МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)
червня 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 – Комп'ютерних наук,
освітньо-наукової програми «Інформатика»
на тему: «Інформаційна система визначення та прогнозування курсу криптовалют»
здобувача групи ІН-06-2 Єшенка Назарія Владиславовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Назарій ЄШЕНКО
(підпис)

Керівник, старш. викл. каф. КН Олег БЕРЕСТ

_____
(підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття освітнього ступеня бакалавра
зі спеціальності 122 – Комп'ютерних наук, освітньо-професійної програми «Інформатика»
здобувача групи ІН-06-2 Єшенко Н.В.

1. Тема роботи: «Інформаційна система визначення та прогнозування курсу криптовалют» затверджую наказом по СумДУ від «22» квітня 2024 № 0414-VI
2. Термін здачі здобувачем кваліфікаційної роботи до 13 червня 2024 року
3. Вхідні дані до кваліфікаційної роботи
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз проблеми предметної області, постановка й формування завдань дослідження.
2) Огляд технологій, для розробки системи прогнозування на основі історичних даних.
3) Розробка інформаційної системи автоматизованого прогнозування курсу за історичними даними.
4) Аналіз результатів.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до Керівник
_____ _____ _____
(підпис) (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*		
2	*Огляд технологій, для розробки системи прогнозування на основі історичних даних.*		
3	*Розробка інформаційної системи автоматизованого прогнозування курсу за історичними даними.*		
4	*Аналіз отриманих результатів*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ Керівник _____

(підпис) (підпис)

АНОТАЦІЯ

Записка: 76 стр., 20 рис., 5 додатків, 16 використаних джерел.

Обґрунтування актуальності теми роботи – Тема кваліфікаційної роботи є актуальною через її зосередженість на вирішенні нагальної практичної проблеми – визначення та прогнозування курсу криптовалют. Це передбачає розробку методів, моделей та інформаційної технології, що дозволяють забезпечити точний та швидкий аналіз криптовалютного ринку. В умовах стрімкого розвитку фінансових технологій та зростання інтересу до криптовалют ця тема стає надзвичайно важливою для вивчення та застосування у фінансовому секторі.

Об’єкт дослідження — Процес визначення та прогнозування курсів криптовалют.

Мета роботи — Розробити інформаційну систему, що використовує алгоритми машинного навчання та аналіз великих даних для визначення та прогнозування курсу криптовалют.

Методи дослідження — Інструменти для збору та обробки великих обсягів даних, алгоритми машинного навчання для фінансового аналізу.

Результати — Створено інформаційну систему для визначення та прогнозування курсів криптовалют. Ця система забезпечує аналіз історичних даних, прогнозування з використанням різних моделей машинного навчання та візуалізацію результатів. Проведене тестування на реальних даних показало високу точність прогнозування курсів криптовалют.

ІНФОРМАЦІЙНА СИСТЕМА ВИЗНАЧЕННЯ ТА ПРОГНОЗУВАННЯ КУРСУ КРИПТОВАЛЮТ, PYTHON, PANDAS, AIOGRAM, SKLEARN, MATPLOTLIB.

ЗМІСТ

ВСТУП

Зростання цін на криптовалюту, зміна їхнього курсу та велика волатильність створюють певні виклики та можливості для інвесторів та торговців. В умовах такої нестабільності інструменти для аналізу та прогнозування курсів криптовалют стають надзвичайно важливими.

Вибір телеграм-бота для реалізації інформаційної системи для визначення та прогнозування курсу криптовалют може бути дуже актуальним і доцільним з кількох причин.

По-перше, телеграм-боти надають можливість доступу до інформації та функціоналу в будь-який час та з будь-якого місця. Користувачі можуть отримувати оновлення щодо курсів криптовалют та результатів прогнозування безпосередньо на свої смартфони або інші пристрої з доступом до Telegram.

По-друге, телеграм-боти можуть надавати інтерактивний інтерфейс, що дозволяє користувачам отримувати не лише інформацію, але і взаємодіяти з системою, наприклад, отримувати сповіщення про зміни курсів, вводити запити на аналіз певних криптовалютних активів тощо.

По-третє, Telegram відомий своєю високою захищеністю та конфіденційністю даних, що є критичним аспектом для інформаційних систем, пов'язаних з фінансовими даними та особистою інформацією користувачів.

Використання телеграм-бота може бути досить простим у розгортанні та управлінні, що дозволить швидко розпочати роботу з інформаційною системою без складнощів, пов'язаних зі створенням та підтримкою складних інтерфейсів або мобільних додатків.

Отже, вибір телеграм-бота для реалізації інформаційної системи для аналізу та прогнозування курсу криптовалют може бути доцільним з точки зору зручності, доступності, безпеки та простоти використання для користувачів.

Метою роботи є створення інформаційної системи, яка забезпечить аналіз та прогнозування курсів криптовалют на основі різноманітних даних та аналітичних методів.

Методи дослідження включають аналіз ринкової динаміки криптовалют, застосування математичних та статистичних моделей для прогнозування курсів, а також використання інструментів машинного навчання для аналізу великих обсягів даних.

Результатом роботи буде створена інформаційної системи визначення та прогнозування курсів криптовалют. Система буде приймати історичні дані та продемонструє свою ефективність у реальних умовах фінансових ринків.

1 АНАЛІТИЧНИЙ ОГЛЯД

1.1 Еволюція ринку криптовалют

Початок ери криптовалют належить до 2009 року, коли була запущена перша криптовалюта - Bitcoin. Створення Bitcoin було відповіддю на бажання створити альтернативну фінансову систему, незалежну від урядового контролю та фінансових інституцій. Bitcoin відрізнявся від традиційних валют тим, що не мав централізованого органу управління, а замість цього базувався на технології блокчейну - децентралізованій системі, яка забезпечувала безпеку та прозорість транзакцій.

Початково Bitcoin не мав значної вартості, та з часом його ціна почала зростати, викликаючи зростання інтересу до цієї нової форми цифрових активів. Відкриття нових можливостей для інвестування та торгівлі криптовалютами привернуло увагу як індивідуальних інвесторів, так і великих фінансових установ.

У наступні роки після появи Bitcoin було створено безліч альтернативних криптовалют, які отримали назву "альткоїни" (альтернативні монети). Кожна з них мала свої унікальні характеристики та цілі, що сприяло розмаїтості на ринку криптовалют.

Пізніше, виникли такі інновації, як смарт-контракти, які дозволили використовувати блокчейн для виконання програмних кодів, а не лише для зберігання транзакцій. Це відкрило нові можливості для створення децентралізованих додатків та послуг.

Разом з розвитком технологій блокчейну з'явилися інші напрямки, такі як DeFi (децентралізовані фінанси), NFT (унікальні токени), та інші, які привертають увагу як інвесторів, так і творців.

Сьогодні ринок криптовалют став не лише новою формою інвестування, але й об'єктом дослідження, інновацій та розвитку. Він продовжує змінюватися та еволюціонувати, привертаючи увагу як індивідуальних інвесторів, так і

великих корпорацій та фінансових установ, що робить його одним з найцікавіших сегментів сучасного фінансового світу.

1.2 Технологічні засоби сучасних криптовалютних систем

Однією з ключових технологій, що лежить в основі сучасних криптовалютних систем, є блокчейн [1]. Блокчейн є децентралізованою базою даних, яка записує всі транзакції, зроблені з використанням певної криптовалюти. Кожен блок в ланцюжку містить криптографічний хеш попереднього блоку, що робить систему надзвичайно стійкою до маніпуляцій та шахрайства. Децентралізація означає, що кожен учасник мережі має копію блокчейну та може підтверджувати та проводити транзакції без посередництва централізованої установи.

Криптовалютні системи використовують криптографічні методи для забезпечення безпеки та анонімності транзакцій [2]. Кожна транзакція підписується унікальними цифровими ключами, які гарантують її автентичність та непідробленість. Більшість криптовалютних транзакцій є псевдоанонімними, оскільки вони не пов'язані з особистою інформацією користувачів, але вони все ще можуть бути відстежені в межах блокчейну.

Для підтвердження транзакцій та створення нових блоків у блокчейні використовуються різні консенсус-протоколи. Найпоширенішим з них є Proof of Work (PoW), який вимагає великих обчислювальних потужностей для розв'язання складних математичних завдань, і Proof of Stake (PoS), який базується на володінні криптовалютою та не вимагає таких великих енергетичних витрат.

Смарт-контракти - це програмні коди, що запускаються на блокчейні, які автоматично виконують угоди, коли виконуються певні умови. Вони дозволяють

створювати децентралізовані додатки та послуги, які не потребують централізованого управління.

Оскільки криптовалютні активи зберігаються у цифровому форматі, безпека є надзвичайно важливою. Криптовалютні гаманці використовують шифрування для захисту приватних ключів користувачів, а також багатошарові системи захисту, щоб запобігти несанкціонованому доступу до кошельків та транзакцій.

Технологічні засоби сучасних криптовалютних систем відіграють ключову роль у забезпеченні безпек.

1.3 Математичні моделі прогнозування курсу криптовалют

Для прогнозування курсів криптовалют використовуються різноманітні математичні моделі, які базуються на аналізі історичних даних та різноманітних факторів, що впливають на ціни. Однак, важливо пам'ятати, що жодна модель не може гарантувати 100% точність прогнозу, оскільки ринок криптовалют дуже вразливий до змін та маніпуляцій.

Random Forest (Випадковий ліс): Випадковий ліс [3] - це ансамбль дерев рішень, який використовується для регресії та класифікації. Він включає в себе кілька дерев рішень, кожне з яких навчається на підмножині даних та враховується при прийнятті рішення. Випадковий ліс може добре працювати з великою кількістю вхідних факторів та враховувати їх важливість у прогнозуванні.

Decision Tree (Дерево рішень): Дерево рішень [4]- це модель машинного навчання, яка вирішує проблеми регресії та класифікації шляхом розбиття вхідних даних на набір правил рішень. Вона використовує інформацію про вхідні фактори для побудови дерева, де кожен вузол відповідає певному правилу

розбиття. Дерева рішень можуть бути добрими варіантами для простих моделей з невеликою кількістю вхідних факторів.

