних змінних, заповнення пропущених значень тощо.

– Розробка та навчання моделей – розробка та навчання вибраних моделей аналізу на основі підготовлених даних.

– Тестування та оцінка моделей – перевірка та оцінка якості та ефективності моделей на тестовому наборі даних, який не був використаний під час навчання моделей.

– Впровадження та моніторинг моделей – інтеграція моделей в бізнес-процеси або системи для отримання висновків та прийняття рішень. Забезпечення постійного моніторингу, оновлення та налаштування моделей, враховуючи нові дані та зміни у ситуації.

Попри технічну складність, ІАД даних може принести значні вигоди, а саме, в підвищенні ефективності, виявленні нових можливостей, зменшенні ризиків та підтримці інновацій [21]. Більше того, він може забезпечити важливі переваги в конкуренції, розкриваючи цінні висновки та прогнози, які можуть бути недоступні через традиційні методи аналізу.

У підсумку, ІАД є важливим елементом в стратегії багатьох сучасних організацій. Через свою здатність виявляти значущу інформацію в великих наборах даних, він відіграє вирішальну роль в розумінні та використанні даних для досягнення бізнес-цілей.

1.4 Мета, завдання та вимоги до реалізації програмного бота

Метою кваліфікаційної роботи бакалавра є підвищення ефективності процесу моніторингу оголошень онлайн-ринку вживаних авто України.

Для досягнення поставленої мети необхідно виконати наступні завдання:

1. Провести аналіз онлайн-платформ для розміщення оголошень онлайн-ринку вживаних авто України.

2. Провести аналіз відомих методів та засобів ІАД для отримання та оброблення даних з онлайн-платформ та обрати найкращий для розв’язання задачі автоматизованого моніторингу оголошень.

3. Реалізувати обраний засіб ІАД у вигляді програмного боту системи обліку миттєвими повідомленнями для розв’язання задачі автоматизованого моніторингу оголошень.

4. Провести експериментальне тестування програмного бота в реальних умовах.

Розділ 2 Проєктування програмного бота для автоматизованого моніторингу оголошень онлайн-ринку вживаних авто

2.1 Аналіз критеріїв моніторингу оголошень онлайн-ринку вживаних авто

Для того щоб почати моніторинг оголошень, користувач має обрати критерії за якими відбуватиметься сортування .

Аналіз критеріїв моніторингу оголошень ринку вживаних авто передбачає визначення ключових параметрів, на які користувачі звертають увагу під час пошуку автомобіля. Існує ряд параметрів, які слід враховувати у першу чергу при моніторингу оголошень про продаж транспортних засобів.

Початковим етапом у процесі пошуку транспортного засобу є визначення його типу. Даний критерій є важливим, оскільки він сприяє конкретизації діапазону пошуку, відкидаючи відразу невідповідні варіанти.

Другим релевантним параметром є вартість. Оскільки більшість користувачів мають обмеження у фінансових ресурсах, вилучення з області пошуку варіантів, що перевищують встановлені бюджетні рамки, є необхідною мірою.

Наступними значимими критеріями відбору є марка та модель автомобіля. Популярність та престижність марки можуть істотно впливати на рішення користувача. Крім того, різні моделі однієї марки можуть мати різні технічні характеристики та комплектації, що також варто враховувати при виборі.

Рік випуску автомобіля теж відіграє важливу роль, оскільки він впливає на ступінь зношеності автомобіля, його технологічні характеристики, а також ціну. Отже, цей критерій допомагає більш точно відповідати вимогам користувача.

Пробіг транспортного засобу – інший важливий параметр, який безпосередньо впливає на стан автомобіля та може свідчити про потенційну необхідність в майбутньому ремонту.

Географічне розташування автомобіля може впливати на зручність його огляду та придбання, тому варіанти, що не задовольняють вимоги користувача за цим критерієм, можуть бути відкинуті.

В контексті підбору критеріїв для пошуку, існує спектр додаткових параметрів, які, незважаючи на їх, здавалося б, вторинну роль, мають потенціал значно оптимізувати та підвищити ефективність процесу пошуку:

– Тип двигуна. Існує низка варіацій двигунів, включаючи дизель, бензин, газ, та електродвигуни. Кожен із цих типів має свої специфіки, які впливають на оперативні витрати автомобіля, вплив на довкілля, а також на експлуатаційні особливості.

– Об’єм двигуна. Об’єм двигуна являє собою важливий показник, який корелює з потужністю автомобіля та його паливною економічністю.

– Тип кузова. Варіанти вибору між седаном, хетчбеком, кросовером, універсалом та іншими типами кузова мають відображати специфічні потреби користувача, його стиль життя та уявлення про комфорт.

– Колір автомобіля. Параметр, який може мати значення з точки зору естетичних уподобань та сприйняття автомобіля.

– Стан автомобіля. Аналіз того, чи був автомобіль у аварії, які ремонти були проведені, який технічний стан автомобіля, а також наявність гарантій від виробника чи дилера можуть суттєво вплинути на рішення про покупку.

– Трансмісія. Вибір між механічною або автоматичною трансмісією може бути залежним від особистих переваг та звичок водія.

– Комплектація автомобіля. Наявність певних додаткових опцій та систем, таких як круїз-контроль, навігаційна система, підігрів сидінь, парктроніки та інше, може значно впливати на вартість автомобіля, а також на його привабливість для потенційного покупця.

– Власники. Кількість попередніх власників може свідчити про історію експлуатації автомобіля та його реальний стан, що має важливе значення при оцінці перспективного автомобіля.

– Споживання палива. Показник, який має особливу важливість для осіб, які планують активно використовувати автомобіль. Автомобілі з високим споживанням палива можуть стати причиною значних витрат у довгостроковій перспективі.

– Технічний стан. Об’єктивна оцінка технічного стану автомобіля, включаючи стан двигуна, гальмівної системи, підвіски та інших критично важливих компонентів, є важливою складовою при процесі прийняття рішення.

– Історія обслуговування. Документація, яка свідчить про проведене обслуговування та ремонтні роботи, може допомогти виявити потенційні проблеми та дає можливість оцінити, наскільки відповідально та якісно автомобіль був підтримуваний.

– Комфорт і зручність. Ці параметри, такі як простір для пасажирів та багажу, комфорт сидінь, якість звуку, навігаційна система та інші додаткові особливості, можуть суттєво вплинути на загальний досвід користувача і його задоволення від володіння автомобілем.

В рамках даного підрозділу було проведено детальний аналіз критеріїв моніторингу оголошень онлайн-ринку вживаних авто. Розглядаючи основні параметри, що впливають на процес вибору автомобіля, було встановлено, що кожний з цих критеріїв має свою вагу та значимість, яка може варіюватися в залежності від особистих пріоритетів кожного конкретного покупця. Тому в процесі пошуку автомобіля важливо правильно визначити основні потреби та вимоги, що дозволить найбільш ефективно використовувати ці критерії для вибору найкращого варіанту.

2.2 Спосіб автоматизованого моніторингу оголошень онлайн-ринку вживаних авто

Спосіб автоматизованого моніторингу оголошень на вторинному ринку автомобілів полягає у розробці комплексної системи, що забезпечує автоматичний збір, аналіз та розсилку інформації про продаж вживаних

автомобілів, враховуючи задані користувачем параметри а також визначення середньоринкової ціни та рекомендація ціни на основі прогнозу моделі машинного навчання .

Розглянемо основні етапи обробки даних у такій системі (рисунок 2.1).

Рисунок 2.1 – Схема роботи способу автоматизованого моніторингу оголошень онлайн-ринку вживаних авто

Етап 1. Створення профілю користувача:

1. Збір параметрів для пошуку. Отримання від користувача критеріїв через Telegram-бота, які використовуватимуться в якості фільтрів для оголошень.

2. Створення профіля користувача. Створення профіля користувача, у якому будуть зберігатися актуальні параметри для пошуку та статус пошуку.

3. Збереження параметрів для пошуку. Запис параметрів для пошуку у профіль користувача.

Етап 2. Запит до веб-ресурсу та збір списку оголошень:

1. Формування запиту. Отримання параметрів для пошуку з профілю користувача та запит до сайту з цими параметрами.

2. Парсинг нових оголошень. Зчитування та запис посилань на оголошення з списку оголошень отриманих за вказаним запитом.

Етап 3. Збір даних про оголошення

1. Прийом списку посилань. Прийом спуску посилань, отриманих на попередньому етапі.

2. Парсинг даних з оголошень. Парсинг даних оголошення для кожного елементу з списку посилань на оголошення. Отримання даних про оголошення, таких як марка, модель, ціна, фото, рік виготовлення, опис, пробіг, місце продажу.

Етап 4. Розрахунок середньої та рекомендованої ціни.

Цей етап має на увазі процес визначення оптимальної вартості продукту чи послуги на основі аналізу ринку. Він складається з трьох основних частин:

1. Прийом даних про оголошення. Отримання даних про оголошення, отриманих на попередньому етапі.

2. Пошук схожих оголошень та підрахунок середньої ціни по ринку. Після збору даних про оголошення, проводиться пошук схожих оголошень, що дозволяє оцінити виправданість ціни в оголошенні. Пошук схожих оголошень відбувається за оголошеннями з авто однієї маки та моделі , у яких різниця у даті випуску не більше 2 років та загальний пробіг не відрізняється більше ніж на 5%. Середня ціна по ринку Average price є середнім арифметичним значенням ціни серед списку оголошень з схожими параметрами, й обчислюється наступним чином :

$$\overline{P} = \frac{1}{N}\sum_{i=1}^{N} P_i \ , \tag{2.1}$$

де $\sum_{i=1}^{N} P_i$ – це сума цін усіх P_i схожих оголошень, N – загальна кількість оголошень.

Використовуючи ці дані, можна визначити середню ціну для схожих оголошень, що допомагає зрозуміти, яким має бути орієнтовна вартість для конкретного оголошення.

3. Використання моделі машинного навчання для прогнозування ціни на основі даних оголошення за формулою (2.1). Останнім кроком є застосування регресійної моделі машинного навчання для прогнозування ціни на основі зібраних даних. Модель аналізує характеристики оголошення, і видає рекомендовану ціну для оголошення. Цей етап допомагає користувачам зрозуміти, яким має бути вартість конкретного товару або послуги.

Завдяки реалізації цієї інформаційної технології, користувачі зможуть ефективно та зручно слідкувати за оголошеннями на ринку вживаних автомобілів та отримувати актуальні пропозиції відповідно до своїх інтересів та вимог.

Ефективність бота E (efficiency) оцінюється за допомогою простої метрики, що бере до уваги кількість оголошень, отриманих користувачем за одиницю часу, й обчислюється наступним чином :

$$E = \frac{N}{T}, \tag{2.2}$$

де N – кількість нових оголошень, які були відправлені користувачем за визначений період, T – час цього періоду в годинах.

Формула (2.2) дозволяє визначити, наскільки ефективно бот знаходить нові оголошення, що відповідають визначеним користувачем критеріям та вимірюється в оголошеннях на годину. Чим вище значення E, тим більш ефективно працює бот.

У даному підрозділі було проведено роботу над розробкою способу автоматизованого моніторингу оголошень онлайн-ринку вживаних авто. Розроблений спосіб забезпечує регулярний збір актуальної інформації та надсилання сповіщень користувачам, відповідно до їхніх параметрів пошуку.

Відповідно до результатів, розроблений спосіб не просто забезпечує користувачів актуальною інформацією про доступні оголошення, але й допомагає їм зрозуміти середню вартість автомобілів, які вони розглядають а також отримати рекомендовану ціну на конкретний автомобіль.

2.3 Функціональна структура програмного бота

Функціональну структуру Telegram-бота для автоматизованого моніторингу оголошень онлайн-ринку вживаних авто можна поділити на 4 функціональні компоненти (рисунок 2.2).

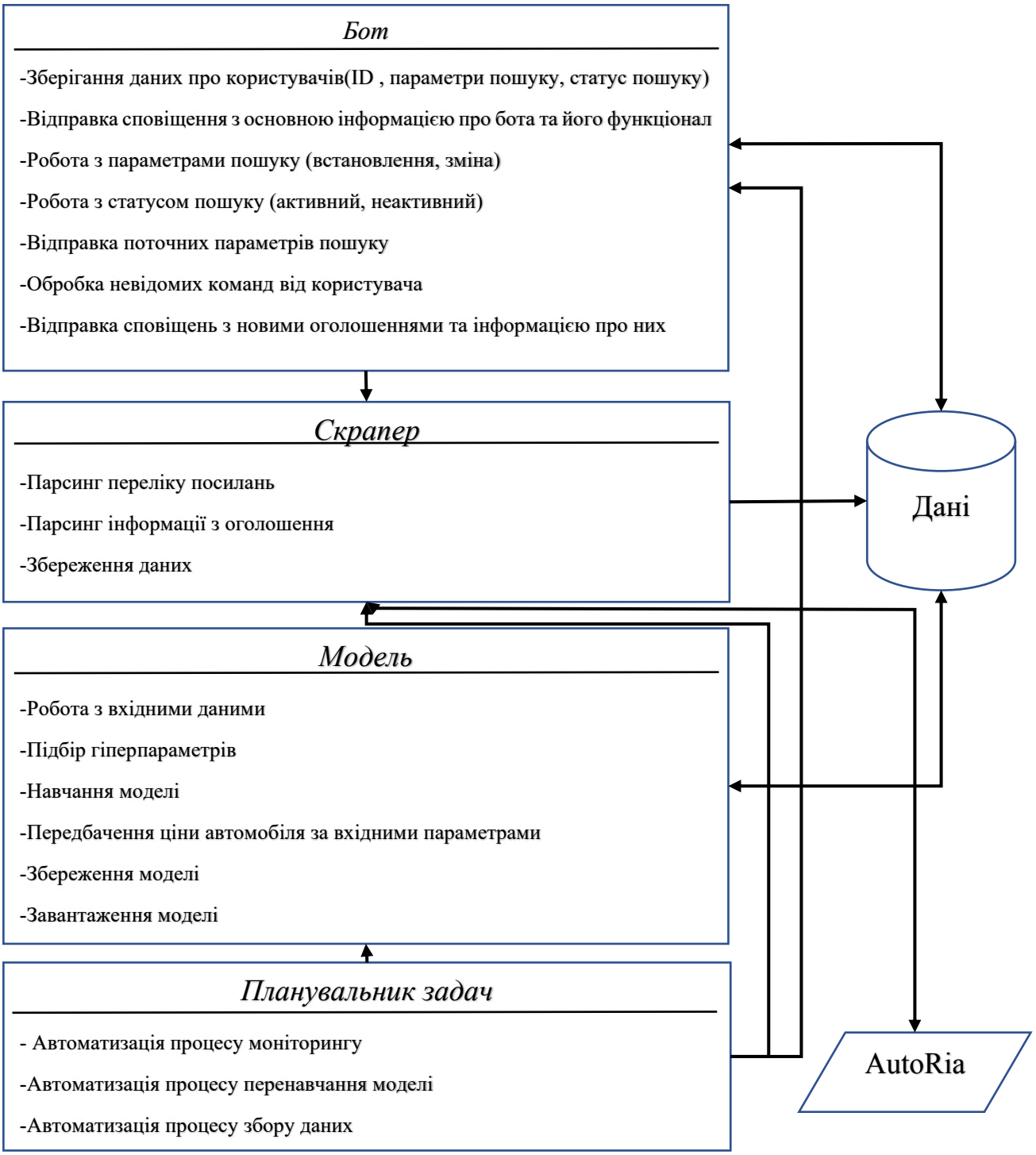


Рисунок 2.2 – Функціональна структура Telegram-бота для автоматизованого моніторингу оголошень онлайн-ринку вживаних авто

Бот – основний модуль для взаємодії з користувачем через інтерфейс Telegram. Цей модуль отримує вхідні повідомлення від користувача, обробляє їх і відправляє відповідні запити до інших функціональних модулів (Моделі, Скрапера, Планувальника задач). Після отримання відповідей, бот формує і відправляє повідомлення користувачу. Бот використовує Telegram API для взаємодії з користувачами. Він підтримує різні типи вхідних команд, такі як текстові повідомлення, команди, кнопки інтерфейсу тощо. Відповіді формуються на основі даних, отриманих від інших модулів, і можуть містити текст, зображення, відео, документи тощо.

Модель – цей модуль використовується для прогнозування ціни автомобіля. Він отримує дані про автомобіль (марка, модель, рік випуску, пробіг тощо), обробляє ці дані і використовує модель машинного навчання для передбачення ціни. Результат передається назад до модуля бота. Він включає різні етапи обробки даних, такі як очищення, кодування, нормалізація, а також тренування і валідація моделі та оптимізація гіперпараметрів.

Скрапер – цей модуль використовується для збору даних з сайту Auto.ria. Він періодично сканує сайт, збирає дані з оголошень (марка, модель, рік, пробіг, ціна тощо) і зберігає їх у базі даних. Ці дані можуть використовуватися модулем моделі машинного навчання для тренування моделі та покращення точності прогнозування. Він спроєктований таким чином, щоб враховувати структуру веб-сайту Auto.ria і забезпечувати коректний збір даних.

Планувальник задач – цей модуль слідкує за новими оголошеннями на сайті. Він перевіряє сайт на наявність нових оголошень кожну годину і, якщо виявляє нові оголошення за запитом користувача, відправляє сповіщення модулю бота. Бот, у свою чергу, повідомляє користувача про нові оголошення, які відповідають його інтересам. Також цей модуль запускає періодичний збір даних для навчання моделі та періодично запускає перенавчання моделі з врахуванням нових даних.

У результаті розробки функціональної структури бота було приділено увагу ефективності використання часу користувачів. Повідомлення про нові оголошення відправлятимуться користувачам, що дозволить їм бути в курсі всіх актуальних пропозицій і встигнути реагувати на них. Також модуль прогнозування цін допомагатиме користувачам робити обґрунтовані вибори і приймати виважені рішення.

Використовуючи сучасні технології, такі як машинне навчання, засоби ІАД і автоматизований моніторинг, є можливим створення бота, який не тільки вирішує задачу пошуку потрібних оголошень, але й надає користувачам цінну інформацію, прогнозовану на основі накопичених даних.

Описані вище функціональні компоненти дають змогу реалізувати потужного та зручного для використання бота для моніторингу оголошень онлайн-ринку вживаних авто. Завдяки такому боту користувачі зможуть швидко та легко знаходити відповідні оголошення відповідно до своїх критеріїв пошуку та отримувати сповіщення про нові оголошення. Це відкриває нові можливості для розвитку ринку вживаних автомобілів, сприяє зручності і комфорту користувачів, а також сприяє підвищенню їх задоволення від процесу вибору та покупки автомобіля.

2.4 Проєктування структури та інтерфейсу програмного бота

Автоматизований моніторинг оголошень за допомогою Telegram-бота значно підвищує продуктивність пошуку користувачами, звільняючи їх від необхідності вручну перевіряти велику кількість оголошень і витрачати значний час. Бот автоматично перевіряє оновлення на веб-сайті AUTO.RIA та надсилає користувачам оголошення, які відповідають їхнім критеріям пошуку. Це дозволяє користувачам заощадити значний час та зосередитися на найбільш перспективних пропозиціях.