Linear Regression (Лінійна регресія): Лінійна регресія [5]- це проста модель, яка використовує лінійну функцію для прогнозування значень залежної змінної на основі вхідних факторів. Вона шукає найкращий фітований пряму лінію, що найкращим чином відображає відношення між залежною та незалежними змінними.

Ridge (Гребенева регресія): Гребенева регресія [6]- це метод регуляризації лінійної регресії, який додає штраф за великі значення коефіцієнтів моделі. Це допомагає уникнути перенавчання та покращити узагальнення моделі.

Кожна з цих моделей має свої переваги та недоліки. Часто використовується ансамблевий підхід, коли результати кількох моделей об'єднуються для отримання кращого прогнозу.

Математичні моделі є основним інструментом у багатьох наукових дисциплінах, технічних галузях та інженерії. Вони використовуються для аналізу, прогнозування та розв'язання різноманітних проблем в різних областях, від фізики та економіки до біології та соціології. Дозволяють розуміти, як працює система, передбачати її поведінку та впливати на неї шляхом управління параметрами.

Основна ідея математичної моделі полягає в тому, що складний об'єкт або процес апроксимується більш простою системою математичних співвідношень. Це дозволяє представити реальну систему у вигляді формальної структури, яку можна аналізувати та використовувати для прогнозування.

Основні причини використання математичних моделей:

Розуміння системи: Моделі допомагають розкрити принципи роботи складних систем і виявити ключові фактори, що впливають на їхню поведінку.

Прогнозування: Моделі можуть використовуватися для передбачення майбутнього стану системи або результатів певних подій.

Оптимізація та управління: Вони дозволяють визначити оптимальні параметри системи для досягнення певної мети або заданого стану.

Експериментування: Моделі можуть служити заміною або доповненням фізичних експериментів, що дозволяє випробовувати різні сценарії без реальних витрат часу та ресурсів.

Навчання: Математичні моделі є важливим інструментом у навчанні, допомагаючи студентам розуміти та аналізувати складні системи.

У цілому, математичні моделі допомагають вирішувати реальні проблеми, спрощуючи їхнє розуміння, прогнозування та управління. Вони є важливим інструментом для наукових досліджень, інженерних розрахунків, прийняття рішень та вирішення різноманітних завдань у різних галузях.

1.4 Використання машинного навчання

Машинне навчання стає все більш популярним методом для прогнозування курсів криптовалют [7]. Від класичних алгоритмів до складних нейронних мереж, ці методи дозволяють ефективно аналізувати великі обсяги даних та виявляти складні зв'язки між різними факторами, що впливають на курси криптовалют.

Застосування машинного навчання в цій галузі відкриває широкі можливості для інвесторів та трейдерів. Моделі можуть аналізувати не лише історичні дані про курси криптовалют, але й враховувати велику кількість різноманітних факторів, таких як новини, соціальні мережі, технічні показники та інші. Це дозволяє створювати більш точні та прогностичні моделі, які можуть допомогти в прийнятті обґрунтованих рішень на ринку криптовалют.

Використання машинного навчання також дозволяє виявляти складні зв'язки між різними факторами та прогнозувати майбутні тренди на ринку. Наприклад, за допомогою алгоритмів класифікації можна виявити певні патерни поведінки курсів криптовалют та реагувати на них заздалегідь.

Однією з ключових переваг машинного навчання є його здатність до автоматизації та постійного вдосконалення. Моделі можуть навчатися на нових

даних та адаптуватися до змін на ринку, що дозволяє тримати прогнози актуальними та надійними в умовах постійної зміни ситуації.

Загалом, використання машинного навчання для прогнозування курсів криптовалют відкриває широкі перспективи для інвесторів та трейдерів, допомагаючи їм приймати обґрунтовані рішення та знижувати ризики на фінансових ринках.

1.5 Постановка задачі

Метою даного проекту є створення автоматизованої інформаційної системи для визначення та прогнозування курсу криптовалют, що базується на використанні алгоритмів машинного навчання та аналізу історичних даних ринку [8].

Розроблена система повинна відповідати наступним вимогам:

– Збір та збереження історичних даних про курси криптовалют, включаючи цінові показники, обсяги торгів та інші фактори впливу.

– Розробка та навчання моделей машинного навчання для прогнозування майбутніх змін у курсах криптовалют на основі історичних даних та зовнішніх факторів.

– Візуалізація та аналіз прогнозів курсів криптовалют, що дозволяє інвесторам та трейдерам приймати обґрунтовані рішення на фінансових ринках.

– Можливість автоматичного оновлення та вдосконалення моделей прогнозування з використанням нових даних та технік машинного навчання.

Для досягнення поставлених цілей необхідно вирішити наступні завдання:

– Провести аналіз сучасних підходів та алгоритмів машинного навчання для прогнозування фінансових ринків, зокрема курсів криптовалют.

– Порівняти різні моделі машинного навчання та їх можливості у прогнозуванні курсів криптовалют.

– Вибрати найбільш ефективні та точні алгоритми для розробки інформаційної системи.

– Реалізувати модель інформаційної системи для визначення та прогнозування курсів криптовалют на основі обраного алгоритму машинного навчання.

Предметом дослідження є методологія визначення та прогнозування курсу криптовалют за допомогою алгоритмів машинного навчання.

Наукова новизна полягає в можливості використання алгоритмів машинного навчання для прогнозування курсів криптовалют та аналізу їхніх змін.

Практичне значення полягає в підвищенні ефективності та точності прогнозів курсів криптовалют за допомогою автоматизованої інформаційної системи, що базується на алгоритмах машинного навчання.

2 ВИБІР МЕТОДІВ РОЗВ’ЯЗАННЯ ЗАДАЧІ

2.1 Огляд мови програмування Python

Вибір мови програмування для реалізації інформаційної системи визначення та прогнозування курсу криптовалют є критичним етапом, оскільки це визначає ефективність, швидкість та зручність розробки, а також масштабованість та підтримку системи в майбутньому. У цьому контексті Python виявляється однією з найбільш перспективних мов програмування [9].

Порівняємо написання телеграм-ботів на PHP, Java, Python та C++ з точки зору плюсів та мінусів:

Python:

Плюси:

Простота та зручність: Python має простий та зрозумілий синтаксис, що робить розробку та обслуговування коду більш простими.

Багата екосистема: Python має велику кількість бібліотек та фреймворків для розробки телеграм-ботів, таких як aiogram та python-telegram-bot.

Підтримка асинхронності: Python має потужну підтримку асинхронного програмування, що дозволяє обробляти багато запитів одночасно та зменшує навантаження на сервер.

Активна спільнота: Python має велику та активну спільноту розробників, яка постійно розвивається та надає підтримку.

Мінуси:

Швидкодія: У порівнянні з іншими мовами, такими як C++ або Java, Python може бути повільнішим у виконанні завдань, особливо у випадку обробки великих обсягів даних або інтенсивних обчислень.

Java:

Плюси:

Кросплатформенність: Java є кросплатформеною мовою програмування, що дозволяє розробляти боти, які працюють на різних операційних системах.

Широке застосування: Java використовується в багатьох великих проектах, тому вона може бути зручним вибором для комерційних та корпоративних застосувань.

Мінуси:

Складність: Java має складну синтаксичну структуру порівняно з Python, що може зробити процес розробки більш складним та тривалим.

Великий обсяг коду: Для написання простих телеграм-ботів у Java може знадобитися більше рядків коду, ніж у Python, що може призвести до більшого обсягу програмного коду.

PHP:

Плюси:

Швидкодія: PHP є швидкою мовою програмування, що може бути важливим фактором для веб-застосунків та сервісів з великою кількістю користувачів.

Велика спільнота: PHP має велику та активну спільноту розробників, яка надає багато корисних ресурсів та підтримку.

Мінуси:

Неструктурованість: У порівнянні з Python або Java, PHP може мати менш структурований та менш чіткий код, що може призвести до складнощів у розумінні та підтримці коду.

Обмежена функціональність: PHP не має такого розгалуженого екосистеми бібліотек та фреймворків для розробки телеграм-ботів, як Python.

C++:

Плюси:

Швидкодія: C++ є однією з найшвидших мов програмування, що дозволяє створювати високоефективні та швидкі боти.

Близькість до металу: C++ надає прямий доступ до пам'яті та апаратного забезпечення, що дозволяє розробникам максимально оптимізувати та керувати ресурсами.

Мінуси:

Складність: C++ вважається складною мовою програмування з високим рівнем складності та вимогами до розробника.

Обмеженість інструментів: C++ може мати обмежений набір бібліотек та інструментів для розробки телеграм-ботів порівняно з Python або Java.

Порівняно з альтернативами, такими як Java, PHP, або C++, Python вирізняється простотою використання, швидкістю розробки та великим спектром доступних бібліотек для аналізу даних та розвитку машинного навчання. Він дозволяє зосередитися на аналізі даних та розробці прогностичних моделей, замість того, щоб витрачати час на вирішення технічних проблем, що можуть виникнути при використанні інших мов програмування.

2.2 Огляд бібліотек Pandas, Aiogram, NumPy, Ta , Telebot, Scikit-learn

При розробці інформаційної системи були використані бібліотеки, назви яких представлені на рис.2.2.1:

```
import os
import ccxt
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression, Ridge
from ta.trend import MACD
from sklearn.pipeline import make_pipeline
from sklearn.compose import make_column_transformer
from aiogram import Bot, Dispatcher, types
from aiogram.contrib.fsm_storage.memory import MemoryStorage
from aiogram.dispatcher import FSMContext
import asyncio
from aiogram.types import ParseMode
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
```

Рисунок 2.2.1 – Використані бібліотеки для інформаційної системи

Pandas (pd): Pandas [10] - це основна бібліотека для обробки та аналізу даних у форматі таблиць. З його допомогою можна легко завантажувати, обробляти та маніпулювати фінансовими даними. Pandas надає зручний інтерфейс для вирішення багатьох завдань, таких як фільтрація, сортування, об'єднання таблиць тощо.