Інтерфейс Telegram-бота, призначеного для автоматизованого моніторингу онлайн-ринку вживаних авто, а саме пошуку нових оголошень на онлайн-

платформі AUTO.RIA, є текстовим і складається з команд та повідомлень, які забезпечують взаємодію між користувачем та ботом.

Крім основних команд, Telegram-бот також надсилатиме користувачам системні повідомлення, які повідомлятимуть про статус бота, наприклад, коли пошук активовано чи деактивовано, або коли пошуковий запит було оновлено. Це забезпечує прозорість роботи бота та дозволяє користувачам краще контролювати процес моніторингу оголошень.

Бот, що виконує роль корисного інструменту з моніторингу потрібних користувачу оголошень, повинен мати здатність відслідковувати будь-який пошуковий запит від користувача. Його алгоритм передбачатиме щогодинну перевірку користувацьких запитів та надсилання нових пропозицій, якщо такі з’являться.

Основні команди інтерфейсу такого бота включатимуть:

– /start , це команда, яка викликатиметься при запуску бота, та надсилатиме користувачу вітальне повідомлення з основною інформацією про бота та детальною інструкцією з використання. Інструкція буде містити у собі перелік усіх наявних команд , їх опис та приклади їх використання;

– /set , це команда, яку користувач зможе використати для задання потрібних йому параметрів для пошуку авто;

– /get , це команда, яку користувач зможе використати, щоб отримати повідомлення від бота, у якому буде міститися його поточний пошуковий запит;

– /on , команда, яку користувач зможе використати, щоб змінити свій статус пошуку на активний;

– /off , команда, яку користувач зможе використати, щоб змінити свій статус пошуку на неактивний.

Ще однією важливою частиною інтерфейсу є повідомлення від бота , які під час роботи бот надсилає користувачу знайдені оголошення з детальним описом. Бот забезпечує систему сповіщень, яка повідомляє користувачів про нові оголошення, що відповідають їхнім критеріям. Кожне сповіщення містить детальну інформацію про автомобіль, яка включатиме:

– фото;

– марка;

– модель;

– ціна;

– пробіг;

– місцезнаходження;

– середня ціна по ринку;

– рекомендована ціна;

– опис;

– посилання на оголошення.

Така деталізація дозволяє користувачам зрозуміти, чи відповідає пропозиція їхнім вимогам, не відкриваючи оголошення на сайті.

У доповнення до опису інтерфейсу, бот також інтегрований з моделлю машинного навчання. Ця модель використовується для прогнозування ціни на автомобілі, оголошення про продаж яких з'явились на AUTO.RIA. Якщо розглядати послідовність взаємодій між користувачем, ботом Telegram, профілем користувача, веб-сайтом AUTO.RIA та моделлю машинного навчання в процесі автоматизованого моніторингу оголошень на ринку вживаних автомобілів (рисунок 2.3), то отримаємо більш загальну картину, яка відображає процес збереження параметрів пошуку, перевірки нових оголошень і прогнозування ціни за допомогою моделі машинного навчання

Взаємодія користувача, зазвичай, буде відбуватися за наступним алгоритмом:

1. Користувач починає взаємодію з ботом, надсилаючи команду «/start».

2. Бот відповідає, відправляючи користувачеві стартове повідомлення, яке, наприклад, може містити інструкції про те, як використовувати бота.

3. Користувач вводить команду «/set», щоб задати параметри пошуку, включаючи URL.

4. Бот зберігає цей пошуковий URL у профілі користувача, що знаходиться в базі даних.

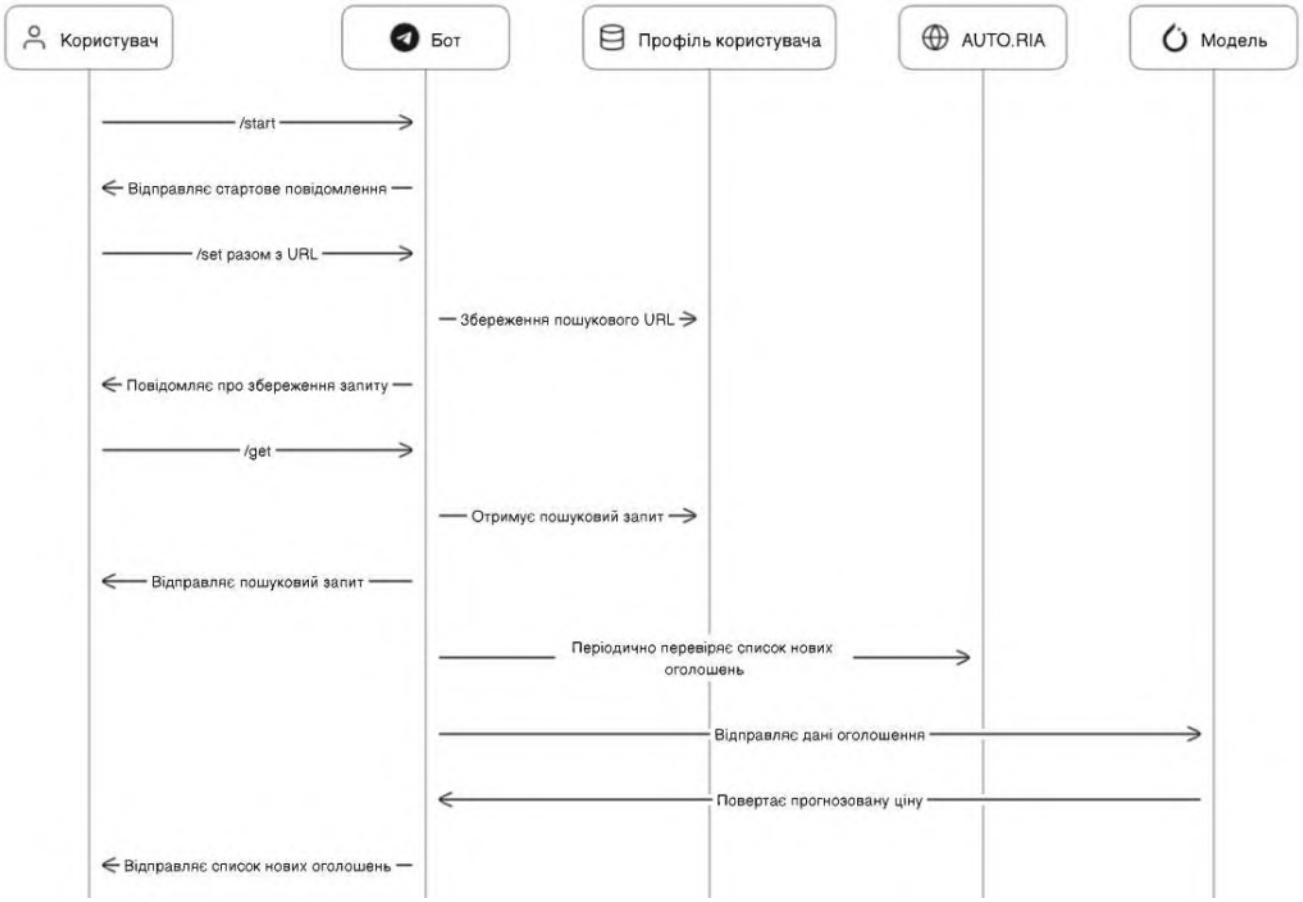


Рисунок 2.3 – Послідовність взаємодій між користувачем та модулями системи

5. Після збереження URL бот відправляє повідомлення користувачеві, що запит був збережений успішно.

6. Коли користувач вводить команду «/get», бот відправляє запит до профілю користувача в базі даних, щоб отримати збережений URL.

7. Бот відправляє отриманий пошуковий запит користувачу.

8. За цим URL бот періодично відправляє запити до веб-сайту AUTO.RIA, перевіряючи нові оголошення.

9. Отримані дані оголошення відправляються ботом на модель машинного навчання для прогнозування ціни.

10. Модель аналізує вхідні дані і повертає боту прогнозовану ціну.

11. Бот, в свою чергу, надсилає користувачу список нових оголошень із прогнозованою ціною.

Розроблений інтерфейс Telegram-бота для автоматизованого моніторингу оголошень вирізняється високою зручністю та інтуїтивною зрозумілістю для користувача.

Текстовий формат команд спрощує взаємодію з ботом, не вимагаючи спеціальних технічних знань або навичок. Це дозволяє користувачу легко та швидко виконувати необхідні дії, що значно підвищує ефективність використання сервісу.

Автоматизований моніторинг і відправка нових оголошень відповідно до пошукового запиту користувача значно спрощує процес пошуку авто, дозволяючи користувачу зосередитися на аналізі вже відібраних пропозицій.

Таким чином, інтерфейс даного бота демонструє високий рівень користувацької зручності, що значно спрощує і оптимізує процес пошуку автомобілів на онлайн-платформі.

2.5 Висновки до розділу

В рамках цього розділу було проведено проектування програмного бота для автоматизованого моніторингу оголошень онлайн-ринку вживаних авто. У підсумку, спроектований бот для моніторингу оголошень онлайн-ринку вживаних авто дозволяє користувачам ефективно та зручно слідкувати за актуальними пропозиціями, враховуючи їхні індивідуальні інтереси та вимоги. Реалізація даної технології дозволяє автоматизувати потоки даних, забезпечуючи регулярний збір актуальної інформації та відправку сповіщень користувачам.

Використовуючи об’ємний, але не вичерпний список критеріїв вибору автомобіля, користувачі зможуть здійснювати глибокий, деталізований аналіз потенційних варіантів, що найкраще відповідають їх індивідуальним потребам та вимогам.

Спроектований бот включає низку функціональних компонентів, які роблять його потужним та зручним інструментом для пошуку автомобілів. Він автоматизує процес моніторингу і відправки нових оголошень, спрощуючи

процес пошуку авто та дозволяючи користувачу зосередитися на аналізі вже відібраних пропозицій.

Спроектований інтерфейс даного бота демонструє високий рівень користувацької зручності, що значно спрощує і оптимізує процес пошуку автомобілів на онлайн-платформі. Незважаючи на велику кількість параметрів, які слід враховувати при виборі автомобіля, ця технологія допоможе обробляти великі обсяги даних, використовуючи високий рівень автоматизації, що дозволить користувачам зосередитися на виборі найкращого варіанту, враховуючи їх індивідуальні потреби та вимоги.

Тому, використання телеграм-бота для моніторингу та аналізу ринку вживаних авто є прекрасним прикладом того, як технології можуть бути використані для покращення процесу прийняття рішень, забезпечуючи користувачам можливість ефективно та зручно вибирати автомобіль, що найкраще відповідає їх потребам та вимогам.

Розділ 3 Програмна реалізація програмного боту з використанням обраного способу оброблення даних

3.1 Структура та функціональне призначення програмних складових програмного бота

Для виконання поставлених задач та вимог було розроблено програмний бот для автоматизованого моніторингу оголошень онлайн-ринку вживаних авто.

Коректна робота та ефективність програмного бота для автоматизованого моніторингу онлайн-ринку вживаних авто в значній мірі залежать від його внутрішньої структури та взаємодії між окремими компонентами. У цьому розділі детально розглянемо структуру програмного бота, а також розкриємо функціональне призначення його основних складових.

Розглянемо компоненти бота (рисунки 3.1-3.2). Кожний з цих класів відповідає за виконання певних завдань, що дозволяє боту працювати як єдиний цілісний механізм. Це дозволить нам краще розуміти, як бот взаємодіє з даними, і яким чином він виконує свої основні функції.

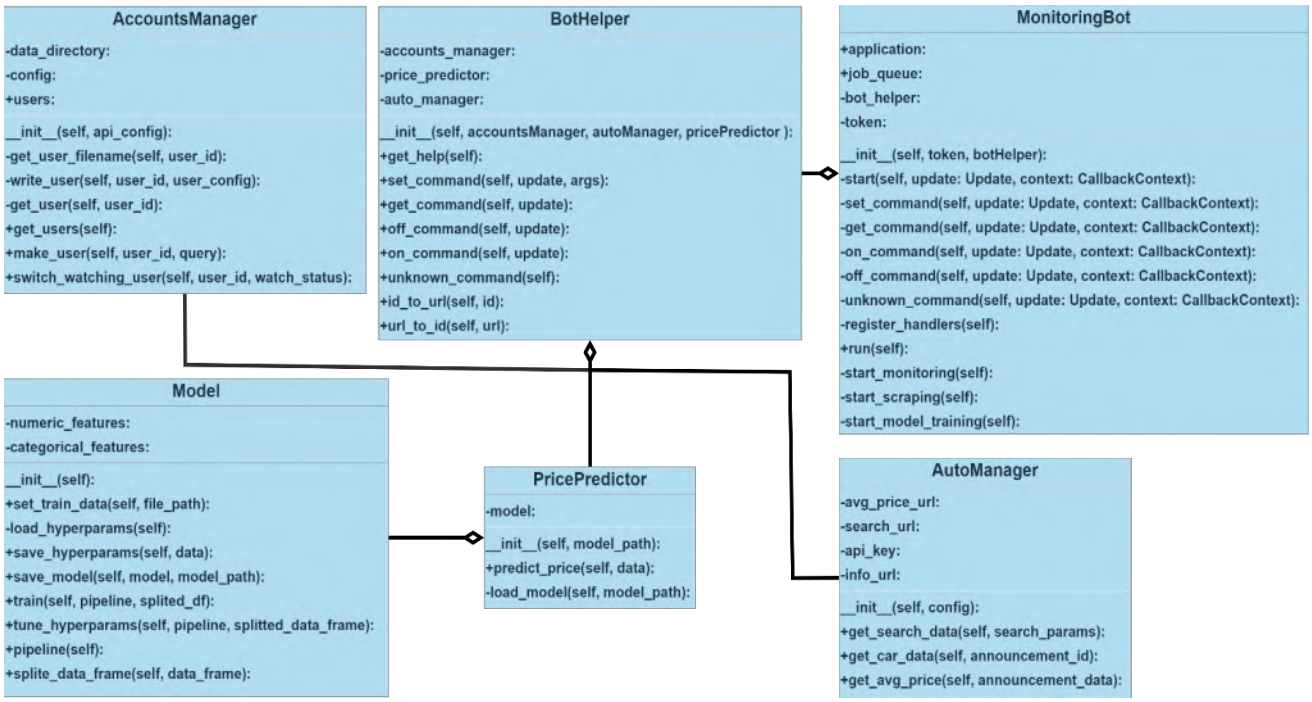


Рисунок 3.1 – Основна діаграма класів

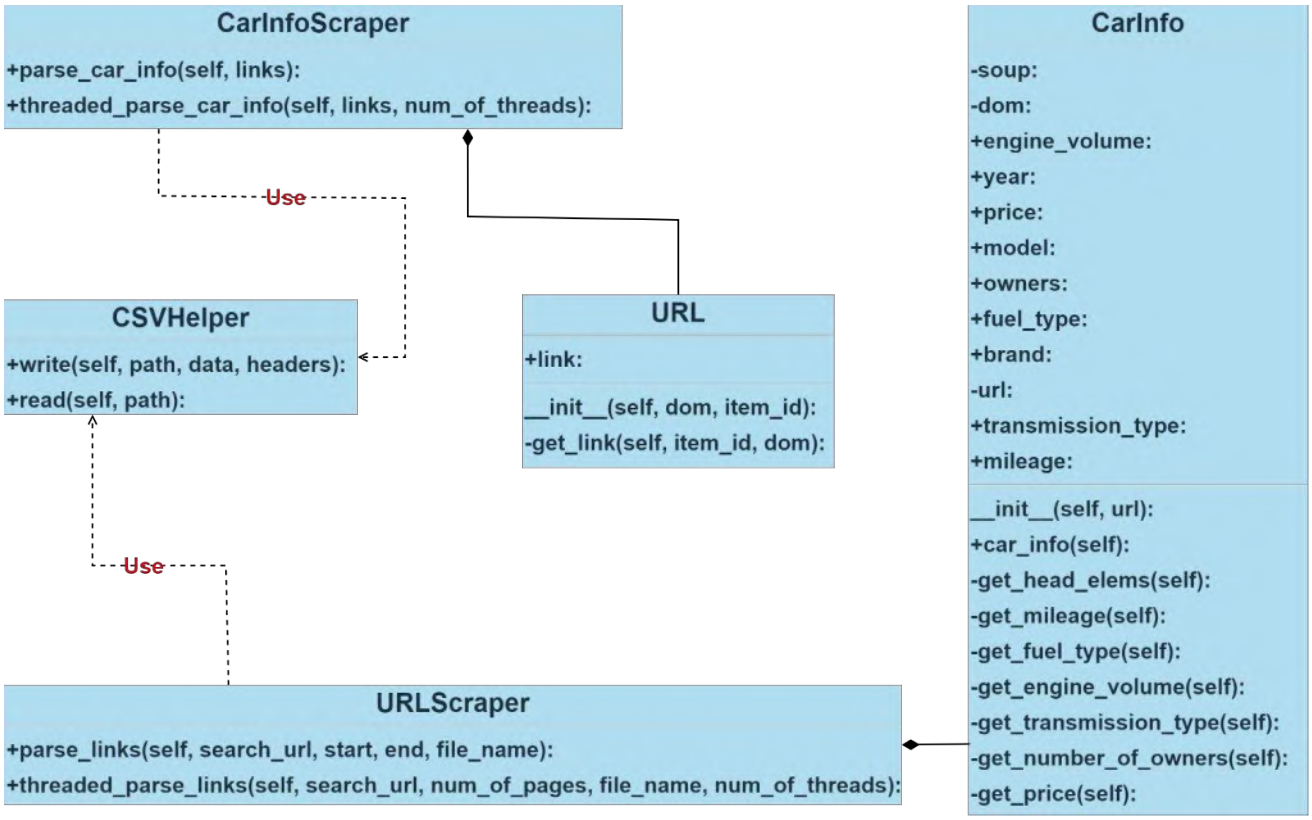


Рисунок 3.2 – Діаграма класів скрапера

Система містить наступні класи:

– CSVHelper – робота з CSV-файлами;

– BotHelper – надання основних допоміжних функцій;

– AccountsManager – управління даними користувачів;

– AutoManager – взаємодія з API AutoRIA;

– MonitoringBot – створення та управління телеграм-ботом;

– PricePredictor – прогнозування ціни автомобіля;

– Model – робота з моделю;

– URL – парсинг посилання;

– CarInfo – парсинг інформації про автомобіль;

– URLScraper – збір посилань на оголошення;

– CarInfoScraper – збір інформації про автомобілі з кількох веб-сторінок.

Розглянемо задачі, які вирішує кожен з класів.

Клас CSVHelper використовується для роботи з CSV-файлами. Він має два основних методи:

– Метод write – використовується для запису даних у CSV-файл. Цей метод приймає три параметри: path (шлях до файлу), data (дані, які потрібно записати) та headers (заголовки стовпців для даних).

– Метод read – цей метод використовується для читання даних з CSV-файлу. Метод приймає один параметр: path (шлях до файлу). Метод читає файл, вказаний в параметрі path, і повертає дані у вигляді об’єкта Pandas DataFrame.

Загалом, цей клас спрощує процес роботи з CSV-файлами, забезпечуючи зручний інтерфейс для читання та запису даних.