NumPy (np): NumPy [11] - це бібліотека для наукових обчислень у Python. Вона надає векторизовані обчислення та підтримку для масивів та матриць, що робить її ідеальним інструментом для роботи з числовими даними, такими як ціни криптовалют.

Scikit-learn: Scikit-learn [12]- це бібліотека машинного навчання, яка надає реалізації багатьох алгоритмів класифікації, регресії та кластеризації. Для системи прогнозування курсу криптовалют можна використовувати моделі регресії, такі як RandomForestRegressor, DecisionTreeRegressor, LinearRegression, Ridge, для прогнозування цінових тенденцій на основі історичних даних.

ta (Technical Analysis Library) [13]: Бібліотека ta надає інструменти для технічного аналізу фінансових даних. Наприклад, вона включає показники, такі як MACD (Moving Average Convergence Divergence), який дозволяє визначити тенденції та сигнали купівлі/продажу на основі аналізу історичних цін.

Aiogram: Aiogram [14]- це бібліотека для створення ботів у месенджерах Telegram. Вона дозволяє легко реалізувати функціональність обміну даними та надання користувачам доступу до інформації про курс криптовалют через чат-бот.

Telebot: Telebot - це бібліотека для розробки телеграм-ботів на мові програмування Python. Вона надає простий та зручний інтерфейс для взаємодії з API Telegram та швидкої розробки різноманітних ботів.

asyncio: asyncio - це бібліотека для асинхронного програмування в Python. У контексті інформаційної системи для визначення та прогнозування курсу криптовалют, asyncio може бути використана для асинхронного взаємодії з

мережевими ресурсами, такими як API криптовалютних бірж, що дозволяє забезпечити швидке та ефективне отримання даних.

os: Бібліотека os надає інтерфейс до операційної системи, що дозволяє взаємодіяти з файловою системою, створювати, перейменовувати та видаляти файли та теки. У випадку системи для аналізу криптовалют, os може бути використана для управління локальними файлами, такими як збереження та завантаження даних.

datetime: Бібліотека datetime дозволяє працювати з датами та часом у Python. У контексті системи для визначення та прогнозування курсу криптовалют, datetime може бути використана для обробки дат в історичних даних, розрахунку та аналізу часових рядів цін на криптовалюту та для розпізнавання та форматування дат у повідомленнях чат-бота.

Кожна з цих бібліотек має свої переваги та недоліки, але разом вони утворюють потужний інструментарій для аналізу та прогнозування курсу криптовалют, що дозволить створити ефективну та надійну інформаційну систему для цієї цілі.

Порівняння між aiogram та telebot може бути корисним для вибору найбільш підходящого інструменту для розробки телеграм-бота. Ось детальний огляд обох бібліотек:

aiogram:

Асинхронне програмування: aiogram підтримує асинхронну модель програмування за допомогою бібліотеки asyncio. Це дозволяє реалізувати ефективний та швидкий бот, який може обробляти багато запитів одночасно.

Багатофункціональність: aiogram надає широкий спектр можливостей для розробки телеграм-ботів, включаючи роботу з повідомленнями, клавіатурами, файлами, медіа та іншими елементами.

Документація та спільнота: aiogram має добре написану документацію та активну спільноту розробників, що дозволяє швидко знайти відповіді на питання та вирішити проблеми.

telebot:

Простота використання: telebot відомий своєю простотою використання та зрозумілістю інтерфейсу. Він підходить для початківців та швидко дозволяє створювати базові боти.

Швидкість розробки: telebot може бути швидким інструментом для створення простих та базових телеграм-ботів, оскільки він має простий API та не вимагає глибоких знань програмування.

Менший обсяг коду: telebot може забезпечити можливість швидко написати нескладний бот з меншою кількістю коду, що може бути важливим для невеликих проектів.

Узагальнюючи, вибір між aiogram та telebot залежить від конкретних потреб та вимог проекту. Я обрав aiogram для розробки телеграм-бота через його потужність, можливості асинхронного програмування та широкий функціонал, який відповідає моїм потребам.

2.3 Огляд математичних моделей Random Forest, Decision Tree, Linear Regression, Ridge Regression

Розглянемо кожну з використаних моделей [4], їх особливості, плюси та мінуси (рис. 2.3.1):

```
models = [
    ('Random Forest', make_pipeline(column_transformer, RandomForestRegressor())),
    ('Decision Tree', make_pipeline(column_transformer, DecisionTreeRegressor())),
    ('Linear Regression', make_pipeline(column_transformer, LinearRegression())),
    ('Ridge', make_pipeline(column_transformer, Ridge()))
]
```

Рисунок 2.3.1 – Математичні моделі для прогнозування

Random Forest (Випадковий ліс) - це ансамбль деревних моделей, що використовуються як для класифікації, так і для регресії. Кожне дерево

випадкового лісу обчислює прогноз окремо, а потім результати комбінуються для отримання кінцевого результату.

Плюси:

– Ефективний для роботи з великими наборами даних та високоузагальненими моделями.

– Можливість оцінки важливості ознак, що дозволяє здійснювати відбір ознак.

– Здатність автоматично обробляти відсутні дані.

Мінуси:

– Може бути схильний до перенавчання, особливо при наявності великої кількості дерев.

– Може бути менш інтерпретований порівняно з іншими моделями, такими як лінійна регресія.

– Може бути часоємним при навчанні на великих наборах даних.

Decision Tree (Дерево рішень) - це деревоподібна структура, що складається з умовних вузлів і листків. Кожен вузол представляє собою тест на ознаку, кожне ребро відповідає результату тесту, а кожен лист - класу або значенню. У випадку регресії, листям може бути прогнозоване значення.

Плюси:

– Легко інтерпретується і пояснюється.

– Може працювати з якісними та кількісними даними.

– Відсутність потреби у нормалізації даних.

Мінуси:

– Схильний до перенавчання, особливо коли дерево занадто глибоке.

– Не так ефективний, як інші моделі, такі як випадковий ліс.

– Не дуже стійкий до зміни в даних.

Linear Regression (Лінійна регресія) - це модель, яка шукає лінійну залежність між вхідними змінними та вихідною змінною. Модель припускає, що залежність є лінійною та усі ознаки є незалежними.

Плюси:

– Простота інтерпретації результатів.
– Швидкість навчання та передбачення на великих наборах даних.
– Ефективність для лінійно залежних даних.

Мінуси:

– Обмежена здатність моделі враховувати нелінійні залежності між змінними.
– Чутливість до викидів.
– Можливість недооцінки складності відносин між ознаками та вихідним значенням.

Ridge Regression (Регресія Ріджа) - це метод регресії, який додає штраф до суми квадратів коефіцієнтів моделі (L2-регуляризація), що допомагає уникнути перенавчання.

Плюси:

– Зменшує ефект перенавчання шляхом регуляризації.
– Працює добре в умовах мультиколінеарності (коли ознаки досить сильно корелюють між собою).
– Ефективний для використання з великою кількістю ознак.

Мінуси:

– Потребує правильного підбору параметра регуляризації.
– Може бути менш інтерпретованим порівняно з простішими моделями, такими як лінійна регресія.
– Зменшується точність прогнозування порівняно з нерегуляризованими моделями у випадку, коли немає перенавчання.
– Кожна з цих моделей має свої переваги та недоліки, і вибір конкретної моделі залежить від конкретного завдання, даних та потреб користувача.

2.4 Огляд баз даних SQLite, MySQL та PostgreSQL

Бази даних є невід'ємною частиною багатьох програм та систем, надаючи зручний та ефективний спосіб зберігання та організації даних. Обираючи базу даних для конкретного проекту, важливо враховувати потреби та вимоги самого проекту, а також рівень складності та масштабування системи.

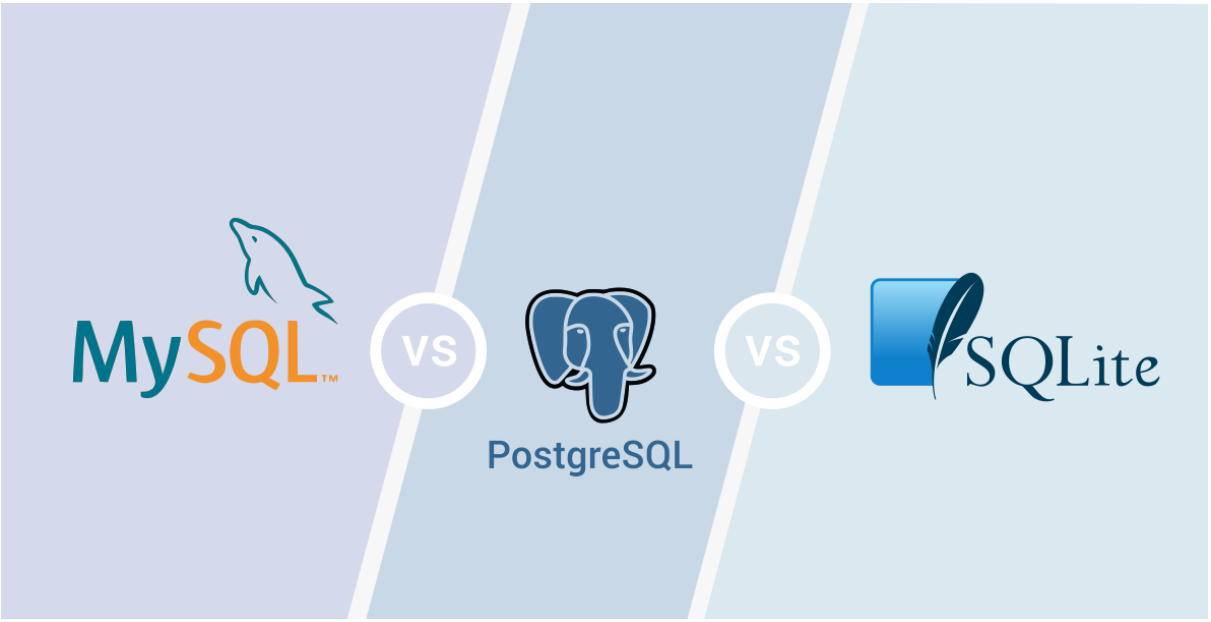

Рисунок 2.4.1 – СУБД MySQL, PostgreSQL, SQLite

SQLite, MySQL та PostgreSQL (рис. 2.4.1) - це різні системи управління базами даних зі своїми властивостями, перевагами та недоліками. Розглянемо їх порівняльні характеристики та обгрунтуємо вибір SQLite:

SQLite:

Плюси:

Простота використання: SQLite [15]- це легка та проста у використанні база даних, яка не потребує окремого сервера та налаштувань.

Низькі вимоги до ресурсів: SQLite використовує мінімальні ресурси, що робить її ідеальним вибором для вбудованих систем та додатків з обмеженими ресурсами.

Підтримка ACID-властивостей: SQLite підтримує транзакції з властивостями ACID (атомарність, послідовність, ізоляція, стійкість).

Широке розповсюдження: SQLite використовується великою кількістю додатків та підтримується на багатьох платформах.

Мінуси:

Обмеженість функцій: SQLite може бути обмеженою для деяких завдань, особливо коли потрібні розширені можливості обробки даних або масштабування.

Одночасність: SQLite може мати проблеми з одночасним доступом до бази даних в багатопроцесорних середовищах порівняно з MySQL або PostgreSQL.

MySQL:

Плюси:

Широка підтримка: MySQL має велику та активну спільноту розробників та користувачів, що надає багато ресурсів та підтримку.

Висока продуктивність: MySQL є швидкою та ефективною системою управління базами даних, що може оптимізувати роботу з великим обсягом даних.

Масштабованість: MySQL має можливості для масштабування, що дозволяє розширювати його функціональність для великих та складних проектів.

Мінуси:

Вимоги до ресурсів: MySQL може вимагати більше ресурсів порівняно з SQLite, що робить його менш підходящим для вбудованих систем та додатків з обмеженими ресурсами.

Складність налаштування: Налаштування та керування MySQL може вимагати більшого рівня експертизи та знань у порівнянні з SQLite.

PostgreSQL:

Плюси:

Розширені можливості: PostgreSQL має багатий набір функцій та розширень, що робить його відмінним вибором для складних аналітичних та даних-орієнтованих застосунків.

Підтримка ACID-властивостей: PostgreSQL підтримує транзакції з властивостями ACID, що робить його надійним та стійким.

Мінуси:

Високі вимоги до ресурсів: PostgreSQL може бути вимогливим до ресурсів, особливо при роботі з великим обсягом даних.

Складність конфігурації: PostgreSQL може мати складну структуру конфігурації та налаштувань, що може потребувати додаткового часу та експертизи для належного налаштування та оптимізації.

Отже, у контексті інформаційної системи для визначення та прогнозування курсу криптовалют, SQLite може бути оптимальним вибором через свою легкість використання, ефективність, вбудований механізм та портативність. Вона надає надійний та ефективний спосіб зберігання та організації даних, що важливо для успішної реалізації проекту.

3 ІНФОРМАЦІЙНА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Розробка інформаційної системи

Під час розробки інформаційної системи для автоматизованого аналізу та прогнозування цін на криптовалюти було використано історичні дані криптовалют.

Основною метою було створення ефективної системи, яка забезпечить точний і швидкий аналіз та прогнозування цін на криптовалюти на основі історичних даних, використовуючи алгоритми та телеграм-бот для автоматизованого виконання завдань.

Функціональні можливості системи:

Система надає користувачам такі ключові функції:

– Телеграм-бот інтерфейс: Користувачі можуть взаємодіяти з ботом через телеграм, використовуючи команди та кнопки для отримання прогнозів та іншої інформації.

– Вибір криптовалюти: Користувачі можуть вибирати криптовалюту для аналізу, ввівши її символ через інтерфейс телеграм-бота.

– Завантаження історичних даних: Бот завантажує історичні дані про ціни криптовалют з біржі Binance за допомогою API. Дані включають ціну відкриття, максимум, мінімум, закриття і обсяги торгівлі.

– Обробка даних: Історичні дані очищуються та обробляються для подальшого аналізу, включаючи розрахунок індикаторів технічного аналізу, таких як RSI, ковзне середнє і MACD.

– Моделювання та прогнозування: Система використовує кілька моделей машинного навчання, таких як випадковий ліс, дерево рішень, лінійна регресія і гребенева регресія, для навчання на історичних даних і прогнозування цін на наступні 30 днів.

– Візуалізація прогнозів: Прогнози моделей візуалізуються у вигляді графіків, які показують прогнозовані ціни на наступні 30 днів, і надсилаються користувачам у телеграм.

– Збереження та передача даних: Прогнози зберігаються у CSV-файлах, які також надсилаються користувачам. Після цього файли видаляються для оптимізації використання простору.

– Інтерактивні повідомлення: Бот надсилає користувачам інтерактивні повідомлення та інструкції, що дозволяє легко та швидко отримати необхідну інформацію і прогнози.

Основні компоненти системи:

– Імпорт та обробка даних: Використання бібліотек ccxt для завантаження даних з Binance, pandas для обробки даних та бібліотеки ta для розрахунку технічних індикаторів.

– Моделювання та прогнозування: Використання бібліотек scikit-learn для створення моделей машинного навчання та виконання прогнозів.

– Інтерфейс телеграм-бота: Використання бібліотеки aiogram для створення телеграм-бота, обробки команд користувачів та управління станами.

– Візуалізація та збереження даних: Використання matplotlib для створення графіків прогнозів і збереження їх у вигляді зображень та CSV-файлів.

Ця система забезпечує зручний і ефективний спосіб аналізу та прогнозування цін на криптовалюти, використовуючи сучасні технології машинного навчання та інтерфейс телеграм-бота.

Розробка USE-CASE Diagram.

Для розуміння взаємодії користувача з системою було розроблено USE-CASE діаграму (рис. 3.1.1). Діаграма представляє собою взаємодію телеграм-бота між користувачем та системою.

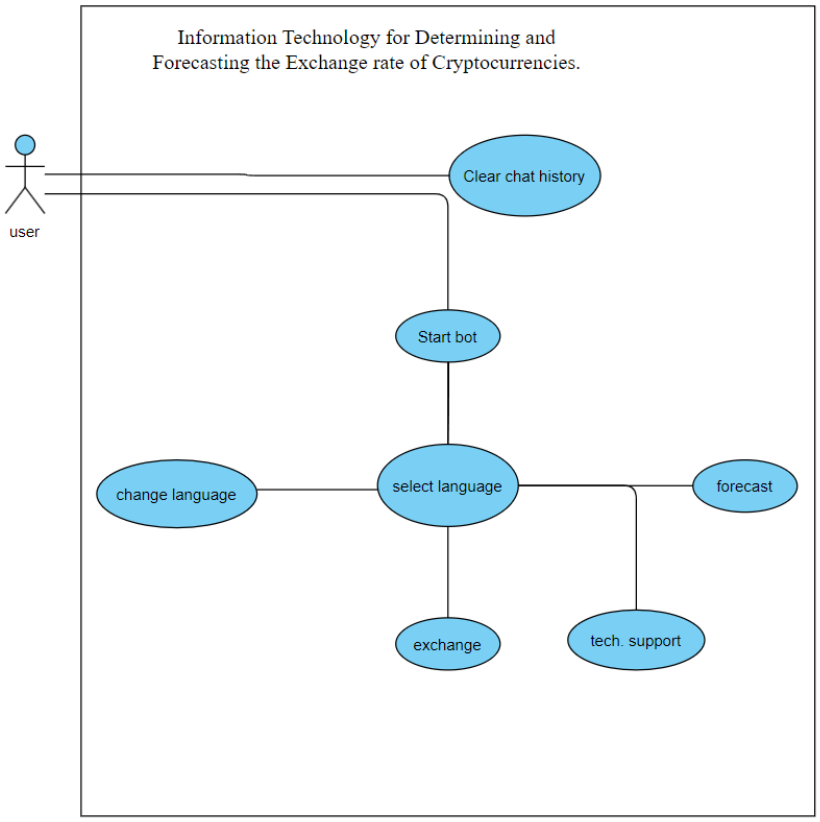

Рисунок 3.1.1 – USE-CASE Діаграма

3.2 Розробка та імплементація

При розробці інформаційної системи головний аспект було приділено інтуїтивно зручному інтерфейсу користування для користувача.

Також для зручності було розбито код на 7 модулів.

Основний модуль bot.py (рис.3.2.1):

Імпорт бібліотек: Спочатку відбувається імпорт необхідних бібліотек (рис. 2.2.1) для роботи з криптовалютними даними (ccxt), обробки даних (pandas, numpy), машинного навчання (scikit-learn), а також бібліотек для роботи з Telegram API (aiogram).