Клас Model використовується для підготовки, тренування і тюнінгу моделі. Він має наступні методи:

– Метод __init__ – це конструктор класу. Основна задача – ініціалізація класу з заданими категоріальними та числовими ознаками.

Метод set_train_data – цей метод читає CSV – файл за заданим шляхом і повертає об’єкт Pandas DataFrame.

– Методи load_hyperparams і save_hyperparams – ці методи відповідають за зчитування та запис гіперпараметрів моделі з/до JSON-файлу відповідно.

– Методи load_model і save_model – ці методи відповідають за зчитування та збереження обєкта моделі машинного навчання з/до файлу.

– Метод train – відповідає за тренування моделі на основі вхідних даних і гіперпараметрів, збережених у JSON-файлі.

– Метод tune_hyperparams – відповідає за виконання пошуку гіперпараметрів за допомогою GridSearchCV, зберігає найкращі параметри в JSON-файл.

– Метод pipeline – відповідає за створення конвеєра для попередньої обробки даних і тренування моделі.

– Метод splite_data_frame – відповідає за розбиття Pandas DataFrame на тренувальний та тестувальний набори даних і повертає їх.

Цей клас надає інтерфейс для підготовки даних, налаштування гіперпараметрів, тренування та збереження моделі машинного навчання, а також

для роботи з моделлю та її гіперпараметрами. Він є досить загальним та адаптивним, що дозволяє йому працювати з різними наборами даних і моделями.

Клас PricePredictor призначений для прогнозування ціни автомобіля на основі ряду характеристик, використовуючи попередньо навчену модель машинного навчання. Він має такі методи:

– Метод __init__ – це конструктор класу. Основна задача – ініціалізація класу. В процесі ініціалізації завантажується модель машинного навчання.

– Метод predict_price – цей метод приймає дані про автомобіль, обробляє ці дані та формує DataFrame з цими даними, після чого на основі цих даних робить прогноз за допомогою моделі та повертає цей прогноз.

– Метод load_model – цей метод відповідає за завантаження моделі машинного навчання.

Загалом, цей клас відповідає за використання попередньо навченої модель для прогнозування ціни автомобіля на основі вхідних даних.

Класи URL і URLScraper, призначені для веб-скрапінгу, процесу автоматичного збору інформації з Інтернету.

Клас URL призначений для парсингу посилань на основі HTML дерева (DOM) і ідентифікатора елемента.

Метод get_link витягує посилання з вказаного елемента за допомогою XPath.

Клас URLScraper використовується для збору посилань на оголошення з веб-сторінок. Він має два основних методи:

– Метод parse_links – цей метод використовується для збору посилань з певного діапазону веб-сторінок та збереження цих посилань в CSV-файл. Він проходить через кожну сторінку в заданому діапазоні, збирає HTML-код сторінки, перетворює його в DOM, а потім використовує клас `URL` для витягування посилань з кожного елемента на сторінці.

– Метод threaded_parse_links – цей метод використовується для паралельного збору посилань за допомогою багатопоточності. Він розбиває

заданий діапазон сторінок на піддіапазони, кожен з яких обробляється окремим потоком. Це може значно прискорити процес збору даних.

Загалом, ці класи роблять збір посилань з веб-сторінок більш структурованим та ефективним.

Класи CarInfo та CarInfoScraper, призначені для автоматичного збору інформації про автомобілі з Інтернету.

Клас CarInfo призначений для збору інформації про конкретний автомобіль з веб-сторінки. Він включає такі дані як марка, модель, рік випуску, пробіг, тип палива, об’єм двигуна, тип трансмісії, кількість власників та ціна. Цей клас використовує BeautifulSoup та lxml для парсингу HTML структури сторінки і витягування необхідної інформації.

Клас CarInfoScraper використовується для збору інформації про автомобілі з кількох веб-сторінок. Він має два основних методи:

– Метод parse_car_info – цей метод проходить через список URL-адрес, збирає інформацію про кожен автомобіль за допомогою класу CarInfo та зберігає ці дані в CSV-файл.

– Метод threaded_parse_car_info – цей метод використовується для паралельного збору інформації про автомобілі за допомогою багатопоточності. Він розбиває список URL-адрес на підсписки, кожен з яких обробляється окремим потоком. Це може значно прискорити процес збору даних.

Ці класи використовуються для автоматизованого збору детальної інформації про автомобілі з веб-сторінок, що може бути корисним, наприклад, для аналізу ринку автомобілів або побудови моделей прогнозування цін на автомобілі.

Клас AutoManager призначений для взаємодії з API AutoRIA. Цей клас містить такі методи:

– Метод __init__- конструктор класу. При створенні екземпляра цього класу, йому передається конфігурація, яка включає посилання на різні API-посилання (пошук, інформація про авто, середня ціна) та API-ключ для авторизації.

– Метод get_search_data – цей метод відповідає за отримання списку посилань. Приймає параметри пошуку та використовує API для отримання даних. Він формує URL, виконує запит GET та перетворює отриманий JSON-файл відповідь у Python-об’єкт.

– Метод get_car_data – цей метод відповідає за отримання інформації про автомобіль. Він приймає ідентифікатор оголошення і використовує API, щоб отримати деталі про конкретний автомобіль. Цей метод також формує URL, виконує запит GET та перетворює JSON-відповідь у Python-об’єкт.

– Метод get_avg_price – цей метод відповвідає за отримання середньої ціни на авто. Він приймає дані оголошення, формує URL і використовує API для отримання інформації про середню ціну автомобілів подібного типу. Він також робить деякі обчислення для визначення діапазону років та пробігу для пошуку середньої ціни.

Цей клас є основним інструментом для роботи з вказаним API. Він може бути використаний у будь-якому контексті, де потрібно отримувати та обробляти дані про автомобілі через цей API.

Клас AccountsManager призначений для управління обліковими записами користувачів. Цей клас має наступні методи:

– Метод __init__ – конструктор класу. При створенні екземпляра цього класу, йому передається конфігурація API. Він також створює каталог для зберігання даних користувачів, якщо він ще не існує.

– Метод write_user – метод, який відповідає за запис даних користувача. Він записує дані користувача у відповідний JSON-файл.

– Метод get_user – метод, який відповідає за зчитування даних користувача. Він витягує дані користувача з відповідного JSON-файлу.

– Метод get_users – метод, який відповідає за отримання даних про всіх користувачів. Він читає всі JSON-файли в каталозі користувачів та повертає словник з активними користувачами та їхніми даними.

– Метод make_user – метод, який відповідає за створення нового облікового запису користувача. Він створює новий обліковий запис користувача з заданим ідентифікатором користувача та запитом.

– Метод switch_watching_user – метод, який відповідає за зміну статусу спостереження користувача. Він дозволяє змінити статус спостереження для конкретного користувача.

Клас AccountsManager використовується для управління даними користувачів, зокрема для створення нових користувачів, запису та зчитування даних користувача та переключення статусу спостереження. Це корисний інструмент для будь-якої системи, яка потребує зберігання та управління даними користувачів.

Клас MonitoringBot використовується для створення та управління телеграм-ботом, який слідкує за автомобільним ринком. Ось основні методи цього класу:

– Метод __init__ – конструктор класу. При створенні об’єкту цього класу, йому передається токен та допоміжний об’єкт botHelper. За допомогою цих параметрів створюється застосунок бота.

– Методи обробки команд – це різні асинхронні методи (start, set_command, get_command, on_command, off_command, unknown_command), які використовуються для обробки відповідних команд від користувачів.

– Метод register_handlers – це метод, який відповідає за реєстрацію обробників. Він додає обробник для кожної команди, яка має бути оброблена ботом.

– Метод start_monitoring – цей метод відповідає за процес моніторингу онлайн-ринку вживаних авто. Він запускає фонову задачу, яка регулярно перевіряє ринок автомобілів і надсилає оновлення користувачам.

– Метод start_scraping запускає щоденну фонову задачу, яка використовує скрапери для збору даних з автомобільного ринку.

– Метод start_model_training запускає щотижневу фонову задачу, яка навчає модель на зібраних даних.

– Метод run використовується для початку роботи бота, включаючи опитування на наявність нових оновлень.

Клас MonitoringBot є комплексним інструментом, який використовує Telegram API, скрапінг веб-сайтів, обробку даних і машинне навчання для створення інформативного телеграм-бота для моніторингу автомобільного ринку.

Клас BotHelper використовується для надання основних допоміжних функцій та службових повідомлень для телеграм-бота. Ось основні методи цього класу:

– Метод __init__ – конструктор класу. При створенні об’єкту BotHelper йому передаються об’єкти AcountsManager, AutoManager і PricePredictor, які використовуються для виконання різних функцій бота.

– Метод get_help повертає текст довідки, який описує, як користувач може використовувати бота.

– Методи set_command, get_command, off_command, on_command та unknown_command використовуються для обробки відповідних команд від користувачів. Ці команди дозволяють користувачам налаштовувати та керувати моніторингом автомобільного ринку.

– Методи id_to_url та url_to_id використовуються для конвертації між ідентифікаторами автомобілів та відповідними URL-адресами на веб-сайті AUTO.RIA.

Клас BotHelper допомагає в організації взаємодії між користувачем та телеграм-ботом, обробляє дії користувача та формує відповіді бота.

Було детально проаналізовано два основних класи – MonitoringBot та BotHelper, які створюють основу для функціонування телеграм-бота, що моніторить ринок вживаних автомобілів, використовуючи методи ІАД.

Клас MonitoringBot має в своєму складі основні методи, потрібні для запуску та роботи бота, включаючи методи, які обробляють вхідні команди від користувачів, реєструють обробник команд, розпочинають моніторинг та скрапінг даних, а також здійснюють навчання моделі для прогнозування цін. Цей клас ефективно поєднує всі складові бота та управляє їх виконанням.

Клас BotHelper використовується для надання основних допоміжних функцій та службових повідомлень для телеграм-бота, а також обробки команд від користувачів та формування відповідей бота. Цей клас спрощує взаємодію між користувачем та ботом та дозволяє користувачам керувати моніторингом автомобільного ринку.

Обидва класи разом відповідають за основний функціонал, який забезпечує автоматизований моніторинг онлайн-ринку вживаних авто з використанням методів ІАД. Використання такого бота значно полегшує процес відстеження нових пропозицій на ринку та дозволяє користувачам отримувати найактуальнішу інформацію в зручній формі.

3.2 Особливості реалізації програмних складових бота

У реалізації програмного бота використовуються два ключові класи: MonitoringBot та BotHelper. Вони поєднують у собі ряд технологій, що включають веб-скрапінг, використання API Telegram, роботу з даними, а також методи машинного навчання для прогнозування цін. Перейдемо до детального розгляду особливостей реалізації цих класів.

Реалізація класу MonitoringBot базується на бібліотеці python-telegram-bot, що забезпечує взаємодію з API Telegram (рисунок 3.3) [22]. Однією з ключових особливостей даного класу є використання внутрішнього класу бібліотеки python-telegram-bot – JobQueue для запуску процесу періодичного виконання скрапінгу та надсилання оновлень користувачам. Це виконується у фоновому режимі, що дозволяє боту ефективно відповідати на команди користувачів без втрати продуктивності.

Також важливою особливістю є інтеграція з моделлю машинного навчання для прогнозування цін на автомобілі (рисунок 3.4). Це забезпечує значну додаткову цінність для користувачів, оскільки бот не просто надає інформацію про нові пропозиції, але і допомагає розуміти, чи є запропонована ціна доречною.

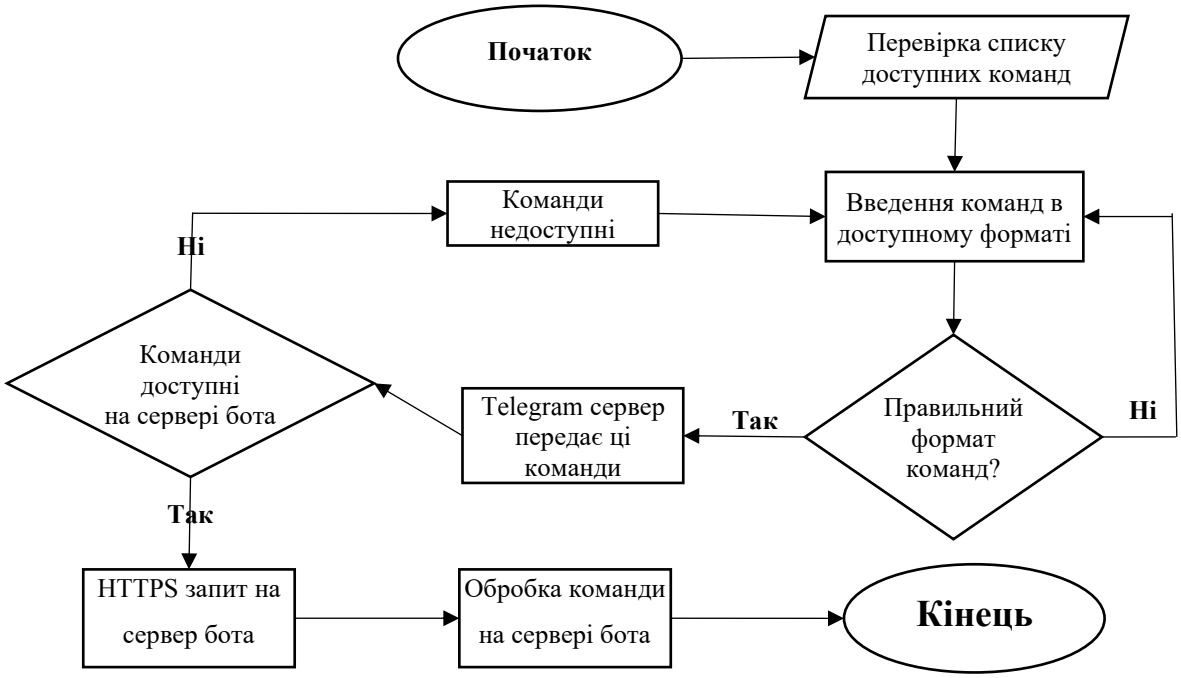


Рисунок 3.3 – Схема взаємодії з API Telegram

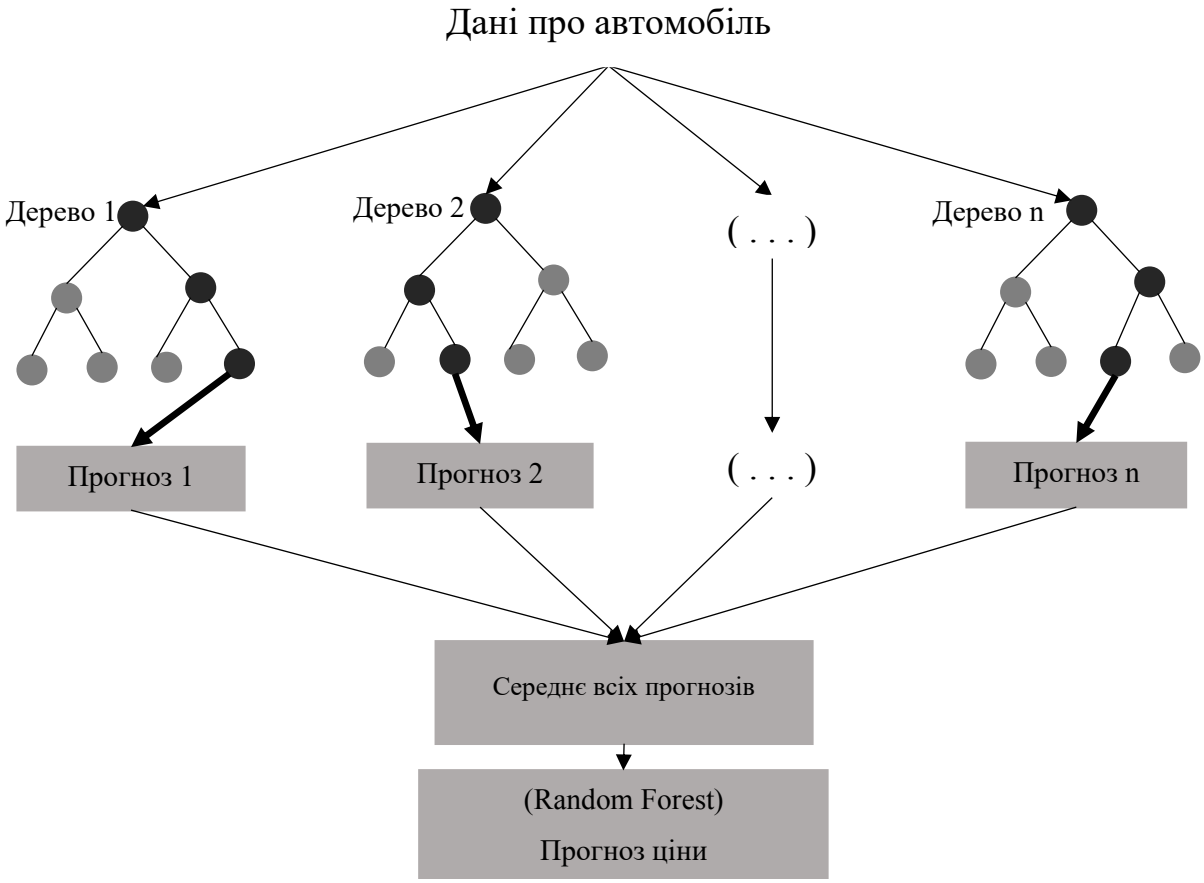


Рисунок 3.4 – Схема роботи моделі машинного навчання для прогнозування ціни на автомобіль

Клас BotHelper відповідає за обробку вхідних команд від користувачів і генерацію відповідей. Він інтегрований із класом AccountsManager, що забезпечує роботу з даними користувачів, і класом AutoManager, що керує скрапінгом даних. Крім того, клас BotHelper використовує функції для перетворення URL на ідентифікатор автомобіля і навпаки, що допомагає керувати даними, отриманими з сайту AUTO.RIA.com.

Він також включає розгорнуту систему помилок та повідомлень для користувачів, що дозволяє розробнику моніторити та управляти потенційними проблемами, що виникають під час використання бота.

Однією з основних особливостей реалізації бота є його модульність. Дизайн програмного забезпечення був ретельно продуманий, щоб кожна частина системи могла функціонувати незалежно та була легко замінна. Це полегшує внесення змін та розширення функціоналу в майбутньому.

Ще однією важливою особливістю є акцент на безпеку та конфіденційність. Бот не збирає більше інформації, ніж необхідно для його роботи.

Також включено важливі принципи управління помилками, що дозволяють швидко відслідковувати та вирішувати проблеми, забезпечуючи стабільну роботу бота.

У загальному, бот поєднує в собі актуальні технології для надання високоякісного сервісу моніторингу онлайн-ринку вживаних автомобілів, роблячи його простим у використанні, але потужним інструментом.

3.3 Експериментальне тестування програмного бота

Важливим етапом розробки програмного бота для автоматизованого моніторингу онлайн-ринку вживаних авто є його експериментальне тестування. Метою цього є перевірка працездатності та ефективності бота, а також визначення його здатності виконувати заявлені функції.