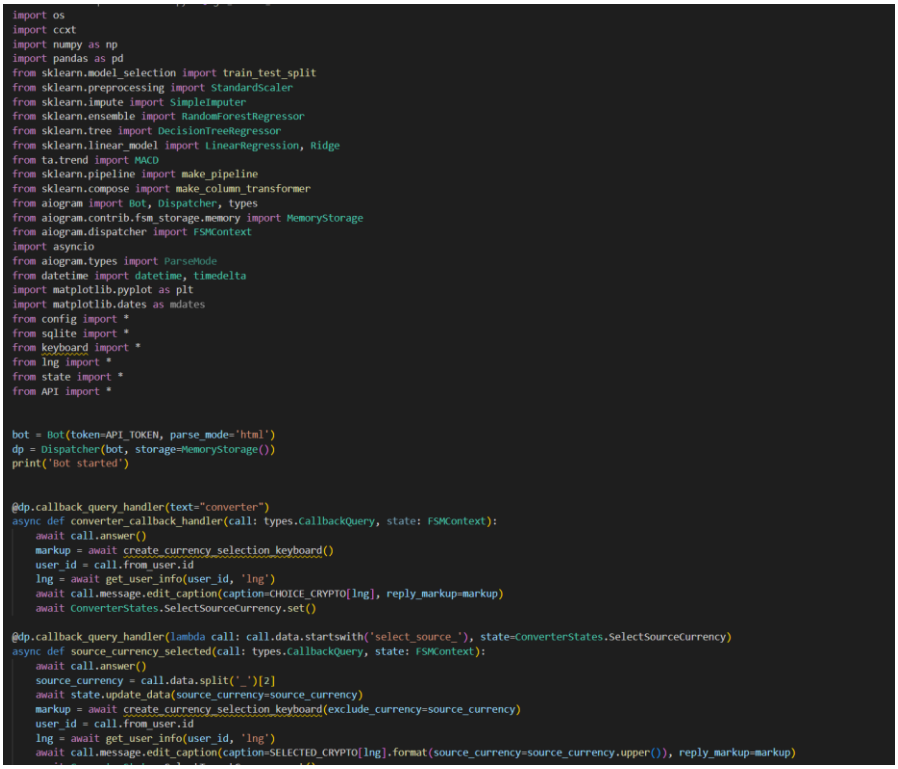

Рисунок 3.2.1 - Основний модуль bot.py

Ініціалізація бота та диспетчера: Створюється об'єкт бота та диспетчера для обробки повідомлень та подій в Telegram (рис 3.2.2).

Рисунок 3.2.2 - Ініціалізація бота та диспетчера

Обробники повідомлень та подій: код містить різні обробники, які реагують на події, такі як натискання кнопок, введення тексту користувачем тощо. Наприклад, є обробник для вибору криптовалюти, обробник для введення користувачем суми для конвертації, обробник для прогнозування ціни криптовалюти тощо.

Загалом, цей код використовується для створення телеграм-бота, а також для підключення інших модулів.

Наступний модуль API.py (рис.3.2.3):

Ця частина коду відповідає за отримання ціни криптовалюти з API CoinGecko. Давайте розглянемо кожну частину:

Імпорт бібліотек: Імпортується бібліотека requests для виконання HTTP-запитів та клас Decimal з бібліотеки decimal для роботи з точністю чисел з плаваючою комою.

Змінна API_URL: Визначається URL API для взаємодії з CoinGecko.

Функція get_crypto_price: Ця функція приймає аргумент coin, що є символом криптовалюти (наприклад, "bitcoin", "ethereum" тощо). Функція складає URL для запиту до API CoinGecko для отримання ціни криптовалюти в доларах США. Після виконання запиту та отримання відповіді в JSON-форматі, вона витягує ціну криптовалюти з відповіді та повертає її у вигляді об'єкту Decimal [16].

Обробка помилок: Якщо отримання ціни криптовалюти завершилося невдачею або ціна не була знайдена, функція піднімає виняток з повідомленням про помилку.

Ця функція є важливою для отримання актуальних цін на криптовалюти з CoinGecko та використовується в коді для обчислення конвертацій та прогнозування цін.

```python
import requests
from decimal import Decimal
API_URL = "https://api.coingecko.com/api/v3"

async def get_crypto_price(coin):
    url = f"{API_URL}/simple/price?ids={coin}&vs_currencies=usd"
    response = requests.get(url)
    data = response.json()
    price = data.get(coin, {}).get('usd')
    if price is None:
        raise ValueError(f"Error symbol {coin}")
    return Decimal(str(price))
```

Рисунок 3.2.3 - Отримання ціни криптовалюти

Наступний модуль config.py (рис. 3.2.4):

Цей модуль відповідає за основні налаштування інформаційної системи, тобто шлях до бази даних, токен телеграм-бота і т.п..

```
API_TOKEN = '1710347244:AAGcas3fih2vkPWpNLt45nT3eJ7zX0Cf7LI'
DB_PATH='C:/Users/Yeshenko/Desktop/bot/main.db'
SUPPORT_URL = 'https://t.me/impu1se228'
SUPPORT_USERNAME = '@impu1se228'
PHOTO_ID='AgACAgIAAxkDAAIJ22Zaa1vVX_M7Pylv92FAppVZctM0AAJl0zEboSC4SsbYcLrtDsRqAQADAgADeAADNQQ'
```

Рисунок 3.2.4 – Налаштування конфігурацій бота

Модуль keyboard.py (рис.3.2.5):

Цей модуль відповідає за клавіатуру інтерфейсу бота, який допомогає користувачу виконувати різні дії при користуванні інформаційної системи, тобто зміна мови, вибір прогнозування і т.п..

```
from aiogram import types
from lng import *
from config import *
from sqlite import *


back_btn_text = '⬅️'
back_to_main_page = types.InlineKeyboardButton(text=back_btn_text, callback_data='main_page')

def choice_lng_keyboard(back_btn_status=False):
    markup = types.InlineKeyboardMarkup(row_width=1).add(
    types.InlineKeyboardButton(text='UA Український', callback_data='choice_lng;uk'),
    types.InlineKeyboardButton(text='GB English', callback_data='choice_lng;eng'),
    types.InlineKeyboardButton(text='PL Polski', callback_data='choice_lng;pl'),
    )
    if back_btn_status:
        markup.add(back_to_main_page)
    return markup

def main_navigation_keyboard(lng):
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    markup.add(PROFILE_BUTTON_TEXT[lng])
    return markup

def main_keyboard(lng):
    markup = types.InlineKeyboardMarkup(row_width=2)
    markup.add(types.InlineKeyboardButton(text=CONVERTER[lng], callback_data='converter'))
    markup.add(types.InlineKeyboardButton(text=FORECAST[lng], callback_data='forecast'))
    markup.add(
        types.InlineKeyboardButton(text=CHANGE_LNG_BTN[lng], callback_data='change_lng'),
        types.InlineKeyboardButton(text=SUPPORT_BTN[lng], url=SUPPORT_URL),
        )
    return markup


async def create_currency_selection_keyboard(exclude_currency=None):
    keyboard = types.InlineKeyboardMarkup()
    currencies = ['bitcoin', 'ethereum', 'litecoin', 'usd']
    for currency in currencies:
        if currency != exclude_currency:
            callback_data = f'select_source_{currency}'
            button = types.InlineKeyboardButton(text=currency.upper(), callback_data=callback_data)
            keyboard.add(button)
    keyboard.add(back_to_main_page)
    return keyboard
```

Рисунок 3.2.5 – Код для клавіатури бота

Наступний модуль lng.py (рис. 3.2.6):

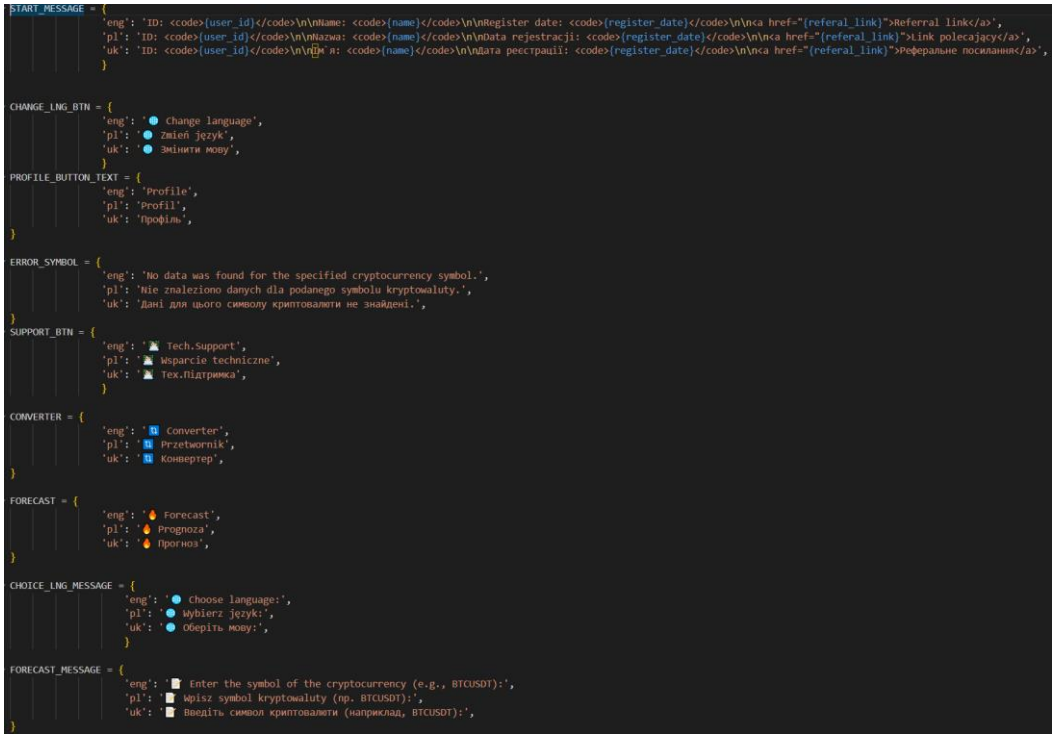

Рисунок 3.2.6 – Локалізація боту

Цей блок коду містить різні переклад мов для більш зручного користування ботом, даний бот має 3 мови для вибору , а саме:

- англійська
- українська
- польська

Далі модуль sqlite.py (рис. 3.2.7):

Модуль sqlite.py відповідає за SQL-запити до СУБД sqlite, використовуючи бібліотеку aiosqlite для асинхроних запитів до бази даних. Даний код виконує реєстрацію нового користувача у системі, записуючи дату реєстрації, ID-користувача, та зберігає мову, яку буде використовувати користувач.

Рисунок 3.2.7 – Код SQL-запитів

Останній модуль state.py (рис. 3.2.8):

Модуль використовується для визначення станів (states) бота у фреймворку aiogram.

Фреймворк aiogram дозволяє використовувати стани для керування логікою роботи бота. Стани визначають структуру взаємодії з користувачем і вказують, в якому стані знаходиться користувач у конкретний момент часу. Кожен стан може мати свої власні дії, які виконуються при переході у цей стан або при виконанні певних подій.

У коді клас ConverterStates визначає стани, які використовуються для перетворення валют. Він має три стани:

SelectSourceCurrency: стан, в якому користувач обирає вихідну валюту для конвертації.

SelectTargetCurrency: стан, в якому користувач обирає цільову валюту для конвертації.

EnterAmount: стан, в якому користувач вводить суму для конвертації.

Клас CryptoState також визначає стани, але вже для іншої частини логіки бота, пов'язаної з криптовалютами. У даному випадку цей клас має лише один стан - WaitingForSymbol, який вказує на очікування введення користувачем символу криптовалюти.

Використання станів дозволяє боту ефективно керувати взаємодією з користувачем і виконувати потрібні дії в залежності від поточного контексту.

```python
from aiogram.dispatcher.filters.state import State, StatesGroup

class ConverterStates(StatesGroup):
    SelectSourceCurrency = State()
    SelectTargetCurrency = State()
    EnterAmount = State()

# Define states
class CryptoState(StatesGroup):
    WaitingForSymbol = State()
```

Рисунок 3.2.8 – Використання станів для вибору дій

3.3 Тестування

Після запуску телеграм-бота, ви отримаєте повідомлення про вибір мови для користування (рис. 3.3.1).

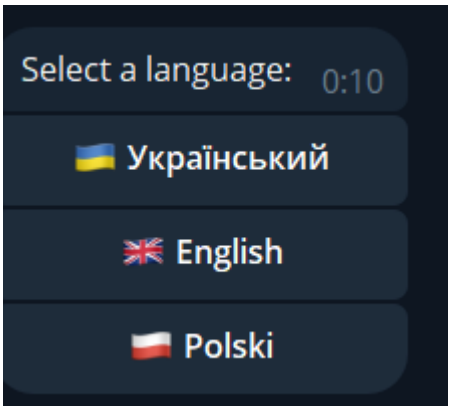

Рисунок 3.3.1 – Вибір мови при запуску бота

Далі у нас з’виться основне меню користування ботом (рис. 3.3.2), де можна вибрати з меню конвертацію, прогнозування , зміну мови, тех.підтримку:

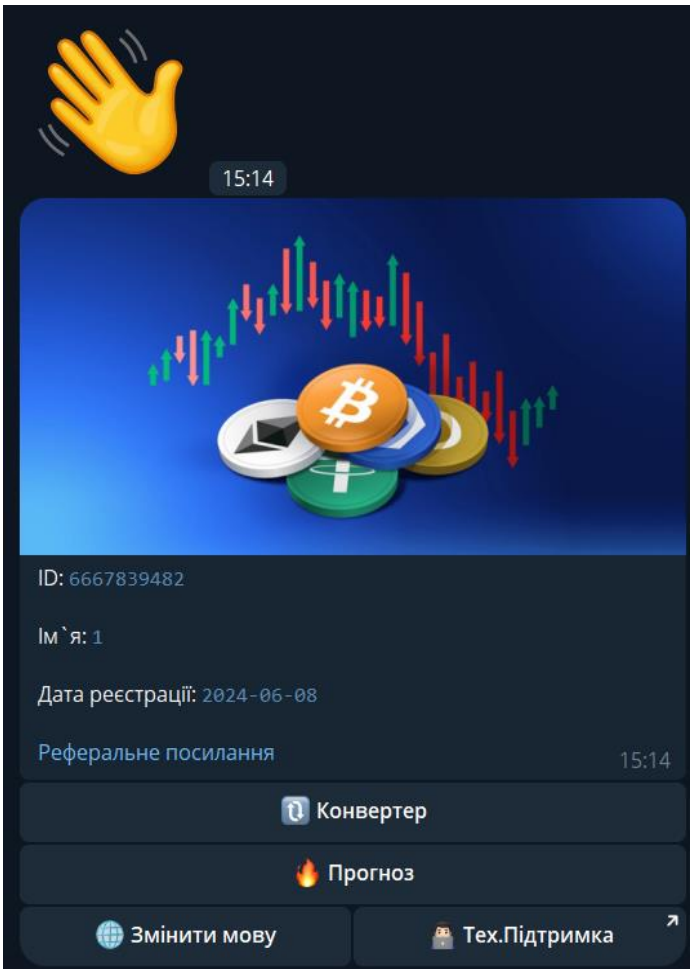

Рисунок 3.3.2 – Основне меню бота

Для конвертації валют потрібно вибрати, пару валют, які будемо конвертувати, наприклад BITCOIN – USD (рис.3.3.3).

Рисунок 3.3.3 – Конвертування валют

Далі нам потрібно ввести число, для конвертування, наприклад 1. Тоді ми отримаємо результат конвертування з BITCOIN у USD (рис.3.3.4).

Рисунок 3.3.4 – Результат конвертування валют

Наступним етапом протестуємо роботу прогнозування на 30 днів, обравши прогнозування , потрібно ввести пару валют , як зазначено у прикладі (рис. 3.3.5).

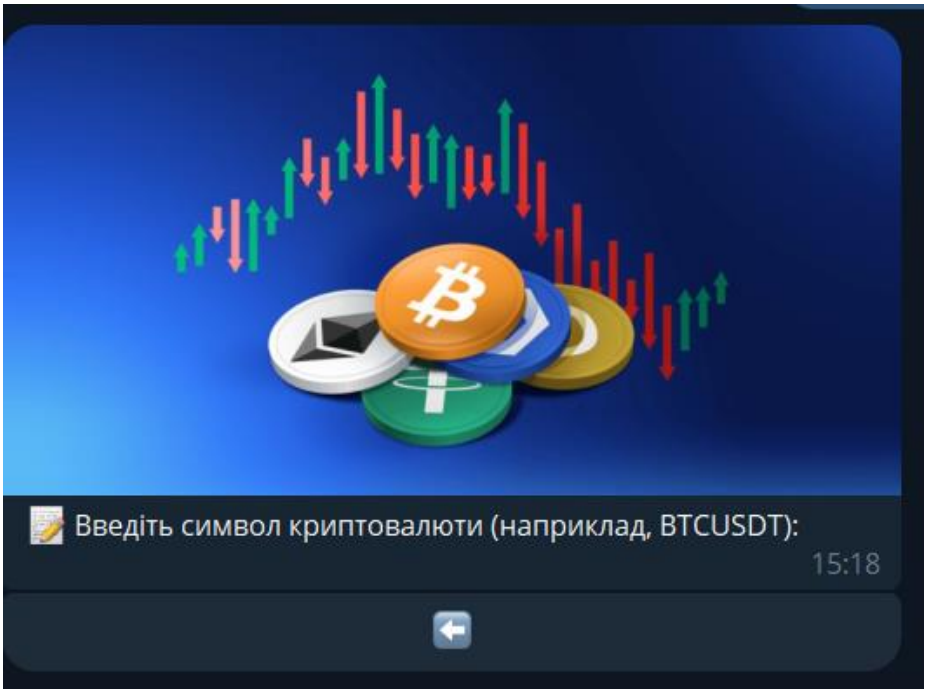

Рисунок 3.3.5 – Прогнозування валют

Введемо значення ethusdt, та отримаємо відповідь від бота (рис.3.3.6).

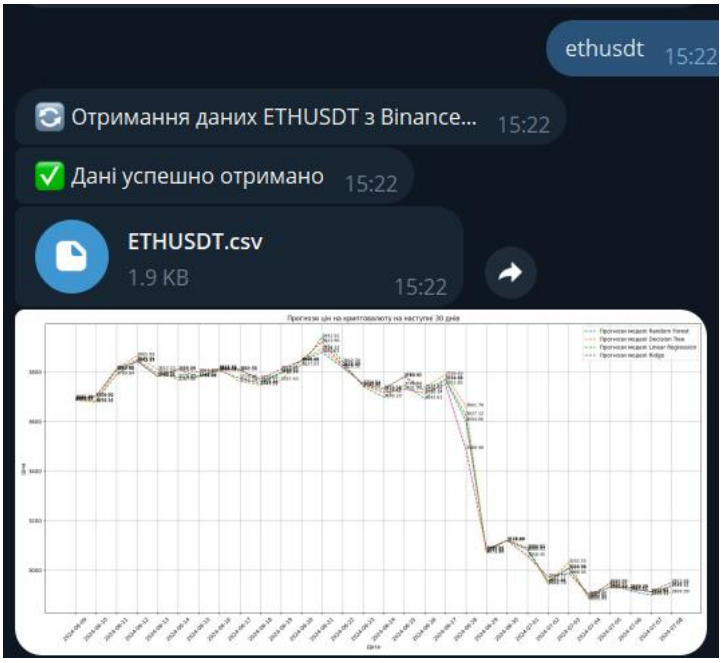

Рисунок 3.3.6 – Успішний результат прогнозування

На зображеному рисунку (рис. 3.3.7) , можна побачити результат чотирьох математичних моделей для прогнозування для пари валют.

Рисунок 3.3.7 – Прогноз усіх чотирьох моделей

Також можна побачити логування по кожній моделі і її точність (рис. 3.3.8).

Ці логи вказують на оцінку точності моделей регресії, яка вимірюється за допомогою коефіцієнта детермінації (R-squared). R-squared вказує на те, яка частина варіації в цільовій змінній (у даному випадку ціну криптовалюти) пояснюється використаною моделлю. Чим ближче значення R-squared до 1, тим краще модель пояснює варіацію цільової змінної.

Отримані значення (рис.3.3.8) R-squared для моделей Random Forest (99.64%), Decision Tree (99.55%), Linear Regression (99.83%) і Ridge (99.83%) говорять про те, що ці моделі дуже добре відповідають навчальним даним і можуть точно передбачати ціну криптовалюти на основі вхідних факторів. Значення близькі до 100% свідчать про високу точність моделей.