Експериментальне тестування допоможе нам зрозуміти, наскільки добре система впорається зі збором та аналізом даних, а також наскільки вона зручна для користувачів. Проведемо низку тестів, які включатимуть в себе перевірку роботи основних функцій бота. Результати цього тестування допоможуть нам виявити можливі недоліки та визначити напрямки для подальшого удосконалення системи.

Для тестування бота можна розглянути наступні тест-кейси.

Перший тест-кейс перевіряє коректність виконання команди «/set» з валідним URL, який було скопійовано користувачем на вкладці «Розширений пошук» сайту AUTO.RIA (таблиця 3.1).

Таблиця 3.1 – Тест-кейс AT0001

Тест-кейс ID: AT0001	**Пріоритет:** 1
Назва: Перевірка коректності виконання команди «/set» з валідним URL **Вхідні дані:** Команда = «/set», URL = «Валідне посилання»	
Кроки	**Очікуваний результат**
1. Відкрити бота 2. Натиснути кнопку «Розпочати» 3. Відкрити сайт AUTO.RIA 4. Перейти на сторінку пошуку 5. Обрати фільтри 6. Копіювати посилання 7. Повернутися до бота 8. Викликати команду «/set» разом з скопійованим посиланням	Бот підтверджує отримання URL та запам’ятовує пошуковий запит користувача
Результат виконання тест-кейсу: Пройдено успішно	

Необхідно відкрити бота, далі натиснути на кнопку «Розпочати», яка знаходиться на вкладці бота. Після необхідно відкрити сайт AUTO.RIA та вкладку «Розширений пошук», обрати усі необхідні параметри та скопіювати URL посилання, після чого повернутися до бота та викликати команду «/set» разом з цим посиланням (рисунок 3.5).

Рисунок 3.5 – Виконання команди «/set» з валідним URL

Другий тест-кейс перевіряє коректність виконання команди «/get» після встановлення URL для пошуку оголошень (таблиця 3.2).

Таблиця 3.2 – Тест-кейс AT0002

Тест-кейс ID: AT0002	**Пріоритет:** 1
Назва: Перевірка коректності виконання команди «/get» після встановлення URL **Вхідні дані:** Команда = «/get»	
Кроки	**Очікуваний результат**
1. Відкрити бота 2. Натиснути кнопку «Розпочати» 3. Відкрити сайт AUTO.RIA 4. Перейти на сторінку пошуку 5. Обрати фільтри 6. Копіювати посилання 7. Повернутися до бота 8. Викликати команду «/set» разом з скопійованим посиланням 9. Викликати команду «/get»	Бот надсилає поточний пошуковий запит, встановлений користувачем
Результат виконання тест-кейсу: Пройдено успішно	

Необхідно відкрити бота, далі натиснути на кнопку «Розпочати», яка знаходиться на вкладці бота. Після необхідно відкрити сайт AUTO.RIA та вкладку «Розширений пошук» , обрати усі необхідні параметри та скопіювати URL посилання, після чого повернутися до бота та викликати команду «/set» разом з цим посиланням, зачекати відповіді від бота про успішне виконання та викликати після цього команду «/get» (рисунок 3.6).

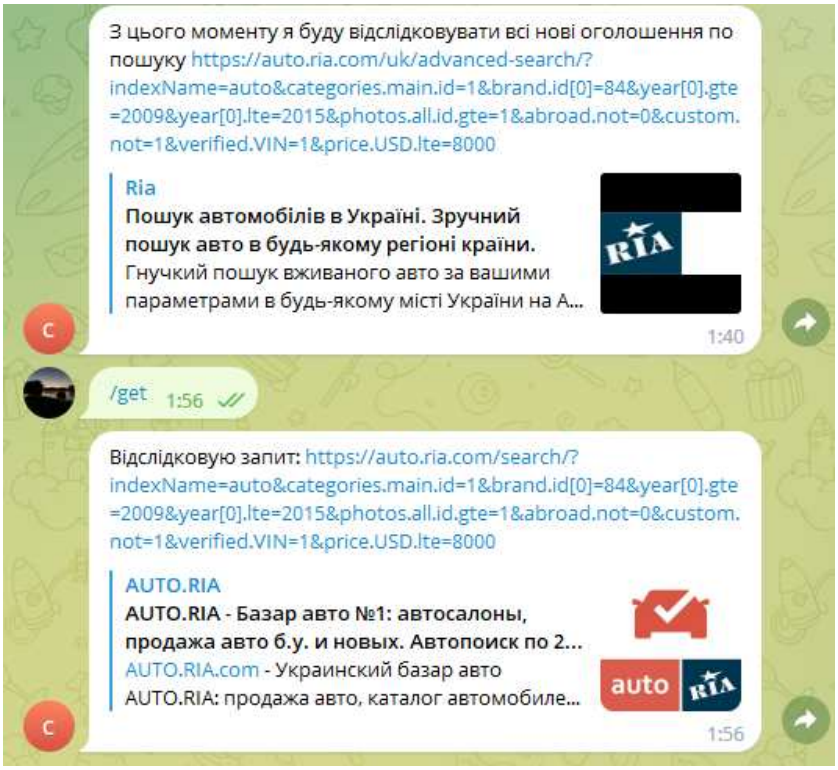


Рисунок 3.6 – Виконання команди «/get» після встановлення URL для пошуку оголошень

Третій тест-кейс перевіряє коректність виконання команди «/on» (таблиця 3.3). Необхідно відкрити бота, далі натиснути на кнопку «Розпочати», яка знаходиться на вкладці бота. Після необхідно відкрити сайт AUTO.RIA та вкладку «Розширений пошук», обрати усі необхідні параметри та скопіювати URL посилання, після чого повернутися до бота та викликати команду «/set» разом з цим посиланням, зачекати відповіді від бота про успішне виконання та викликати після цього команду «/on» (рисунок 3.7).

Таблиця 3.3 – Тест-кейс AT0003

Тест-кейс ID: AT0003	**Пріоритет:** 1
Назва: Перевірка коректності виконання команди «/on» **Вхідні дані:** Команда = «/on»	
Кроки	**Очікуваний результат**
1. Відкрити бота 2. Натиснути кнопку «Розпочати» 3. Відкрити сайт AUTO.RIA 4. Перейти на сторінку пошуку 5. Обрати фільтри 6. Копіювати посилання 7. Повернутися до бота 8.Викликати команду «/set» разом з скопійованим посиланням 9. Викликати команду «/on»	Бот надсилає повідомлення, у якому підтверджує активування режиму відслідковування оголошень
Результат виконання тест-кейсу: Пройдено успішно	

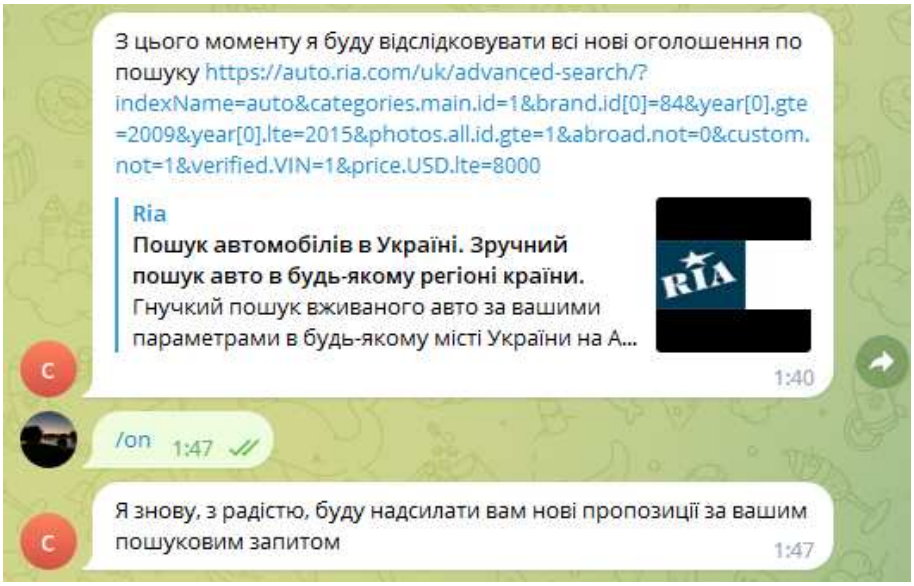


Рисунок 3.7 – Виконання команди «/on» після встановлення URL для пошуку оголошень

Четвертий тест-кейс перевіряє коректність виконання команди «/off» (таблиця 3.4). Необхідно відкрити бота, далі натиснути на кнопку «Розпочати», яка знаходиться на вкладці бота. Після необхідно відкрити сайт AUTO.RIA та

вкладку «Розширений пошук» , обрати усі необхідні параметри та скопіювати URL посилання, після чого повернутися до бота та викликати команду «/set» разом з цим посиланням, зачекати відповіді від бота про успішне виконання та викликати після цього команду «/off» (рисунок 3.8).

Таблиця 3.4 – Тест-кейс AT0004

Тест-кейс ID: AT0004	**Пріоритет:** 1
Назва: Перевірка коректності виконання команди «/off» **Вхідні дані:** Команда = «/off»	
Кроки	**Очікуваний результат**
1. Відкрити бота 2. Натиснути кнопку «Розпочати» 3. Відкрити сайт AUTO.RIA 4. Перейти на сторінку пошуку 5. Обрати фільтри 6. Копіювати посилання 7. Повернутися до бота 8. Викликати команду «/set» разом з скопійованим посиланням 9. Викликати команду «/off»	Бот надсилає повідомлення, у якому підтверджує вимкнення режиму відслідковування оголошень
Результат виконання тест-кейсу: Пройдено успішно	

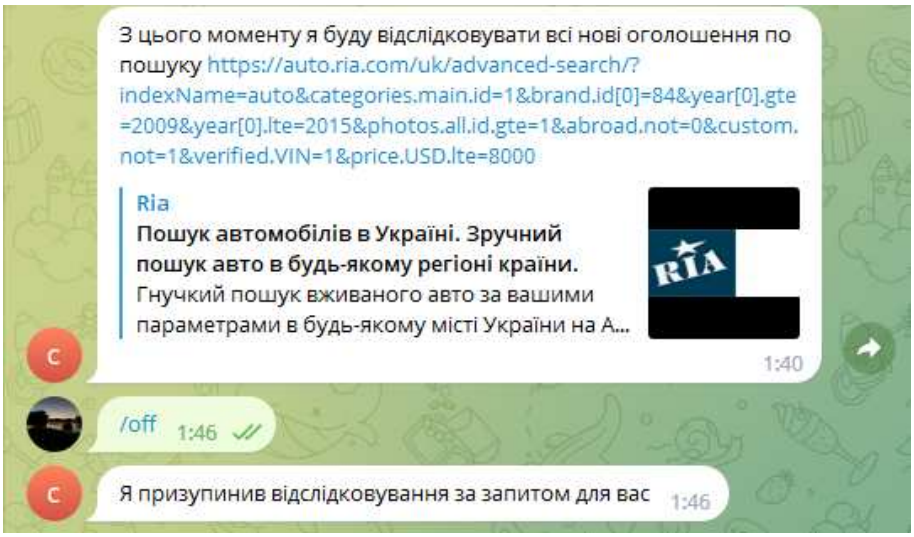


Рисунок 3.8 – Виконання команди «/off» після встановлення URL для пошуку оголошень

П’ятий тест-кейс перевіряє коректність виконання відправки нових оголошень(таблиця 3.5). Необхідно відкрити бота, далі натиснути на кнопку «Розпочати», яка знаходиться на вкладці бота. Після необхідно відкрити сайт AUTO.RIA та вкладку «Розширений пошук» , обрати усі необхідні параметри та скопіювати URL посилання, після чого повернутися до бота та викликати команду «/set» разом з цим посиланням, зачекати відповіді від бота про успішне виконання та зачекати відправки перших оголошень за встановленими параметрами.

Таблиця 3.5 – Тест-кейс AT0005

Тест-кейс ID: AT0005	**Пріоритет:** 1
Назва: Перевірка коректності виконання відправки нових оголошень **Вхідні дані:** Команда = «/set», URL = «Валідне посилання»	
Кроки	**Очікуваний результат**
1. Відкрити бота 2. Натиснути кнопку «Розпочати» 3. Відкрити сайт AUTO.RIA 4. Перейти на сторінку пошуку 5. Обрати фільтри 6. Копіювати посилання 7. Повернутися до бота 8. Викликати команду «/set» разом з скопійованим посиланням 9. Зачекати першої відправки оголошень за вказаним запитом	Бот надсилає нові оголошення користувачу
Результат виконання тест-кейсу: Пройдено успішно	

Статистичне тестування є важливою частиною процесу валідації та верифікації програмного бота. Це дозволяє нам оцінити його функціональність, точність і надійність в реальних умовах. У рамках даного тестування були використані ключову метрику – кількість помилок при відправці оголошень (рисунок 3.9).

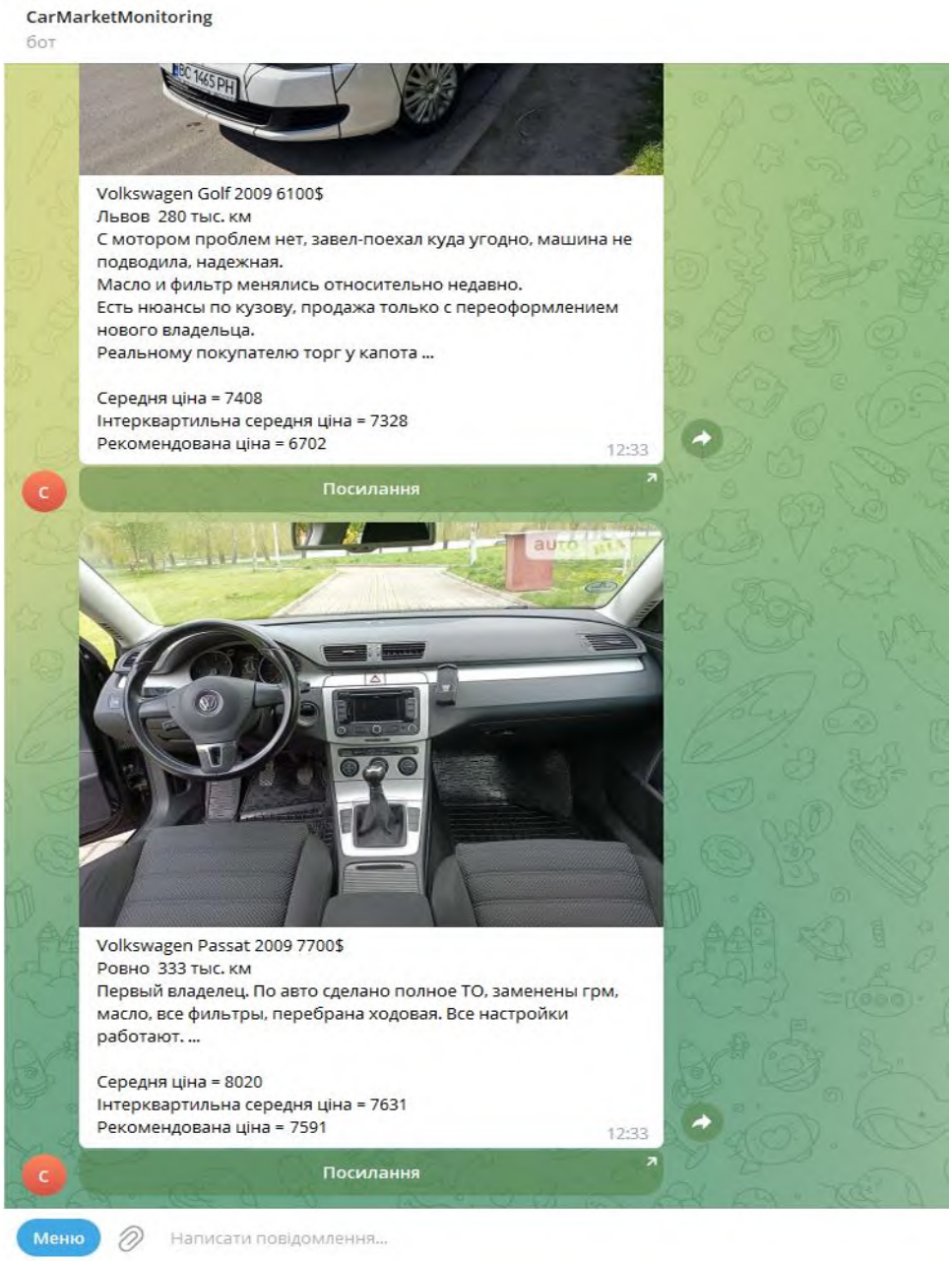


Рисунок 3.9 – Відправлення оголошень за встановленими параметрами

Метрика кількості помилок при відправці оголошень відслідковує кількість випадків, коли бот не вдалося відправити нове оголошення користувачу. Це може виникнути внаслідок різних проблем, таких як помилки відповіді сервера, неправильне формування запиту або невідповідність оголошення заданим критеріям пошуку.

Протягом тестового періоду було зафіксовано 10 помилок відправки з 500 спроб, що становить 2% від загальної кількості. Це свідчить про високий рівень надійності та стабільності бота.

Протягом експерименту, бот працював протягом 8 годин та надіслав користувачеві 154 оголошень, які відповідали його критеріям пошуку. Використовуючи формулу (2.2), можемо обчислити ефективність бота:

$$E = \frac{154}{8} = 19.25.$$

Тобто, ефективність бота становить 19.25 оголошень на годину. Цей результат підкреслює високу ефективність бота у пошуку та відправці оголошень, що відповідають критеріям пошуку користувача, що забезпечує важливе підвищення ефективності процесу пошуку автомобілів.

На основі отриманих результатів статистичного тестування можна зробити висновок, що програмний бот демонструє високий рівень надійності, точності та ефективності. Відповідно то проведених експерементів, бот відправляє оголошення користувачам з помилкою всього у 2% випадків та його ефективність становить 19.25 оголошень в годину відправлених для одного користувача. Ці результати показують, що бот ефективно виконує свої основні функції та може бути корисним інструментом для користувачів при пошуку автомобілів на AUTO.RIA.com.

В рамках експериментального тестування програмного бота для моніторингу оголошень ринку вживаних авто в Україні на платформі AUTO.RIA було проведено ряд випробувань, які демонструють ефективність його роботи.

Бот успішно імплементує всі визначені функції, включаючи здатність зберігати та відслідковувати пошукові запити користувачів, надсилати оновлення за цими запитами, а також надавати користувачам зручні механізми керування налаштуваннями бота.

Було протестовано, що бот стабільно працює і відправляє оновлення оголошень кожну годину, забезпечуючи користувачам актуальну інформацію про оголошення, що відповідають їхнім критеріям пошуку. Бот також успішно реагує на команди користувачів, що дозволяє їм легко керувати налаштуваннями процесу моніторингу.

Тестування показало, що програмний бот є надійним та ефективним інструментом для автоматизації моніторингу оголошень ринку вживаних авто та значно полегшує пошук потрібних оголошень для користувачів. Результати експериментального тестування підтверджують успішність виконання поставленої мети та завдань дипломної роботи, що було спрямоване на підвищення ефективності процесу моніторингу оголошень онлайн-ринку вживаних авто України.