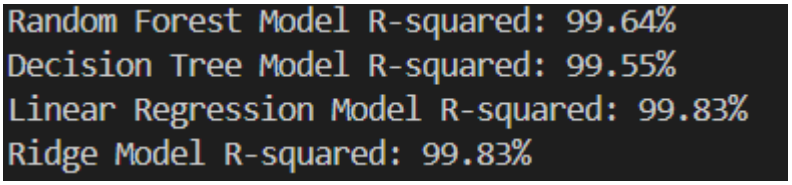

Рисунок 3.3.8 – Логування точності кожної мат. моделі

Точність моделі регресії залежить від різних факторів, включаючи:

– Якість даних: Якість та чистота вхідних даних може суттєво вплинути на точність моделі. Неправильні або відсутні дані можуть призвести до неточних результатів.

– Відповідність моделі: Вибір правильної моделі для конкретного типу даних та завдання може визначити точність передбачень. Деякі моделі можуть краще підходити для певних видів даних або структур.

– Параметри моделі: Налаштування параметрів моделі може покращити її точність. Наприклад, зміна параметрів моделі машинного навчання, таких як глибина дерева рішень у моделі випадкового лісу, може вплинути на її здатність до адаптації до даних.

– Розмір навчального набору: Більші обсяги даних можуть допомогти моделі краще узагальнювати та точніше передбачати нові дані.

– Види функцій та їх обробка: Відповідний вибір функцій для моделі може вплинути на її здатність до знаходження взаємозв'язків у даних. Обробка функцій, така як масштабування, нормалізація або вилучення непотрібних функцій, також може вплинути на точність моделі.

– Крос-валідація: Використання крос-валідації дозволяє оцінити точність моделі на незалежних даних та уникнути перенавчання (overfitting) або недонавчання (underfitting).

– Гіперпараметри: Налаштування гіперпараметрів моделі також може вплинути на її точність. Гіперпараметри - це параметри моделі, які не вивчаються під час навчання і повинні налаштовуватися заздалегідь, такі як швидкість навчання (learning rate) у градієнтному спуску.

Ці фактори можуть взаємодіяти між собою, для досягнення високої точності моделі часто вимагає експериментування з різними підходами та налаштуваннями.

ВИСНОВКИ

Під час написання кваліфікаційної роботи було проведено аналіз та створено інформаційну систему для визначення та прогнозування курсу криптовалют. Визначено актуальність використання таких систем у фінансовій сфері та інвестиційному середовищі.

Під час виконання роботи були вирішені наступні завдання:

1. Проведено аналіз ринку криптовалют та вимог користувачів щодо системи визначення та прогнозування їхнього курсу.
2. Досліджено сучасні підходи та технології у галузі фінансових аналітичних систем та обробки даних.
3. Розроблено алгоритм прогнозування, інтеграцію зі сторонніми сервісами та інтерфейс користувача.
4. Виконано імплементацію інформаційної системи згідно з визначеною архітектурою та вимогами, використовуючи мову програмування Python та бібліотеки для аналізу даних.
5. Забезпечено тестування та оптимізацію розробленої системи з метою забезпечення найвищої точності та ефективності прогнозування курсу криптовалют.

Отже, розроблена інформаційна система для визначення та прогнозування курсу криптовалют виявилася ефективним інструментом для аналізу ринку та прийняття інвестиційних рішень. Вона дозволяє користувачам отримувати актуальну інформацію та проводити аналіз ринкових тенденцій з метою максимізації прибутку та зниження фінансових ризиків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кім, Г. А. Технології блокчейну та криптовалют : Підручник / Г. А. Кім, П. М. Воробей. – Київ : Наукова думка, 2020. – 315 с.
2. Попов, С. А. Основи криптографії та безпеки даних / С. А. Попов. – Харків : Фоліо, 2018. – 280 с.
3. Мельник, А. С. Випадкові ліси для машинного навчання : Підручник / А. С. Мельник. – Київ : Наукова думка, 2018. – 240 с.
4. Сидоренко, О. П. Дерева рішень: Теорія та застосування / О. П. Сидоренко. – Одеса : ОНУ, 2019. – 270 с.
5. Тимошенко, В. О. Лінійна регресія в прогнозуванні фінансових ринків / В. О. Тимошенко. – Харків : Харківський національний університет, 2020. – 250 с.
6. Петренко, Н. С. Гребенева регресія: Теорія та застосування / Н. С. Петренко. – Дніпро : Академія, 2019. – 240 с.
7. Ткаченко, В. М. Використання машинного навчання у фінансових системах / В. М. Ткаченко. – Харків : Харківський національний університет, 2021. – 290 с.
8. Коваленко, М. І. Машинне навчання: Від теорії до практики / М. І. Коваленко. – Харків : Фоліо, 2019. – 315 с.
9. Геращенко А. О. Основи програмування на Python : Навчальний посібник / А. О. Геращенко. – Київ : Наукова думка, 2019. – 320 с.
10. Сидоренко П. О. Обробка та аналіз даних з Pandas : Підручник / П. О. Сидоренко. – Запоріжжя : Класика, 2019. – 265 с
11. Документація NumPy [Електронний ресурс] - URL: https://numpy.org/doc/stable/ (дата звернення: 16.05.2024).
12. Офіційна документація scikit-learn [Електронний ресурс] - URL: https://scikit-learn.org/stable/documentation.html (дата звернення: 10.04.2024).
13. Офіційна документація та бібліотека ta [Електронний ресурс] - URL: https://github.com/bukosabino/ta (дата звернення: 15.05.2024).

14. Офіційна документація aiogram [Електронний ресурс] - URL: https://docs.aiogram.dev/en/latest/ (дата звернення: 05.05.2024).

15. Документація SQLite [Електронний ресурс] - URL: https://www.sqlite.org/docs.html (дата звернення: 03.05.2024).

16. Документація Decimal для Python [Електронний ресурс] - URL: https://docs.python.org/3/library/decimal.html (дата звернення: 15.05.2024).