3.4 Вимоги до розгортання програмного бота та інструкція користувача

3.4.1 Вимоги до розгортання програмного бота

Розгортання бота для моніторингу онлайн-ринку вживаних автомобілів вимагає врахування ряду специфікацій та залежностей. Дотримання цих вимог забезпечить стабільну та ефективну роботу додатку.

Додаток розроблено на Python, тому знадобиться середовище, сумісне з Python 3.7 або новішою версією. Python – це мова програмування, яка підтримує різні операційні системи, включаючи Windows, Linux та MacOS. Проте, рекомендовано використовувати Linux для розгортання через його відмінну підтримку скриптів та управління процесами.

Python – мова програмування з високим рівнем абстракції, що дозволяє зосередитися на бізнес-логіці проєкту, а не на низькорівневих деталях. Також Python підтримує різні стилі програмування (об’єктно-орієнтоване, функціональне, процедурне), що дозволяє використовувати найкращі практики для різних задач. Python має велику кількість бібліотек для різноманітних завдань, включаючи роботу з даними, машинне навчання, веб-скрепінг та роботу з API. Переваги:

– Читабельність – Python відомий своєю чистою та лаконічною синтаксичною структурою, яка сприяє легкому читанню і розумінню коду.

– Python має об`ємну стандартну бібліотеку, яка покриває багато загальних задач програмування.

– Спільнота – Python має одну з найбільших програмістських спільнот, що означає багатий вибір бібліотек та модулів для різноманітних завдань, а також багатий ресурс для навчання та підтримки.

Таким чином, дані інструменти, разом з Python, складають потужний набір інструментів для реалізації заданого проєкту, вони дозволять створити ефективний, гнучкий і водночас потужний телеграм-бот для моніторингу та аналізу ринку вживаних авто.

Бот використовує ряд сторонніх бібліотек Python, які слід встановити. На щастя, Python має прекрасну систему керування пакетами – pip, яка дозволяє легко встановлювати необхідні бібліотеки. Серед важливих залежностей: python-telegram-bot, pandas, scikit-learn, BeautifulSoup та requests.

python-telegram-bot – ця бібліотека надає зручний інтерфейс для роботи з Telegram Bot API. Вона містить всі необхідні інструменти для створення, управління та взаємодії з ботами в Telegram [23]. Переваги:

– Повна підтримка Telegram Bot API – бібліотека підтримує всі типи нововведень, які підтримуються Telegram Bot API.

– Зручність – Python-telegram-bot має об’єктно-орієнтовану архітектуру, зручний для розуміння інтерфейс і дозволяє використовувати Pythonic ідіоми.

BeautifulSoup [24] і requests [25] – ці бібліотеки дозволяють з легкістю виконувати веб-скрепінг. BeautifulSoup надає інструменти для парсингу HTML і XML документів, а requests дозволяє виконувати HTTP запити. З їх допомогою можна зібрати дані з веб-сторінок онлайн-ринку вживаних авто. Переваги:

– Парсинг HTML/XML – BeautifulSoup розуміє та обробляє документи, навіть якщо вони мають неправильну структуру.

– Виконання HTTP запитів – з допомогою requests можна легко виконувати HTTP запити, отримувати і відправляти дані.

Pandas – це бібліотека для обробки та аналізу даних, яка надає широкий спектр інструментів, зокрема структури даних, такі як DataFrame, що дозволяють виконувати швидкі операції з даними [26]. Переваги:

– Обробка даних – Pandas забезпечує засоби для обробки та маніпулювання структурованими даними.

– Швидкість – бібліотека оптимізована для високої продуктивності, зокрема для великих наборів даних.

Scikit-learn – це одна з найпопулярніших бібліотек для машинного навчання в Python. Вона надає широкий вибір алгоритмів навчання, які можна використовувати в реальних умовах [27]. Ось деякі з основних переваг Scikit-learn:

– Консистентність API. Всі моделі в Scikit-learn використовують уніфікований інтерфейс, який робить бібліотеку дуже зручною для використання. Основні методи, що доступні для моделей, включають fit (для навчання моделі), predict (для здійснення прогнозів) та score (для оцінки якості моделі).

– Обширні можливості. Scikit-learn надає інструменти для багатьох етапів роботи з даними: передобробка даних, зменшення розмірності, моделювання, валідація моделі, вибір параметрів моделі.

– Інтеграція з іншими бібліотеками Python. Scikit-learn добре інтегрується з багатьма іншими бібліотеками Python, як-от NumPy та Pandas. Це робить Scikit-learn особливо зручним для роботи з даними.

– Документація та спільнота. Scikit-learn має відмінну документацію з багатьма прикладами і порадами щодо використання бібліотеки. Широка спільнота користувачів також означає, що допомога легко доступна у випадку проблем або питань.

– Ефективність. Більшість алгоритмів в Scikit-learn оптимізовані і написані на C/C++, що робить їх ефективними для роботи з великими наборами даних.

Вибір Python та вказаних бібліотек для розробки телеграм-бота для моніторингу онлайн-ринку вживаних авто є досить обґрунтованим. Python є універсальною, гнучкою мовою програмування з широким спектром

можливостей та великою спільнотою. Це робить його відмінним вибором для розробки високорівневих застосунків. Бібліотека python-telegram-bot спрощує створення та управління ботами в Telegram, надаючи зручний інтерфейс для роботи з Telegram Bot API. BeautifulSoup і requests надають засоби для веб-скрепінгу, що дозволяє зібрати дані з веб-сторінок для подальшого аналізу. Pandas є потужною бібліотекою для обробки даних, яка надає швидкі і зручні структури даних та інструменти аналізу. Scikit-learn, разом з моделлю RandomForestRegressor, надає ефективні та гнучкі інструменти для машинного навчання, що дозволяє розробити точну модель для прогнозування цін на автомобілі.

Можливе використання віртуального середовища (наприклад, venv або Conda) для ізоляції залежностей бота від глобального середовища Python. Це також спрощує процес розгортання, оскільки лише потрібно розгорнути віртуальне середовище з відповідним файлом вимог.

Оскільки бот використовує Telegram API для взаємодії з користувачами, він повинен мати стабільне з’єднання з Інтернетом.

Сервер повинен мати достатні ресурси для обробки запитів від користувачів та виконання обчислень. Конкретні вимоги можуть відрізнятися в залежності від кількості користувачів та обсягу даних для обробки.

Було детально описано вимоги до розготання програмного бота. Після забезпечення вищезгаданих вимог, можливо розгорнути бота. Також можуть знадобитися також додаткові кроки, наприклад налаштування процесів для автоматичного запуску бота після перезавантаження сервера.

3.4.2 Інструкція користувача

Бот є експериментальним та створений для зручності користувачів, щоб вони могли отримувати оновлення оголошень відповідно до своїх індивідуальних пошукових запитів. Інструкція розроблена для допомоги користувачам у

використанні програмного бота, створеного для автоматизованого моніторингу оголошень ринку вживаних авто в Україні на платформі AUTO.RIA.com.

Для початку, необхідно відкрити Telegram та ввести в пошукову строку назву бота – «CarMarketMonitoringBot». Для того щоб відкрити його, в отриманих результатах пошуку потрібно знайти бота та натиснути на нього.

Для початку роботи з ботом потрібно натиснути кнопку «Розпочати», після чого автоматично відправиться повідомлення з командою «/start» , у відповідь на цю команду бот відправить повідомлення з короткою інформацією про себе, списком команд та їх описом (рисунок 3.10).

Першим кроком у роботі з ботом буде налаштування параметрів пошуку. Щоб задати пошуковий запит для моніторингу, використовується команда /set із посиланням на сторінку розширеного пошуку авто сайту AUTO.RIA з обраними параметрами параметрами. Щоб отримати це посилання, потрібно перейти на сайт AUTO.RIA та натиснути кнопку «Розширений пошук» (рисунок 3.11).

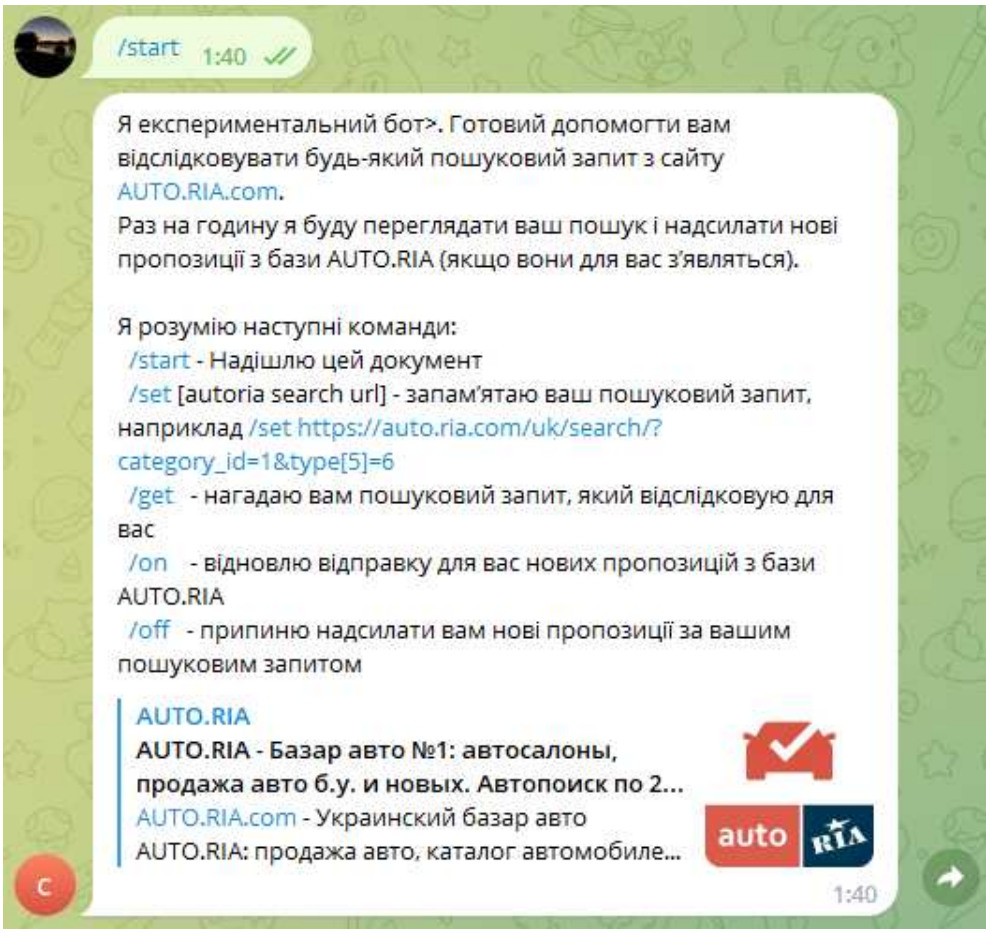


Рисунок 3.10 – Початок роботи з ботом

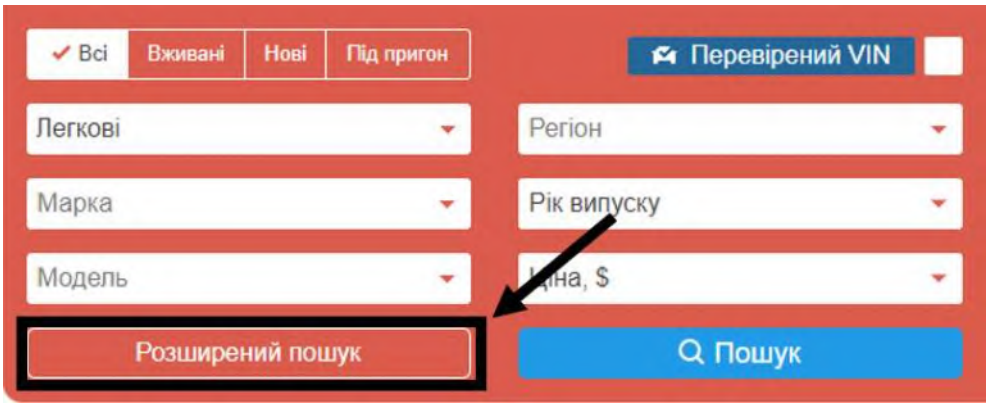


Рисунок 3.11 – Кнопка «Розширений пошук»

Після чого потрібно вибрати усі необхідні параметри та скопіювати посилання (рисунок 3.12).

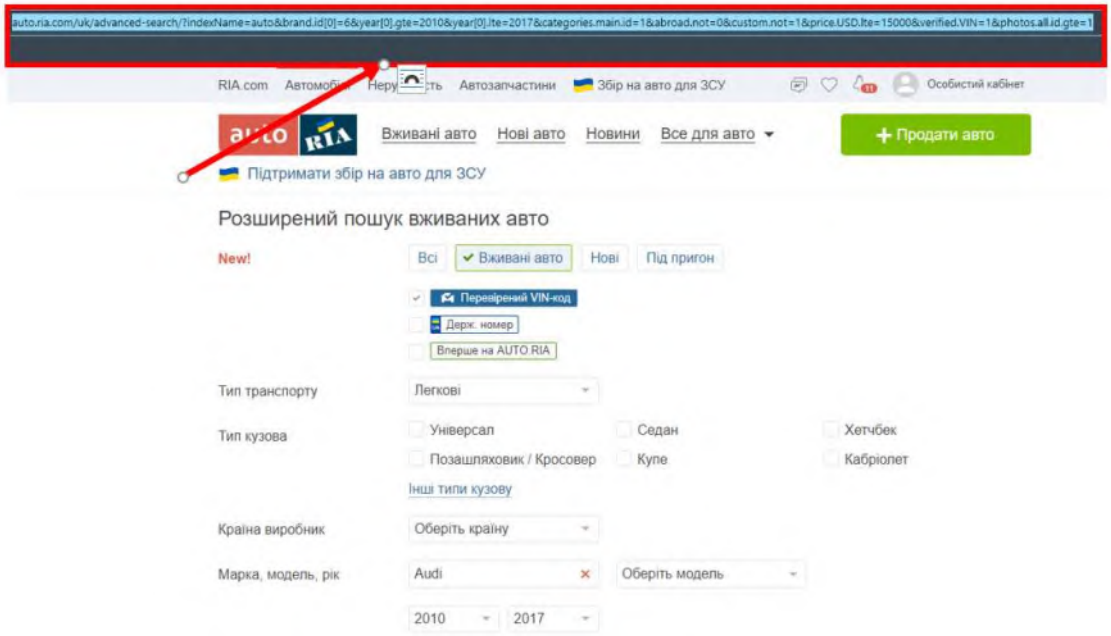


Рисунок 3.12 – URL посилання з параметрами для пошуку

Далі потрібно вписати скопійоване посилання одразу після команди «/set» та відправити це повідомлення (рисунок 3.13).

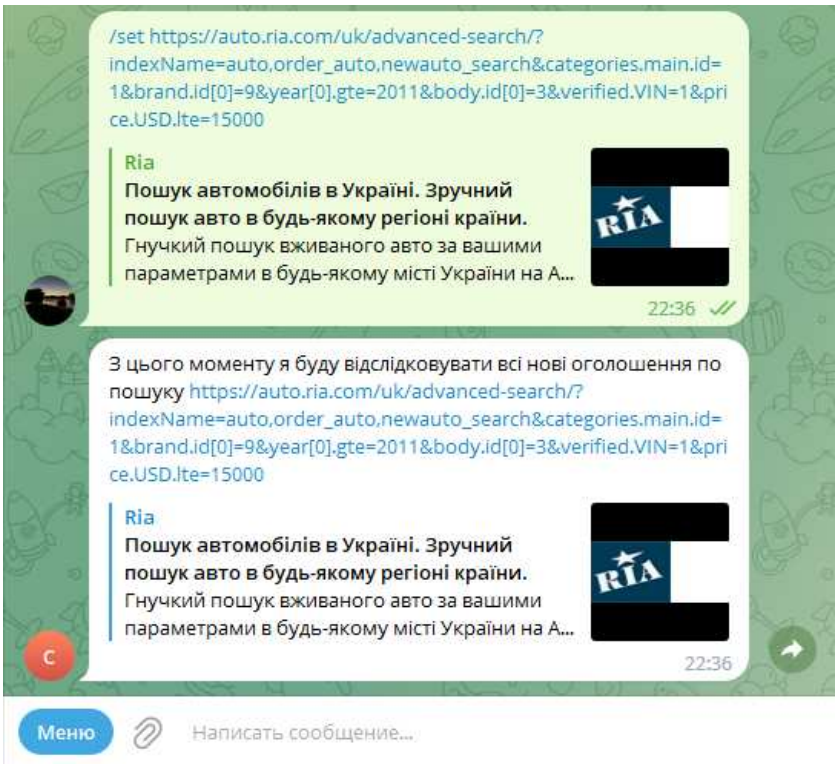


Рисунок 3.13 – Приклад виконання команди «/set»

Бот запам’ятає наданий пошуковий запит і буде відслідковувати оновлення за цим запитом на платформі AUTO.RIA.

Якщо потрібно переглянути, який пошуковий запит в даний час відслідковує бот, використайте команду «/get». Бот наддасть посилання на поточний пошуковий запит, який він відстежує (рисунок 3.14).

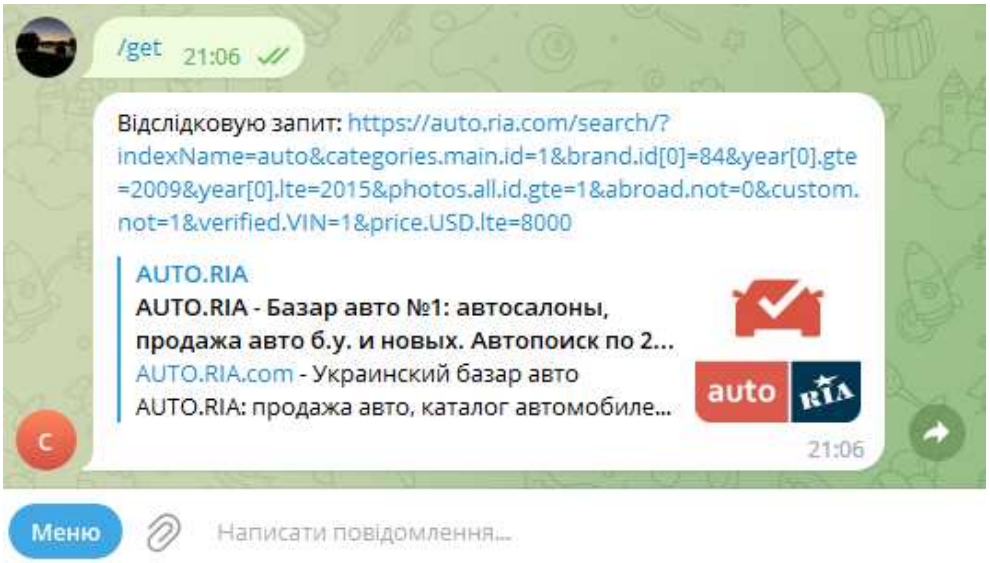


Рисунок 3.14 – Приклад виконання команди «/get»

Якщо потрібно припинити отримання повідомлень про нові оголошення від бота, використовуйте команду «/off». Бот припинить надсилати нові пропозиції згідно з пошуковим запитом (рисунок 3.15).

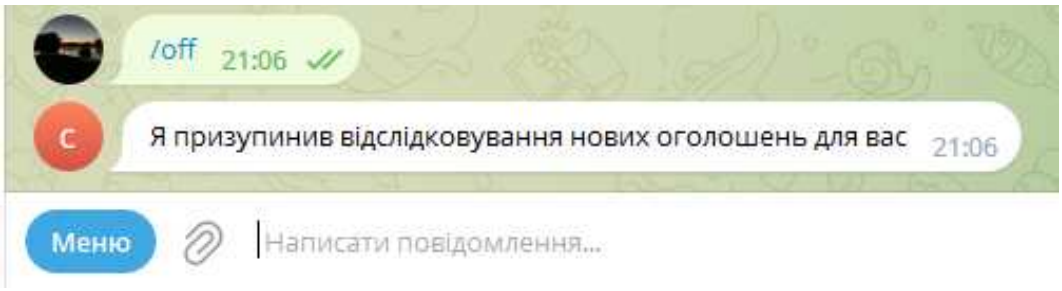


Рисунок 3.15 – Приклад виконання команди «/off»

Якщо бот був вимкнений або відправка оголошень була призупинена, можна використати команду «/on», щоб відновити відправку нових пропозицій з сайту AUTO.RIA (рисунок 3.16).

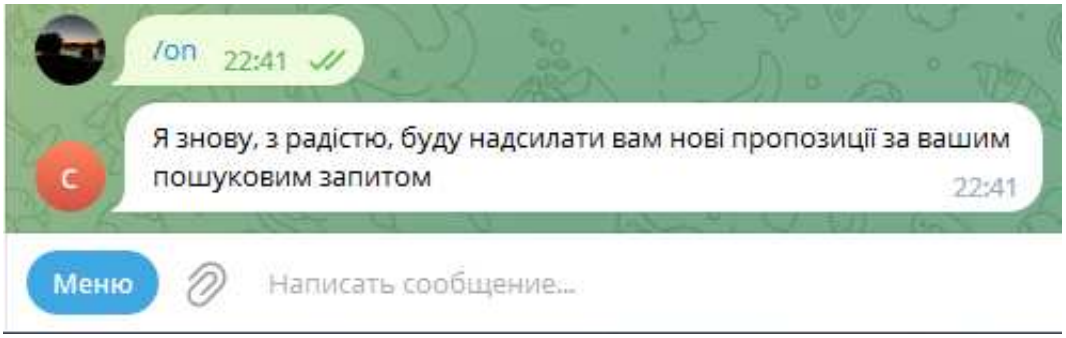


Рисунок 3.16 – Приклад виконання команди «/on»

Бот відправляє оновлення щогодини, якщо з’являються нові пропозиції, що відповідають вашому пошуковому запиту. Це означає, що не потрібно вручну перевіряти сайт AUTO.RIA для пошуку нових оголошень – бот автоматично інформує вас про нові пропозиції (рисунок 3.17).

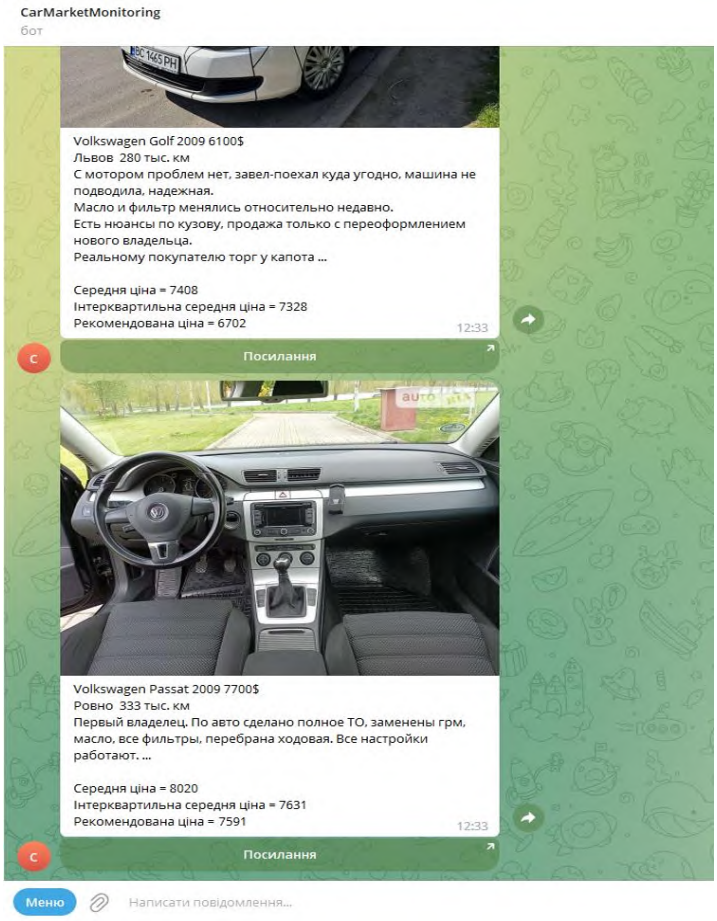


Рисунок 3.17 – Результати пошуку за обраними параметрами

У підсумку, було детально описано процес встановлення, налаштування і використання програмного бота для моніторингу онлайн-ринку вживаних автомобілів в Україні. В інструкції надані докладні кроки по здійсненню початкового налаштування бота, установлення пошукових критеріїв, а також роботи з отриманою інформацією.

Таким чином, була розроблена інструкція користувача, яка слугує як зручний та детальний посібник для ефективної роботи з програмним ботом, що в свою чергу сприятиме підвищенню загальної продуктивності процесу моніторингу оголошень на онлайн-ринку вживаних автомобілів.

Описано основні команди, що використовуються для взаємодії з ботом. Детальне викладення їх функціональності допоможе користувачам легко і ефективно керувати ботом, налаштовувати параметри пошуку, а також контролювати надсилання нових оголошень.

Важливо підкреслити, що розроблена інструкція допоможе користувачам без проблем розпочати використання бота незалежно від їх технічного рівня. Завдяки простій і зрозумілій мові, користувачі матимуть можливість максимально ефективно використовувати усі переваги автоматизованого моніторингу ринку вживаних авто.

3.5 Висновки до розділу

У кваліфікаційній роботі бакалавра розроблено бот для автоматизованого моніторингу онлайн-ринку вживаних авто за допомогою засобів ІАД.

Проаналізувавши основні аспекти програмного бота, детально розглянули його структуру та функціональне призначення. Бот використовує засоби веб-скрапінгу та аналізу даних для відстеження нових оголошень на сайті AUTO.RIA і повідомляє користувачам про нові відповідні оголошення відповідно до їхніх пошукових запитів.

Використання об’єктно-орієнтованого підходу при проєктуванні програми дозволило створити систему, що легко масштабується і модернізується. Окремі компоненти системи можуть бути оновлені або змінені без необхідності втручання в інші частини програми.

В процесі розробки було враховано різні вимоги до розгортання застосунку. Проведене експериментальне тестування допомогло забезпечити високий рівень стабільності та ефективності бота.

У підсумку, розроблений бот є корисним інструментом, який дасть змогу спростити та автоматизувати процес пошуку вживаних авто на онлайн-ринку. Він може бути корисним не тільки для кінцевих користувачів, але і для дослідників та аналітиків, що працюють у сфері автомобільного ринку. Майбутнє бота може включати додаткову інтеграцію з іншими джерелами даних та розширення функціоналу для забезпечення більш глибокого аналізу даних.

Висновки

У результаті виконання кваліфікаційної роботи бакалавра було розроблено Тереграм-бот автоматизованого моніторингу оголошень онлайн-ринку вживаних авто України засобами ІАД. Ця технологія орієнтована на підвищення ефективності процесу моніторингу та опрацювання відповідних оголошень.

Проведений аналіз онлайн-платформ для розміщення оголошень онлайн-ринку вживаних авто в Україні, зокрема AUTO.RIA, дозволив отримати чітке розуміння специфіки цієї сфери. Також було здійснено аналіз відомих методів та засобів ІАД для отримання та оброблення даних з онлайн-платформ.

Результатом роботи стала програмна реалізація системи автоматизованого моніторингу в ролі бота, який використовує засоби веб-скрапінгу для відстеження нових оголошень на сайті AUTO.RIA та повідомляє користувачам про нові оголошення відповідно до їхніх пошукових запитів.

Проведене експериментальне тестування програмного бота в реальних умовах підтвердило його працездатність та ефективність. Результати тестування свідчать про високий рівень стабільності та ефективності програмного забезпечення, що розроблено для розв’язання поставленої задачі.

Таким чином, розроблена програмна система автоматизованого моніторингу оголошень ринку вживаних авто в Україні може бути корисним інструментом як для окремих користувачів, так і для аналітиків ринку. Майбутнє роботи може передбачати додаткову інтеграцію з іншими джерелами даних та розширення функціоналу для забезпечення більш глибокого аналізу даних.

У підсумку, успішно здійснено розробку бота автоматизованого моніторингу оголошень онлайн-ринку вживаних авто засобами ІАД, всі поставлені завдання виконано та мета дипломної роботи була досягнута.

Перелік посилань

1. Попит на авто в інтернеті: тренди, динаміка, конкуренція. *Дослідження онлайн-ринку авто в Україні*. Ольшанський та партнери, 2021. URL: https://olshansky.ua/wp-content/uploads/2021/09/ukr1609.pdf

2. Бевз Д.М., Крикун І.Г. Динаміка автомобільного ринку України. *Прикладні інформаційні технології*. 2021. С. 7-10.

3. Авторинок в Україні: що впливало у 2022 році та чого чекати у 2023. *Мінфін – все про фінанси: новини, курси валют, банки*. URL: https://minfin.com.ua/ua/2022/12/12/97156082/

4. auto.24tv.ua. Український авторинок: аналіз 2022 року та прогноз на 2023 – Auto24. *Новини: останні автоновини сьогодні на порталі auto.24tv.ua*. URL: https://auto.24tv.ua/ukrainskyi_avtorynok_analiz_2022_roku_ta_prohnoz_na_2023_n42455

5. Pavlova O., Radiuk P., Kravchuk S., Kulbachnyi V. Information system for public places and institutions visualization with opportunities of inclusive access and optimal routing. *Computer systems and information technologies*. 2022. Vol. 1, No 6. Pp. 62-68.

6. Radiuk P., Pavlova O., Avsiyevych V., Kovalenko V. Convolutional neural network for parking slots detection. *The 3rd International Workshop on Intelligent Information Technologies & Systems of Information Security (IntelITSIS-2022)* : CEUR-Workshop Proceedings. Vol. 3156. (Khmelnytskyi, Ukraine, 23-25 March 2022). CEUR-WS.org, Aachen, 2022. Pp. 284-293.

7. Яким буде український ринок вживаних авто у 2023 році: Авто новини від AUTO-Consulting – авторинок. *Все про автобізнес: ринок автомобілів, автобусів, вантажівок. AUTO-Consulting*. URL: https://autoconsulting.ua/article.php?sid=52782

8. Котляр А.С. Автоматизація збору та аналізу даних з онлайн-автомагазинів. *Теоретичні та практичні дослідження молодих вчених* : зб. тез доп. 14-ї Міжнар. наук.-практ. конф. магістрантів та аспірантів, 1-4 грудня 2020 р.

/ ред. Є. І. Сокол ; Нац. техн. ун-т "Харків. політехн. ін-т" [та ін.]. Харків : НТУ "ХПІ", 2020. С. 58-59.

9. Довжик В.С. Удосконалення маркетингових технологій просування товару на автомобільному ринку. Здобувач вищої освіти за освітнім ступенем «Магістр» 075 «Маркетинг» / В.С. Довжик Запоріжжя: НУ «Запорізька Політехніка», 2021. 135 с.

10. AUTO.RIA™ – Автобазар №1. Купити і продати авто легко як ніколи. *AUTO.RIA™*. URL: https://auto.ria.com/uk/

11. RST.UA. *Авто базар – Авторинок* України. URL: https://rst.ua/ukr/

12. OLX.ua – автобазар України: купити авто нове або бу, автомобілі на авторинку. *Сайт безкоштовних оголошень OLX.ua*. URL: https://www.olx.ua/uk/transport/legkovye-avtomobili/

13. Іванов Д.Ю. Агрегатор інформації про продаж автомобілів : дипломний проект … бакалавра : 123 Комп'ютерна інженерія / Іванов Дмитро Юрійович. Київ, 2022. 91 с.

14. Radiuk P.M., Skrypnyk T.K., Karlechuk D.T. Applying mental models to making controlled critically safe decisions in IT project management. *Herald of Khmelnytskyi National University. Technical sciences*. 2021. Vol. 301, No 5. Pp. 32-35.

15. Radiuk P.M., Mazurets O.V., Skrypnyk T.K., Moroz O.V. Intelligent data analysis using artificial neural networks for decision making in the education domain. Herald of Khmelnytskyi National University. Technical sciences. 2021. Vol. 303, No 6. Pp. 111-114.

16. Radiuk P., Pavlova O., Hrypynska N. An ensemble machine learning approach for Twitter sentiment analysis. *The 6th International Conference on Computational Linguistics and Intelligent Systems (CoLInS-2022). Volume I: Main Conference* : CEUR-Workshop Proceedings. Vol. 3171. (Gliwice, Poland, 12-13 May 2022). CEUR-WS.org, Aachen, 2022. Pp. 387-397.

17. Analysis of deep learning methods in adaptation to the small data problem solving / I. Krak et al. *Intellectual Systems of Decision Making and Problem of Computational Intelligence (ISDMCI-2022)* : Lecture Notes in Data Engineering,

Computational Intelligence, and Decision Making. 2022. Vol. 149. (Zalizniy Port, Ukraine, 23-27 May 2022). Springer, Cham. Pp. 333-352.

18. Khder M. Web scraping or web crawling: state of art, techniques, approaches and application. *International Journal of Advances in Soft Computing and its Applications*. 2021. Vol. 13, No. 3. Pp. 145-168.

19. Unsupervised document classification integrating web scraping, one-class SVM and LDA topic modelling / A. Thielmann et al. *Journal of Applied Statistics*. 2021. Pp. 1-18.

20. Li Y., Ni P., Chang V. Application of deep reinforcement learning in stock trading strategies and stock forecasting. *Computing*. 2019. Vol. 102, No. 6. Pp. 1305-1322.

21. Big data and data mining technologies application at road transport logistics / P. Mitroshin et al. *Transportation Research Procedia*. 2022. Vol. 61. Pp. 462-466.

22. Telegram Messenger. *Telegram APIs*. URL: https://core.telegram.org/bots/api

23. python-telegram-bot. *python-telegram-bot v20.3*. URL: https://python-telegram-bot.org

24. Crummy. *Beautiful Soup Documentation - Beautiful Soup 4.12.0 documentation* . URL: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

25. Python™ Package Index. *Requests 2.31.0. PyPI*. URL: https://pypi.org/project/requests/

26. Pandas – Python Data Analisis Library. *pandas documentations – pandas 2.0.2 documentations*. URL: https://web.archive.org/web/20190104211407/ http://pandas.pydata.org/pandas-docs/stable/release.html

27. scikit-learn: Machine Learning in Python. *scikit-learn 1.2.2 documentation*. URL: https://scikit-learn.org/stable/#

ДОДАТКИ

Додаток А

Лістинг програмного коду

Програмні коди класу MonitoringBot:

```
from telegram import Update,InlineKeyboardButton, InlineKeyboardMarkup
from telegram.ext import ApplicationBuilder, CommandHandler, MessageHandler,
CallbackContext, filters

from predictors import RFRModel
from scrapers.CarInfoScraper import CarInfoScraper
from scrapers.URLScraper import URLScraper


class MonitoringBot:
  def __init__(self, token, botHelper):
    self.token = token
    self.bot_helper = botHelper
    self.application = ApplicationBuilder().token(self.token).build()
    self.job_queue = self.application.job_queue
    self.register_handlers()
    self.start_scraping()
    self.start_monitoring()
    self.start_model_training()

  async def start(self, update: Update, context: CallbackContext) -> None:
    chat_id = update.effective_chat.id
    await context.bot.send_message(chat_id, self.bot_helper.get_help())

  async def set_command(self, update: Update, context: CallbackContext) -> None:
    chat_id = update.effective_chat.id
    await context.bot.send_message(chat_id, self.bot_helper.set_command(update,
                                    context.args))

  async def get_command(self, update: Update, context: CallbackContext) -> None:
    chat_id = update.effective_chat.id
    await context.bot.send_message(chat_id, self.bot_helper.get_command(update))

  async def on_command(self, update: Update, context: CallbackContext) -> None:
    chat_id = update.effective_chat.id
    await context.bot.send_message(chat_id, self.bot_helper.on_command(update))

  async def off_command(self, update: Update, context: CallbackContext) -> None:
    chat_id = update.effective_chat.id
    await context.bot.send_message(chat_id, self.bot_helper.off_command(update))

  async def unknown_command(self, update: Update, context: CallbackContext) -> None:
    chat_id = update.effective_chat.id
    await context.bot.send_message(chat_id, self.bot_helper.unknown_command())

  def register_handlers(self):
    self.application.add_handler(CommandHandler("start", self.start))
    self.application.add_handler(CommandHandler("set", self.set_command))
    self.application.add_handler(CommandHandler("get", self.get_command))
    self.application.add_handler(CommandHandler("on", self.on_command))
    self.application.add_handler(CommandHandler("off", self.off_command))
    self.application.add_handler(MessageHandler(filters.TEXT & ~filters.COMMAND,
```

```
            self.unknown_command))

    def run(self):
      self.application.run_polling()

    def start_monitoring(self):

      async def car_market_monitoring(context: CallbackContext):
        print("Опрацювання всіх підписок!")
        accounts_manager = self.bot_helper.accounts_manager
        auto_manager = self.bot_helper.auto_manager
        price_predictor = self.bot_helper.price_predictor
        users = accounts_manager.get_users()
        for key in users:
          print(f"Перевіряєм підписку користувача №{key}")
          data = auto_manager.get_search_data(users[key]["queryOld"])
          if len(data["result"]["search_result"]["ids"]) > 0:
            print(data["result"]["search_result"]["ids"])
            for id in data["result"]["search_result"]["ids"][:2]:
              car_data = auto_manager.get_car_data(id)
              avg_price = auto_manager.get_avg_price(car_data)
              predicted_price = price_predictor.predict_price(car_data)

              await context.bot.send_photo(chat_id=key,
               photo=car_data["photoData"]["seoLinkF"],
               caption=
               f'{car_data["title"]} {car_data["autoData"]["year"]} {car_data["USD"]}$\n'
               f'{car_data["stateData"]["name"]} {car_data["autoData"]["race"]}\n'
               f'{car_data["autoData"]["description"][:500]} ...\n'
               f'\narithmeticMean= {round(avg_price["arithmeticMean"])}\n'
               f'interQuartileMean= {round(avg_price["interQuartileMean"])}\n'
               f'predictedPrice= {round(predicted_price)}',
               reply_markup=InlineKeyboardMarkup([[InlineKeyboardButton(
                           text='Посилання',url=self.bot_helper.id_to_url(id))]]))

      return self.job_queue.run_repeating(car_market_monitoring,interval=3600,first=5)

    def start_scraping(self):
      async def scraper():

              URLScraper.threaded_parse_links(search_url='https://auto.ria.com/uk/search/?
              indexName=auto&categories.main.id=1&verified.VIN=1&country.import.usa.not=1&
              price.currency=1&top=11&abroad.not=0&custom.not=1',

                num_of_pages=100,
                num_of_threads=os.cpu_count() - 4,
                file_name='daily_links')
         df_links = pd.read_csv('./data/cars/daily_links.csv')
         links = df_links.values.tolist()
         CarInfoScraper.threaded_parse_car_info(links= links,
                           num_of_threads=os.cpu_count() - 4)

      return self.job_queue.run_daily(callback=scraper,
                                      time=datetime.time(hour=23, minute=30),
                                      days=(0, 1, 2, 3, 4, 5, 6))


    def start_model_training(self):
      async def model_training():
        df = RFRModel.set_train_data('./data/cars/data.csv')
        splited_df = RFRModel.splite_data_frame(df)
        pipeline = RFRModel.pipeline()
        model = RFRModel.train(pipeline, splited_df)
```

```
    RFRModel.save_model(model, './data/models/trained_model.pkl')

  return self.job_queue.run_daily(callback=model_training,
                                  time=datetime.time(hour=23, minute=30),days=(0,))
```

Програмні коди класів CarInfo та CarInfoScraper:

```
from helpers.CSVHelper import CSVHelper

class CarInfo:

  def __init__(self, url):
    self.url = url
    page = requests.get(url)
    self.soup = BeautifulSoup(page.content, "html.parser")
    self.dom = etree.HTML(str(self.soup))

    self.brand = self.get_head_elems()[0]
    self.model = self.get_head_elems()[1]
    self.year = int(self.get_head_elems()[-1])
    self.mileage = self.get_mileage()
    self.fuel_type = self.get_fuel_type()
    self.engine_volume = self.get_engine_volume()
    self.transmission_type = self.get_transmission_type()
    self.owners = self.get_number_of_owners()
    self.price = self.get_price()

  def car_info(self):
    return {
      'brand': self.brand,
      'model': self.model,
      'year': self.year,
      'mileage': self.mileage,
      'fuel_type': self.fuel_type,
      'engine_volume': self.engine_volume,
      'transmission_type': self.transmission_type,
      'owners': self.owners,
      'price': self.price
    }

  def get_head_elems(self):
    head_elem = self.dom.xpath('//*[@id="heading-cars"]/div/h1/text()')[0]
    splitted_head_elem = head_elem.split(' ')
    return splitted_head_elem

  def get_mileage(self):
       milage_elem = self.dom.xpath('//*[@class="technicalinfo"][@id="details"]/dl/
                     dd[2]/span[@class="argument"]/text()')[0]
    milage = ''.join([s for s in milage_elem if s.isdigit()])
    return int(milage)

  def get_fuel_type(self):
    fuel_type_elem = self.dom.xpath('//*[@class="technical-info"][@id="details"]/dl/
                      dd[3]/span[@class="argument"]/text()')[1]
    fuel_type = str(fuel_type_elem)

    return fuel_type.replace(" ", ' ').replace('\n', '').strip()
```

```
def get_engine_volume(self):
  engine_volume_elem = self.dom.xpath('//*[@class="technical-info"][@id="details"]
                          /dl/dd[3]/span[@class="argument"]/text()')[0]
      engine_volume = list(map(float, (float(num) for num in re.findall(r'[-+]?
                                                \d*\.\d+|\d+', engine_volume_elem))))

  try:
    if engine_volume[0] > 9.0 and engine_volume[0] < 100:
      return round(float(engine_volume[0]) / 10, 1)
    if engine_volume[0] > 100.0:
      return round(float(engine_volume[0]) * 1.36 / 200, 1)
    else:
      return float(engine_volume[0])
  except:
    return 2.0

def get_transmission_type(self):
  try:
    # Знаходимо елемент з класом "technical-info" та ідентифікатором "details"
    technical_info = self.soup.find('div', {'class': 'technical-info',
                                                'id': 'details'})
       # Знаходимо всі елементи 'dd'
    dd_elements = technical_info.find_all('dd')
    # Список значень для пошуку
    values_to_search = ["Автомат", "Ручна / Механіка", "Робот", "Варіатор",
                        "Типтронік"]
    # Перебираємо елементи 'dd' та елементи 'span' з класом "argument"
    found_value = None
    for dd_element in dd_elements:
      span_argument = dd_element.find('span', {'class': 'argument'})
      if span_argument:
        for value in values_to_search:
          if value in span_argument.text:
            found_value = value
            break
      if found_value:
        break

    if found_value:
      return found_value
    else:
      return 'Ручна / Механіка'
  except:
    return 'Ручна / Механіка'

def get_number_of_owners(self):
  try:
    # Знаходимо елемент з класом "technical-info" та ідентифікатором "details"
    technical_info_checked = self.soup.find('div', {'class':
                                                'technical-info ticket-checked'})

    # Знаходимо всі елементи 'dd'
    dd_elements = technical_info_checked.find_all('dd')
    # Перебираємо елементи 'dd' та елементи 'span' з класом "argument"
    values_to_search = ["Кількість власників"]
    found_value = None
    for dd_element in dd_elements:
      span_label = dd_element.find('span', {'class': 'label'})
      if span_label:
        for value in values_to_search:
          if value in span_label.text:
            span_argument = dd_element.find('span', {'class':
                                                    'argument'})
```

```
            found_value = span_argument.text
            break
        if found_value:
          break

      if found_value: return int(found_value)
      else: return 1
    except: return 1

  def get_price(self):
    price_elem = self.dom.xpath('//*[@id="showLeftBarView"]
                  /section[1]/div[1]/strong/text()')[0]

    price = ''.join([s for s in price_elem.split() if s.isdigit()])
    return int(price)
```

```
class CarInfoScraper:
  def parse_car_info(self, links):
    for i in tqdm(range(len(links))):
      try:
        res = CarInfo(links[i]).car_info()
        CSVHelper.write(path='./data/cars/data.csv', data=[res],
                        headers=res.keys())
      except:
        continue

  def threaded_parse_car_info(self, links, num_of_threads):
    chunk_size = len(links) // num_of_threads

    # Створення потоків
    threads = []
    for i in range(self.num_of_threads):
      start = i * chunk_size
      if i != self.num_of_threads - 1:
        end = start + chunk_size
      else:
        end = len(self.links)
      thread = threading.Thread(self.parse_car_info,
                                args=(self.links[start:end], ))

      threads.append(thread)

    # Запуск потоків
    for thread in threads:
      thread.start()

    # Очікування завершення потоків
    for thread in threads:
      thread.join()
```

Програмні коди класів URL та URLScraper:

```
from helpers.CSVHelper import CSVHelper
```

```
class URL:
  def __init__(self, dom, item_id):
    self.link = self.get_link(item_id, dom)

  def get_link(self, item_id, dom):
    try:
      elems = dom.xpath(f'//*[@id="searchResults"]
                          /section[{item_id}]/div[4]/div[2]/div[1]/div/a')

      link = elems[0].attrib['href']
      return link
    except:
      return ''
```

```
class URLScraper:

  def parse_links(self, search_url, start, end, file_name):
    for page_id in tqdm(range(start, end)):
      tmp_links = []
      url = f"{search_url}&size=100&page={page_id}"
      url = urllib.parse.quote(url, safe='/:?&=')
      page = requests.get(url)
      soup = BeautifulSoup(page.content, "html.parser")
      dom = etree.HTML(str(soup))
      for item_id in range(1, 99):
        link = URL(dom, item_id)
        if link.link != '':
          tmp_links.append(link)
        else:
          pass
      CSVHelper.write(path=f'./data/cars/{file_name}.csv', data=tmp_links,
headers=['links'])


  def threaded_parse_links(self, search_url, num_of_pages, file_name, num_of_threads):
    chunk_size = num_of_pages // num_of_threads
    threads = []
    for i in range(num_of_threads):
      start = 1 + i * chunk_size
      if i != num_of_threads - 1:
        end = start + chunk_size
      else:
        end = 1 + num_of_pages
      thread = threading.Thread(target=self.parse_links, args=(search_url, start, end,
file_name))
      threads.append(thread)
    for thread in threads:
      thread.start()
    for thread in threads:
      thread.join()
```

Програмні коди класів Model та PricePredictor:

```
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.ensemble import RandomForestRegressor
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
```

```
class Model:
  def __init__(self):
    self.categorical_features = ["brand", "model", "fuel_type", "transmission_type"]
    self.numeric_features = ["year", "mileage", "engine_volume", "owners"]

  def set_train_data(self, file_path):
    data_frame = pd.read_csv(file_path)
    return data_frame

  def load_hyperparams(self):
    with open('./data/models/params/best_params.json', "r") as file:
      data = json.load(file)
    return data

  def save_hyperparams(self, data):
    with open('./data/models/params/best_params.json', 'w') as file:
      file.write(json.dumps(data))

  def load_model(model_path):
    return joblib.load(model_path)

  def save_model(self, model, model_path):
    return joblib.dump(model, model_path)

  def train(self, pipeline, splited_df):
    data = self.hyperparams()
    params = data["params"]
    model = pipeline.set_params(**params)
    model.fit(splited_df["X_train"], splited_df["Y_train"])
    print(pipeline.score(splited_df["X_test"], splited_df["Y_test"]))
    return model

  def tune_hyperparams(self, pipeline, splitted_data_frame):
    param_grid = {
      "regressor__n_estimators": [25, 50, 75,100],
      "regressor__max_depth": [None, 15, 30, 45, 60],
      "regressor__min_samples_split": [10, 15, 20],
      "regressor__min_samples_leaf": [3, 5, 7],
      "regressor__max_features": ['auto', 'sqrt', 'log2'],
      "regressor__bootstrap": [True, False]
    }

    grid_search = GridSearchCV(
      pipeline,
      param_grid,
      cv=5,
      scoring="neg_mean_squared_error",
      verbose=1,
      n_jobs=os.cpu_count() - 3)

    grid_search.fit(splitted_data_frame["X_train"], splitted_data_frame["Y_train"])
    to_json = {'params': grid_search.best_params_}
    self.save_hyperparams(to_json)

    return grid_search.score(splitted_data_frame["X_test"],
                             splitted_data_frame["Y_test"])

  def pipeline(self):
    feature_transformer = ColumnTransformer(
      transformers=[
        ("num", StandardScaler(), self.numeric_features),
        ("cat", OneHotEncoder(handle_unknown="ignore"),
```

```
                self.categorical_features),
      ]
    )
    model = RandomForestRegressor()
    pipeline = Pipeline([("preprocessor", feature_transformer), ("regressor", model)])
    return pipeline

  def splite_data_frame(self, data_frame):
    X = data_frame.drop(columns=["price"])
    Y = data_frame["price"]
    X_train, X_test, Y_train, Y_test = train_test_split(X, Y, testsize=0.2,
                                                        random_state=42)

    splitted_data_frame = {
      "X_train": X_train, "Y_train": Y_train,
      "X_test": X_test, "Y_test": Y_test
    }
    return splitted_data_frame
```

```
class PricePredictor:

  def __init__(self, model_path):
    self.model = self.load_model(model_path)

  def predict_price(self, data):
    car_data = {"brand": data["markName"],
          "model": data["modelName"],
          "fuel_type": data["autoData"]["fuelName"],
          "transmission_type": data["autoData"]["gearboxName"],
          "year": data["autoData"]["year"],
          "mileage": data["autoData"]["raceInt"],
          "engine_volume": data["autoData"][" engine_volume "],
          "owners": data["countOwners"]
    }

    df = pd.DataFrame([car_data])
    prediction = self.model.predict(df)

    return prediction[0]

  def load_model(self, model_path):
    return joblib.load(model_path)
```

Програмні коди класів BotHelper, CSVHelper та файлу main

```
class BotHelper:

  def __init__(self, accountsManager, autoManager, pricePredictor ):
    self.accounts_manager = accountsManager
    self.auto_manager = autoManager
    self.price_predictor = pricePredictor

  def get_help(self):
    help_text = 'Я експериментальний бот>. Готовий допомогти вам відслідковувати
              будь-який пошуковий запит з сайту AUTO.RIA.com.' + "\n" \
      'Раз на годину я буду переглядати ваш пошук і надсилати нові пропозиції з
              бази AUTO.RIA (якщо вони для вас з'являться).' + "\n\n" \
```

```
    'Я розумію наступні команди:' + "\n" \
    '/start – Надішлю цей документ' + "\n" \
    '/set [autoria search url] – запам'ятаю ваш пошуковий запит, наприклад
           /set https://auto.ria.com/uk/search/?category_id=1&type[5]=6 ' + "\n" \
    '/get-нагадаю вам пошуковий запит, який відслідковую для вас' + "\n" \
    '/on-відновлю відправку для вас нових пропозицій з бази AUTO.RIA' + "\n" \
    '/off- припиню надсилати вам нові пропозиції за вашим пошуковим запитом'

  return help_text

def set_command(self, update, args):
  if not args:
    response = "Дайте мені команду /set [пошуковий запит з AUTO.RIA.com],
          де \"пошуковий запит з AUTO.RIA.com\"- це скопійована
          адресна строка браузера при пошуку на AUTO.RIA.com,
          наприклад /set https://auto.ria.com/uk/search/?category_id=1&type[5]=6"

    return response

  url = args[0]
  myURL = urlparse(url)

  if not myURL.hostname.endswith('auto.ria.com'):
    response = f"Ви вказали невідомий мені хост {myURL.hostname}"
    return response

  if not myURL.path.endswith('search/'):
    response = f'Пошукова строка має бути скопійована зі сторінки пошуку б/у
                авто https://auto.ria.com/search/?...,
                а у вас інша адреса: "{myURL.hostname}"'

    return response

  query = myURL.query or ''
  hash = myURL.fragment

  if not query:
    query = hash[0:]
  else:
    query = query[0:]

  self.accounts_manager.make_user(update.message.chat_id, query)
  response = f'З цього моменту я буду відслідковувати всі нові оголошення за
              запитом {url}'
  return response

def get_command(self, update):
  user_config = self.accounts_manager.get_user(update.message.chat_id)
  if 'query' in user_config and user_config['query'] is not None:
    response=f'Відслідковую запит:
                https://auto.ria.com/search/?{user_config["query"]}'
  else:
    response = 'Пошукова строка не задана: Встановіть пошукову строку
                за допомогою команди /set'
  return response

def off_command(self, update):
  self.accounts_manager.switch_watching_user(update.message.chat_id, False)
  response ='Я призупинив відслідковування нових оголошень для вас'
  return response

def on_command(self, update):
  result=self.accounts_manager.switch_watching_user(update.message.chat_id, True)
```

```
    if not result:
      response ='Я не можу виконати цю команду: ви не вказали пошукову строку за
                допомогою команди /set'
    else:
      response ='Я знову, з радістю, буду надсилати вам нові пропозиції '
    return response

  def unknown_command(self):
    response = 'Я експериментальний робот і, на жаль, вашу команду не розумію.
                Напишіть мені /start, щоб дізнатися команди, які я вже вивчив.'
    return response

  def id_to_url(self, id):
    link = f'https://auto.ria.com/auto___{id}.html'
    return link

  def url_to_id(self, url):
    pattern = r'https://auto\.ria\.com/uk/auto_[\w_]+_(\d+)\.html'
    match = re.search(pattern, url)

    if match:
      return int(match.group(1))
    else:
      raise ValueError(f"Неможливо зчитати ID з URL: {url}")
```

```
class CSVHelper:
  def write(self, path, data, headers):
    df = pd.DataFrame(data, columns=headers)
    if Path.is_file(path):
      df.to_csv(path, mode='a', index=False, header=False)
    else:
      df.to_csv(path, mode='w', index=False)

  def read(self, path):
    df = pd.read_csv(path, header=None)
    return df
```

Програмні коди класів AccountsManager та AutoManager:

```
class AccountsManager:
  def __init__(self, api_config):
    self.config = api_config
    self.users = {}
    self.data_directory = Path.cwd() / 'data/users'

  def get_user_filename(self, user_id):
    return self.data_directory / f'{user_id}.json'

  def write_user(self, user_id, user_config):
    user_file = self.get_user_filename(user_id)
    with user_file.open('w') as f:
      json.dump(user_config, f, indent=2)

  def get_user(self, user_id):
    user_file = self.get_user_filename(user_id)
    if user_file.exists():
      with user_file.open() as f:
        user_config = json.load(f)
```

```
      return user_config
    return {}

  def get_users(self):
    users = {}
    for file in self.data_directory.glob('*.json'):
      user_id = int(file.stem)
      with file.open() as f:
        user = json.load(f)
      if user['watchStatus']:
        users[user_id] = user
    return users

  def make_user(self, user_id, query):
    user_config = self.get_user(user_id)
    user_config['query'] = query
    user_config['setDate'] = int(time.time() * 1000)
    user_config['watchStatus'] = True

    print(f"Create settings for user: {user_id} ...")

    try:
      response = requests.get(self.config['queryTranslateFromNew'] + '?' + query)
      data = response.json()
      user_config['queryOld'] = unquote(data['string'])
      self.write_user(user_id, user_config)
    except Exception as err:
      print("API call failed...")

  def switch_watching_user(self, user_id, watch_status):
    user_file = self.get_user_filename(user_id)
    if user_file.exists():
      with user_file.open() as f:
        user_config = json.load(f)
      user_config['setDate'] = int(time.time())
      user_config['watchStatus'] = watch_status
      self.write_user(user_id, user_config)
      return True
    return False
```

```
class ApiAutoManager:
  def __init__(self, config):
    self.search_url = config["searchUrl"]
    self.info_url = config["infoUrl"]
    self.avg_price_url = config["avgPriceUrl"]
    self.api_key = config["key"]

  def get_search_data(self, search_params):
    url = f"{self.search_url}?{search_params}&top=1&api_key={self.api_key}"
    response = requests.get(url)
    data = json.loads(response.text)
    return data

  def get_car_data(self, announcement_id):
    url = f"{self.info_url}?api_key={self.api_key}&auto_id={announcement_id}"
    response = requests.get(url)
    data = json.loads(response.text)
    return data

  def get_avg_price(self, announcement_data):
    if (announcement_data['autoData']['year'] +
                  round(announcement_data['autoData']['year'] * 0.0015)) > 2023:
```

```
            to_year = 2023
        else:
            to_year = announcement_data['autoData']['year'] +
                          round(announcement_data['autoData']['year'] * 0.0015)
        url = f"{self.avg_price_url}?api_key={self.api_key}
                  &marka_id={announcement_data['markId']}" \
           f"&model_id={announcement_data['modelId']}" \
           f"&raceInt={announcement_data['autoData']['raceInt'] -
                  round(announcement_data['autoData']['raceInt'] * 0.2)}" \
           f"&raceInt={announcement_data['autoData']['raceInt'] +
                  round(announcement_data['autoData']['raceInt'] * 0.2)}" \
           f"&yers={announcement_data['autoData']['year'] -
                  round(announcement_data['autoData']['year'] * 0.0015)}&yers={to_year}"
        response = requests.get(url)
        data = json.loads(response.text)
        return data
```

Файл main – точка входу в програму

```
from bots.PTBMonitoringBot import PTBMonitoringBot
from helpers.PTBBotHelper import BotHelper
from managers.AccountsManager import AccountsManager
from managers.ApiAutoManager import ApiAutoManager
from predictors.PricePredictor import PricePredictor
import configparser

config = configparser.ConfigParser()
config.read('config.ini')
accountsManager = AccountsManager(config['api'])
autoManager = ApiAutoManager(config['api'])
pricePredictor = PricePredictor('./data/models/trained_model.pkl')
botHelper = BotHelper(accountsManager, autoManager, pricePredictor)
bot = PTBMonitoringBot(config['bot']['token'], botHelper)

if __name__ == "__main__":
    bot.run()
```

Додаток Б

Презентаційний матеріал

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

АВТОМАТИЗОВАНИЙ МОНІТОРИНГ ОГОЛОШЕНЬ ОНЛАЙН-РИНКУ ВЖИВАНИХ АВТО ЗАСОБАМИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ДАНИХ

Виконав:
студент 4 курсу, групи КН-19-2
Ціцьвіра Імір Олегович

Керівник:
старший викладач кафедри КН
Радюк Павло Михайлович

Актуальність

З розвитком технологій автоматизовані системи можуть надавати корисну інформацію, згідно з індивідуальними потребами та уподобаннями користувачів, що сприяє зручності та ефективності пошуку автомобілів.

Автоматизований моніторинг та інтелектуальний аналіз даних на онлайн-ринку вживаних автомобілів мають значну актуальність. Враховуючи величезний обсяг даних та швидкість змін на ринку, автоматизація дозволяє ефективно відстежувати тенденції, виявляти нові можливості та своєчасно реагувати на зміни. Застосування інтелектуального аналізу даних забезпечує об'єктивність та знаходження прихованих закономірностей, які можуть бути непомічені при ручному аналізі.

Автоматизація також допомагає оптимізувати використання часу та ресурсів, зосереджуючись на стратегічних завданнях та прийнятті обґрунтованих рішень. Крім того, автоматизовані системи можуть адаптуватися до індивідуальних потреб користувачів, що підвищує зручність та ефективність пошуку автомобілів.

Отже, впровадження автоматизованого моніторингу та інтелектуального аналізу даних у сфері онлайн-ринку вживаних автомобілів є важливим кроком для забезпечення більшої ефективності, оперативності та адаптивності до змін.

Мета і задачі роботи

Метою кваліфікаційної роботи бакалавра є підвищення ефективності процесу моніторингу оголошень онлайн-ринку вживаних авто України, для чого слід вирішити задачі:

1. Провести аналіз онлайн-платформ для розміщення оголошень онлайн-ринку вживаних авто України.
2. Провести аналіз відомих методів та засобів інтелектуального аналізу даних для отримання та оброблення даних з онлайн-платформ та обрати найкращий для розв'язання задачі автоматизованого моніторингу оголошень.
3. Реалізувати обраний засіб інтелектуального аналізу даних у вигляді програмного боту системи обліку миттєвими повідомленнями для розв'язання задачі автоматизованого моніторингу оголошень.
4. Провести експериментальне тестування програмного бота в реальних умовах.

Розроблений Telegram бот автоматизованого моніторингу онлайн-ринку вживаних авто засобами інтелектуального аналізу даних має виконувати такі основні функцій:

- автоматизований моніторинг нових оголошень онлайн-ринку вживаних авто за параметрами користувача та відправка даних в Telegram;
- обрахунок середньоринкової ціни для кожного з оголошень;
- прогнозування ціни за допомогою моделі машинного навчання.

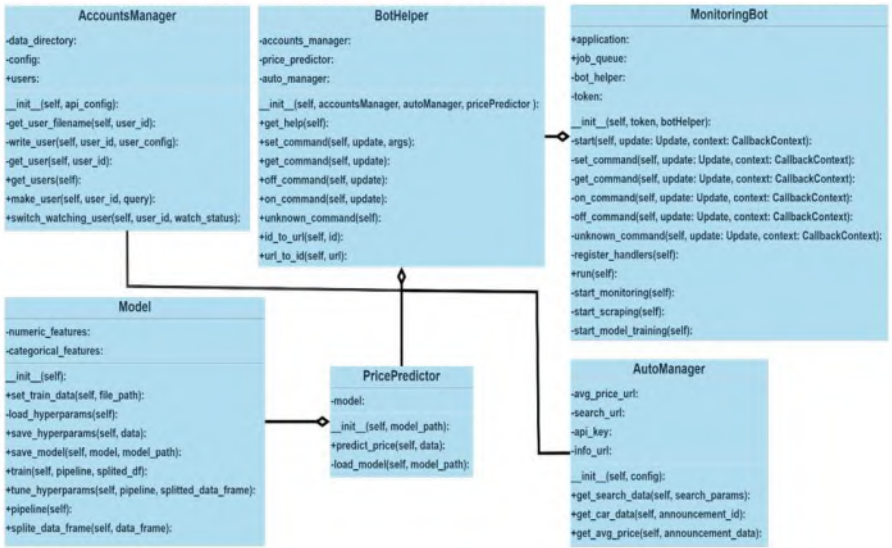
Схема способу автоматизованого моніторингу оголошень онлайн-ринку вживаних авто

5

Функціональна структура Telegram-бота для автоматизованого моніторингу оголошень онлайн-ринку вживаних авто

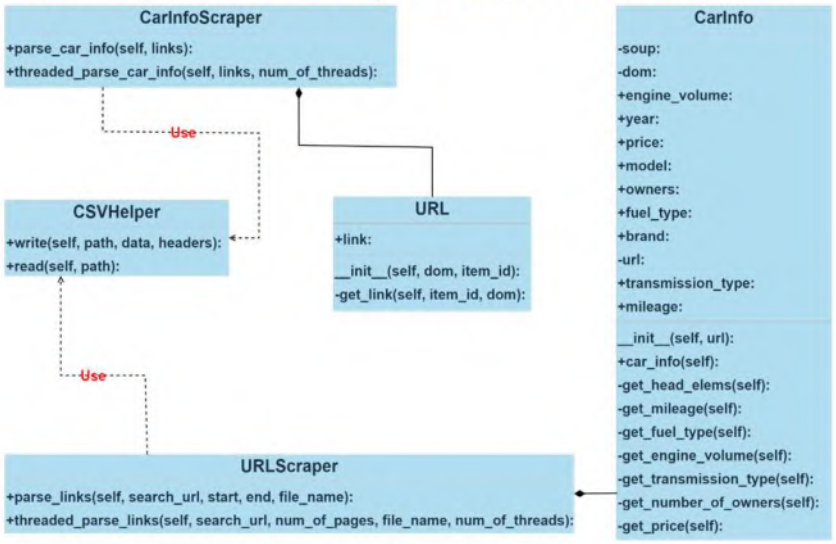
6

Діаграма класів програмної реалізації бота автоматизованого моніторингу онлайн-ринку вживаних авто засобами ІАД



7

Діаграма класів програмної реалізації бота автоматизованого моніторингу онлайн-ринку вживаних авто засобами ІАД



8

Програмна реалізація боту автоматизованого моніторингу онлайн-ринку вживаних авто засобами ІАД

Встановлення параметрів пошуку

Початок роботи з ботом

9

Програмна реалізація боту автоматизованого моніторингу онлайн-ринку вживаних авто засобами ІАД

Відправка знайдених оголошень користувачам

10

Висновки

У рамках виконання кваліфікаційної роботи бакалавра було виконано **розробку й тестування бота** автоматизованого моніторингу онлайн – ринку вживаних авто засобами ІАД. Зокрема, було проведений аналіз онлайн-платформ для розміщення оголошень онлайн-ринку вживаних авто та також було здійснено детальний аналіз відомих методів та засобів інтелектуального аналізу даних для отримання та оброблення даних з онлайн-платформ.

Розроблена **програмна реалізація** бота автоматизованого моніторингу онлайн – ринку вживаних авто засобами ІАД :

- Автоматизований моніторинг нових оголошень онлайн-ринку вживаних авто за параметрами користувача та відправка даних в Telegram;
- Обрахунок середньоринкової ціни для кожного з оголошень;
- Прогнозування ціни за допомогою моделі машинного навчання.

У якості засобів розробки було обрано мову програмування python, бібліотеки python-telegram-bot , BeautifulSoup, requests, scikit-lear.

Mon May 29 12:15:22 EEST 2023, Петровський Сергій Степанович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 4.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 8%**

	Документ		Сумарний збіг по Базі Даних	
ID: 114225 Назва: КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА Додано в БД: 2023-05-29 Автора: І.О. Ціцьвіра Керівники: П.М. Радюк Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	67946	1047	4173 (6%)	60 (6%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK

Ім'я користувача:
Кафедра КН

Дата перевірки:
29.05.2023 13:11:47 EEST

Дата звіту:
29.05.2023 13:14:24 EEST

ID перевірки:
1015298334

Тип перевірки:
Doc vs Internet + Library

ID користувача:
100005671

Назва документа: **КН-19-2 Ціцьвіра**

Кількість сторінок: 61 Кількість слів: 11282 Кількість символів: 85426 Розмір файлу: 1.55 MB ID файлу: 1014970215

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

4.21%
Схожість

Найбільша схожість: **2.01% з джерелом з Бібліотеки** (ID файлу: 1014959353)

3.44% Джерела з Інтернету	277	Сторінка 63
2.85% Джерела з Бібліотеки	85	Сторінка 64

0.26% Цитат

Цитати	2	Сторінка 65

Не знайдено жодних посилань

0%
Вилучень

Деякі джерела вилучено автоматично (фільтри вилучення: кількість знайдених слів є меншою за **8 слів** та **0%**)

0% Вилучення з Інтернету	1	Сторінка 66

Немає вилучених бібліотечних джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування	14 сторінок

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ КАФЕДРИ КОМП'ЮТЕРНИХ НАУК
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: *Автоматизований моніторинг оголошень онлайн-ринку вживаних авто засобами інтелектуального аналізу даних*

Автор: *студент групи КН-19-2 Ціцьвіра Імір Олегович*

Спеціальність: *122 – Комп'ютерні науки*

Освітня програма: *освітньо-професійна*

Науковий керівник: *док. філ., ст. викл. Радюк П.М.*

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	***відповідає***
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

Запозичення, виявлені в роботі Ціцьвіри І.О., не є плагіатом, оскільки: запозичення розміщені в розділі огляду існуючих підходів, не описують безпосередньо авторську роботу і не стосуються її результатів; усі запозичення фрагментарні; до запозичень входять фрагменти програмного коду, що не мають авторства і містять поширені конструкції; поміж запозичень знаходяться загальновідомі терміни та скорочення.

Обсяг запозичень, визначений системами виявлення збігів/ ідентичності/схожості, складає:

- за системою Anti-Plagiarism: 4.0%;

- за системою Unicheck: 4.21 %.

Отже, запозичення є допустимими та відносяться до описаних вище і адресуються до першоджерел, що, з урахуванням наведених обґрунтувань, свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ *Павло РАДЮК*

Гарант ОП _____ *Олександр МАЗУРЕЦЬ*

Завідувач кафедри КН _____ *Олександр БАРМАК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

ВІДГУК НАУКОВОГО КЕРІВНИКА

на кваліфікаційну роботу бакалавра

студента *гр. КН-19-2 Ціцьвіри Іміра Олеговича*

за темою *Автоматизований моніторинг оголошень онлайн-ринку вживаних авто засобами інтелектуального аналізу даних*

1. Актуальність теми

Упродовж останніх років ринок продажу вживаних автомобілей в Україні стрімко зростає. Щодня з'являються десятки нових пропозицій на онлайн-платформах щодо продажу вживаних авто, які вкрай важко опрацювати самостійно. Відповідно виникає потреба в розробленні нового програмного рішення для автоматизації процесу моніторингу оголошень онлайн-ринку вживаних авто.

2. Відповідність роботи предметній області Стандарту спеціальності 122 Комп'ютерні науки

Відповідно до стандарту бакалавра вищої освіти України спеціальності 122 – Комп'ютерні науки, описом предметної галузі, об'єктом та предметом вивчення є математичні, інформаційні та імітаційні моделі реальних явищ, об'єктів, систем і процесів та методи й технології отримання, зберігання, обробки, передачі та використання інформації. Метою поданої роботи є підвищення ефективності процесу моніторингу оголошень онлайн-ринку вживаних авто України. Мету роботи досягнуто завдяки використанню методів, засобів та технологій розв'язання теоретичних і прикладних задач, що виникають у процесі проєктування та розроблення інформаційних технологій. Отже, результати виконання кваліфікаційної роботи відповідають стандарту бакалавра спеціальності 122 – Комп'ютерні науки.

3. Професійні та особистісні якості бакалавра

Під час виконання кваліфікаційної роботи студент Ціцьвіра Імір Олегович проявив себе кваліфікованим фахівцем та дисциплінованим студентом, вчасно виконуючи поставлені перед ним завдання. Студент опанував компетентності та засвоїв результати навчання, що відповідають виконанню освітньо-професійної програми рівня вищої освіти «Бакалавр» за спеціальністю 122 – Комп'ютерні науки.

4. Ступінь самостійності під час виконання кваліфікаційної роботи

Студент особисто одержав результати кваліфікаційної роботи та обґрунтував їхню практичну значущість внаслідок виконання ним усіх поставлених завдань.

5. Ступінь оволодіння методами дослідження

У процесі аналізу способу автоматизованого моніторингу оголошень онлайн-ринку вживаних авто засобами інтелектуального аналізу даних та проєктування і розроблення на його основі програмного бота студент Ціцьвіра І.О. продемонстрував достатній рівень компетентностей та володіння необхідними інструментами й обладнанням, методами, методиками та технологіями галузі інформаційних технологій.

6. Повнота та якість розкриття теми роботи

Тема роботи повністю обґрунтована й розкрита; актуальність предметної галузі та відомі дослідження щодо обраної тематики проаналізовані достатньо. Усі поставлені завдання в роботі виконані повністю. Розроблене програмне забезпечення для валідації та верифікації системи автоматизованого моніторингу оголошень онлайн-ринку вживаних авто засобами інтелектуального аналізу даних відповідає технічним вимогам спеціальності 122 – Комп'ютерні науки.

7. Логічність, послідовність, аргументованість, літературна грамотність викладення матеріалу

Матеріал кваліфікаційної роботи Ціцьвіри І.О. подано логічно, послідовно, аргументовано та є таким, що відповідає поставленій меті. Мова і стиль викладення роботи відповідають стандартам, що забезпечує доступність сприймання матеріалу і відповідає вимогам до сучасних кваліфікаційних робіт.

8. Можливість практичного застосування кваліфікаційної роботи бакалавра, окремих її частин

Розроблене в роботі програмне забезпечення може бути використане торговими агентами та потенційними клієнтами ринку вживаних авто для підвищення ефективності моніторингу оголошень онлайн-ринку вживаних авто України.

9. Висновок про можливість допуску кваліфікаційної роботи бакалавра до захисту, на яку оцінку заслуговує робота

З огляду на рівень виконання та забезпечення всіх необхідних вимог, вважаю, що кваліфікаційна робота Ціцьвіри Іміра Олеговича може бути допущена до захисту. Рекомендована оцінка – «добре».

Керівник _____ *док. філ., ст. викл. каф. КН Павло РАДЮК*

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МОН УКРАЇНИ

Кафедра комп'ютерних наук

РЕЦЕНЗІЯ

на кваліфікаційну роботу бакалавра

студента *гр. КН-19-2 Ціцьвіри Іміра Олеговича*

за темою: *Автоматизований моніторинг оголошень онлайн-ринку вживаних авто засобами інтелектуального аналізу даних*

1. Актуальність обраної теми

В умовах постійного розвитку технологій та збільшення онлайн-торгівлі виникає необхідність у впровадженні нових рішень на основі інформаційних технологій роздрібної торгівлі онлайн. Водночас особливістю сучасного етапу розвитку ринку вживаних автомобілів є зростання ролі інтернет-платформ. Тому, з огляду на постійне зростання популярності онлайн-платформ для продажу вживаних автомобілів і брак ефективних інструментів для автоматизованого моніторингу цих ринків, спрямованість роботи вважаю доречною, а тему – досить актуальною.

2. Повнота розкриття мети та завдань роботи

У результаті виконання кваліфікаційної роботи бакалавра мету та завдання роботи розкрито повністю. Проведено глибокий аналіз предметної галузі, розглянуто та проаналізовано низку літературних джерел щодо обраної тематики наявних засобів інтелектуального аналізу даних. Автором чітко визначено структуру інформаційної технології та реалізовано Telegram-бот для моніторингу оголошень щодо продажу вживаних автомобілів засобами інтелектуального аналізу даних. Отже, мета та завдання поданої до захисту роботи є розкритими.

3. Зміст кожного розділу роботи

Кожен розділ роботи був чітко структурований і добре оформлений. У першому розділі кваліфікаційної роботи проведено аналіз предметної області та наявних засобів інтелектуального аналізу даних, здійснено огляд популярних онлайн-платформ для розміщення оголошень онлайн-ринку вживаних авто. Другий розділ присвячений проєктуванню функціональної структури інформаційної системи автоматизованого моніторингу оголошень онлайн-ринку вживаних авто засобами інтелектуального аналізу даних, розробленню інтерфейсу даної інформаційної системи. У третьому розділі наведено та описано програмну реалізацію спроєктованої інформаційної системи, проведено тестування системи та подано інструкцію з її використання. Загалом, розділи роботи логічно пов'язані та послідовно розкривають її мету.

4. Оцінка розробленої інформаційної системи, її практична цінність

Розроблена інформаційна система показала себе як ефективний інструмент для автоматизації процесу моніторингу оголошень про продаж вживаних авто, що забезпечує користувачів ефективним механізмом та зручним інтерфейсом для прийняття рішень щодо купівлі вживаних авто онлайн. Результат роботи інформаційної системи повністю відповідає поставленій меті та завданням роботи та може бути використаний за практичним призначенням.

5. Якість оформлення кваліфікаційної роботи бакалавра

Записка до кваліфікаційної роботи виконана якісно, з логічним викладенням матеріалу та наведенням вагомих аргументів. Текст записки є чітким та зрозумілим; вдало використані графіки та таблиці. Робота виконана на високому рівні, відповідно до норм наукового оформлення.

6. Недоліки кваліфікаційної роботи бакалавра

Робота містить деякі недоліки. У розділах не розкрито питання про вибір конкретних методів інтелектуального аналізу даних, а також процес валідації цих методів. Також у розробленій інформаційній системі не реалізовано повністю налаштування автоматизованого моніторингу та можливість одночасного моніторингу за декількома запитами від користувача.

7. Загальний висновок (допускається чи не допускається до захисту), та оцінка на яку заслуговує кваліфікаційна робота.

З огляду на рівень виконання та забезпечення всіх необхідних вимог вважаю, що подана кваліфікаційна робота бакалавра може бути допущена до захисту. Рекомендована оцінка – «добре»

Рецензент к.ф.-м.н., доц. каф. ВМКВ Рамський А.О.