ДОДАТОК А. Основний модуль

```
import os
import ccxt
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.ensemble import RandomForestRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.linear_model import LinearRegression, Ridge
from ta.trend import MACD
from sklearn.pipeline import make_pipeline
from sklearn.compose import make_column_transformer
from aiogram import Bot, Dispatcher, types
from aiogram.contrib.fsm_storage.memory import MemoryStorage
from aiogram.dispatcher import FSMContext
import asyncio
from aiogram.types import ParseMode
from datetime import datetime, timedelta
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
from config import *
from sqlite import *
from keyboard import *
from lng import *
from state import *
from API import *
```

```
bot = Bot(token=API_TOKEN, parse_mode='html')
dp = Dispatcher(bot, storage=MemoryStorage())
print('Bot started')


@dp.callback_query_handler(text="converter")
async def converter_callback_handler(call: types.CallbackQuery, state: FSMContext):
    await call.answer()
    markup = await create_currency_selection_keyboard()
    user_id = call.from_user.id
    lng = await get_user_info(user_id, 'lng')
    await call.message.edit_caption(caption=CHOICE_CRYPTO[lng], reply_markup=markup)
    await ConverterStates.SelectSourceCurrency.set()

@dp.callback_query_handler(lambda call: call.data.startswith('select_source_'), state=ConverterStates.SelectSourceCurrency)
async def source_currency_selected(call: types.CallbackQuery, state: FSMContext):
    await call.answer()
    source_currency = call.data.split('_')[2]
    await state.update_data(source_currency=source_currency)
    markup = await create_currency_selection_keyboard(exclude_currency=source_currency)
    user_id = call.from_user.id
    lng = await get_user_info(user_id, 'lng')
```

```
    await
call.message.edit_caption(caption=SELECTED_CRYPTO[lng].format(source_curren
cy=source_currency.upper()), reply_markup=markup)
    await ConverterStates.SelectTargetCurrency.set()

@dp.callback_query_handler(lambda call: call.data.startswith('select_source_'),
state=ConverterStates.SelectTargetCurrency)
async def target_currency_selected(call: types.CallbackQuery, state: FSMContext):
    await call.answer()
    target_currency = call.data.split('_')[2]
    data = await state.get_data()
    source_currency = data['source_currency']
    if source_currency == target_currency:
        await call.message.edit_caption(caption="You cannot select the same currency
for conversion. Please select a different target currency:")
        return
    await state.update_data(target_currency=target_currency)
    user_id = call.from_user.id
    lng = await get_user_info(user_id, 'lng')
    markup= markup = types.InlineKeyboardMarkup(row_width=2)
    markup.add(back_to_main_page)
    await
call.message.edit_caption(caption=SELECTED_CRYPTO2[lng].format(source_curre
ncy=source_currency.upper(),target_currency=target_currency.upper()),reply_marku
p=markup)
    await ConverterStates.EnterAmount.set()

@dp.message_handler(state=ConverterStates.EnterAmount)
async def amount_entered(message: types.Message, state: FSMContext):
```

```
    try:
        amount = Decimal(message.text)
    except:
        user_id = message.from_user.id
        lng = await get_user_info(user_id,'lng')
        await message.answer(INVALID_NUM[lng])
        return

    data = await state.get_data()
    source_currency = data['source_currency']
    target_currency = data['target_currency']
    markup= markup = types.InlineKeyboardMarkup(row_width=2)
    markup.add(back_to_main_page)

    try:
        converted_amount = await convert_currencies(source_currency, target_currency,
amount)
        await                                    message.answer_photo(photo=PHOTO_ID,
caption=f"{source_currency.upper()}  ({amount:,})  →  {target_currency.upper()}
({converted_amount:,.4f})", reply_markup=markup)
    except Exception as e:
        await message.answer(f"Conversion error: {e}")
    await state.finish()

async def convert_currencies(source_currency, target_currency, amount):
    price_source = await get_crypto_price(source_currency)
    price_target = await get_crypto_price(target_currency)
    amount_in_usd = amount * price_source
    amount_in_target_currency = amount_in_usd / price_target
```

```
    return amount_in_target_currency

async def get_binance_data(crypto_symbol,user_id,lng):
    await bot.send_message(user_id, GET_DATA[lng].format(crypto_symbol=crypto_symbol))
    binance = ccxt.binance()
    symbol = crypto_symbol
    timeframe = '1d'
    ohlcv = binance.fetch_ohlcv(symbol, timeframe, limit=1000)


    if ohlcv:

        df = pd.DataFrame(ohlcv, columns=['Open Time', 'Open', 'High', 'Low', 'Close', 'Volume'])


        df['Open Time'] = pd.to_datetime(df['Open Time'], unit='ms')


        df.set_index('Open Time', inplace=True)


        df = df.apply(pd.to_numeric)


        return df
    else:
        print('No data found')
```

```
        return None


def save_data(df, crypto_symbol):

    df.to_csv(f'{crypto_symbol}.csv')
    print(f"Data saved to {crypto_symbol}.csv")

    os.remove(f'{crypto_symbol}.csv')


def load_data(crypto_symbol):

    df = pd.read_csv(f'{crypto_symbol}.csv', index_col='Open Time',
parse_dates=True)

    return df


def calculate_rsi(data, window=14):
    delta = data.diff()
    gain = delta.mask(delta < 0, 0)
    loss = -delta.mask(delta > 0, 0)
    avg_gain = gain.rolling(window).mean()
    avg_loss = loss.rolling(window).mean()
    rs = avg_gain / avg_loss
    rsi = 100 - (100 / (1 + rs))
    return rsi
```

```
def clean_data(df):
    df.dropna(inplace=True)
    df.drop_duplicates(inplace=True)

    df['RSI'] = calculate_rsi(df['Close'], window=14)

    df['MA'] = df['Close'].rolling(window=20).mean()

    macd = MACD(df['Close'], window_slow=26, window_fast=12, window_sign=9)
    df['MACD'] = macd.macd()

    return df

def train_model(df):
    X = df.drop('Close', axis=1)
    y = df['Close']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
    column_transformer = make_column_transformer(
        (SimpleImputer(), X_train.columns),
        remainder='passthrough'
    )


    models = [
        ('Random Forest', make_pipeline(column_transformer, RandomForestRegressor())),
        ('Decision Tree', make_pipeline(column_transformer, DecisionTreeRegressor())),
        ('Linear Regression', make_pipeline(column_transformer, LinearRegression())),
        ('Ridge', make_pipeline(column_transformer, Ridge()))
    ]


    model_scores = {}


    for name, model in models:

        model.fit(X_train, y_train)

        score = model.score(X_test, y_test)
        model_scores[name] = score


    return model_scores
```

```
def print_model_scores(model_scores):

    for i, (name, score) in enumerate(model_scores.items()):
        model_name = list(model_scores.keys())[i]
        print(f'{model_name} Model R-squared: {score*100:.2f}%')


def predict_price(df):

    X = df.drop('Close', axis=1)
    y = df['Close']


    imputer = SimpleImputer()
    X = imputer.fit_transform(X)


    scaler = StandardScaler()
    scaler.fit(X)
    X = scaler.transform(X)


    models = [
        ('Random Forest', RandomForestRegressor()),
        ('Decision Tree', DecisionTreeRegressor()),
        ('Linear Regression', LinearRegression()),
```

```
        ('Ridge', Ridge())
    ]

    predictions = []

    for name, model in models:

        model.fit(X, y)

        prediction = model.predict(X[-30:])

        predictions.append(prediction)

    predictions = np.array(predictions).T

    return predictions


def plot_predictions(predictions, file_name):

    future_dates = [datetime.now() + timedelta(days=i) for i in range(1, 31)]


    model_names = ['Random Forest', 'Decision Tree', 'Linear Regression', 'Ridge']


    plt.figure(figsize=(18, 9))
```

```
    for i, prediction in enumerate(predictions.T):

        reversed_prediction = prediction[::-1]
        plt.plot(range(30), reversed_prediction, label=f'Прогнози моделі {model_names[i]}', linestyle='--')
        for j, price in enumerate(reversed_prediction):
            plt.text(j, price, f'{price:.2f}', fontsize=8)

    plt.xlabel('Дата')
    plt.ylabel('Ціна')
    plt.title('Прогнози цін на криптовалюту на наступні 30 днів')
    plt.legend()
    plt.grid(True)


    plt.xticks(rotation=45)


    ax = plt.gca()
    ax.set_xticks(range(30))
    ax.set_xticklabels([date.strftime('%Y-%m-%d') for date in future_dates])


    plt.tight_layout()

    plt.savefig(file_name)
    plt.close()
```

```
def save_predictions(predictions, crypto_symbol, file_name):

    df = pd.DataFrame(predictions)

    model_names = [f'Model {i+1}' for i in range(predictions.shape[1])]

    df.columns = model_names

    df.to_csv(file_name)
    print(f"Predictions saved to {file_name}")



@dp.message_handler(text=list(PROFILE_BUTTON_TEXT.values()), state='*')
async def profile_button_handler(message: types.Message, state: FSMContext):
    await state.finish()
    user_id = message.from_user.id
    username = message.from_user.username
    first_name = message.from_user.first_name
    if not await get_user_exist(user_id):
        register_date = str(datetime.now()).split()[0]
        referal_id = message.get_args()
        referal_id = referal_id if referal_id!=user_id else None
        await add_user(user_id, username, first_name, referal_id, register_date)
        if referal_id:
            try:
                await bot.send_message(referal_id, f'У вас новий реферал!\n<a
href="tg://user?id={user_id}">{first_name}</a> - <code>{user_id}</code>')
            except: pass
    bot_username = (await bot.get_me()).username
```

```
    referal_link = f'https://t.me/{bot_username}?start={user_id}'
    lng, register_date = await get_user_info(user_id, 'lng, register_date')
    if not lng:
        await message.answer('Select a language:', reply_markup=choice_lng_keyboard())
        return
    markup = main_keyboard(lng)
    await message.answer_photo(photo=PHOTO_ID, caption=START_MESSAGE[lng].format(name=first_name, user_id=user_id, register_date=register_date, referal_link=referal_link), reply_markup=markup)

@dp.message_handler(commands=['start'], state='*')
async def cmd_start(message: types.Message, state: FSMContext):
    await state.finish()
    user_id = message.from_user.id
    username = message.from_user.username
    first_name = message.from_user.first_name
    if not await get_user_exist(user_id):
        register_date = str(datetime.now()).split()[0]
        referal_id = message.get_args()
        referal_id = referal_id if referal_id!=user_id else None
        await add_user(user_id, username, first_name, referal_id, register_date)
        if referal_id:
            try:
                await bot.send_message(referal_id, f'У вас новий реферал!\n<a href="tg://user?id={user_id}">{first_name}</a> - <code>{user_id}</code>')
            except: pass
    bot_username = (await bot.get_me()).username
    referal_link = f'https://t.me/{bot_username}?start={user_id}'
```

```
    lng, register_date = await get_user_info(user_id, 'lng, register_date')
    if not lng:
        await message.answer('Select a language:', reply_markup=choice_lng_keyboard())
        return
    await message.answer('👋', reply_markup=main_navigation_keyboard(lng))
    markup = main_keyboard(lng)
    await message.answer_photo(photo=PHOTO_ID, caption=START_MESSAGE[lng].format(name=first_name, user_id=user_id, register_date=register_date, referal_link=referal_link), reply_markup=markup)

@dp.callback_query_handler(text_startswith='change_lng', state='*')
async def callback_change_lng(call: types.CallbackQuery, state: FSMContext):
    await state.finish()
    user_id = call.from_user.id
    lng = await get_user_info(user_id, 'lng')
    try:
        await call.message.edit_caption(caption=CHOICE_LNG_MESSAGE[lng], reply_markup=choice_lng_keyboard(back_btn_status=True))
    except:
        await call.message.answer_photo(photo=PHOTO_ID, caption=CHOICE_LNG_MESSAGE[lng], reply_markup=choice_lng_keyboard(back_btn_status=True))
    await call.answer()


@dp.callback_query_handler(text_startswith='main_page', state='*')
async def callback_main_page(call: types.CallbackQuery, state: FSMContext):
    await state.finish()
```

```
    user_id = call.from_user.id
    username = call.from_user.username
    first_name = call.from_user.first_name
    lng, register_date = await get_user_info(user_id, 'lng, register_date')
    if not lng:
        await call.message.edit_text('Select a language:', reply_markup=choice_lng_keyboard())
        return
    bot_username = (await bot.get_me()).username
    referal_link = f'https://t.me/{bot_username}?start={user_id}'
    markup = main_keyboard(lng)
    caption = START_MESSAGE[lng].format(name=first_name, user_id=user_id, register_date=register_date, referal_link=referal_link)
    try:
        await call.message.edit_caption(caption=caption, reply_markup=markup)
    except:
        await call.message.answer('👋', reply_markup=main_navigation_keyboard(lng))
        await call.message.answer_photo(photo=PHOTO_ID, caption=caption, reply_markup=markup)


@dp.callback_query_handler(text_startswith='choice_lng', state='*')
async def callback_choice_lng(call: types.CallbackQuery, state: FSMContext):
    await state.finish()
    lng = call.data.split(';')[1]
    user_id = call.from_user.id
    await update_user_info(user_id, 'lng', lng)
    await bot.delete_message(chat_id=call.message.chat.id, message_id=call.message.message_id)
```

```
    call.data = 'main_page'
    await callback_main_page(call, state)



@dp.callback_query_handler(text_startswith='forecast', state='*')
async def callback_forecast(call: types.CallbackQuery, state: FSMContext):
    markup= markup = types.InlineKeyboardMarkup(row_width=2)
    markup.add(back_to_main_page)
    user_id = call.from_user.id
    lng = await get_user_info(user_id,'lng')
    await call.message.edit_caption(caption=FORECAST_MESSAGE[lng],
reply_markup=markup)
    await CryptoState.WaitingForSymbol.set()

@dp.message_handler(state=CryptoState.WaitingForSymbol)
async def process_crypto_symbol(message: types.Message, state: FSMContext):
    async with state.proxy() as data:
        data['crypto_symbol'] = message.text.upper()
    user_id = message.from_user.id
    lng = await get_user_info(user_id, 'lng')

    crypto_symbol = message.text.upper()
    markup = types.InlineKeyboardMarkup()
    markup.add(back_to_main_page)
    try:
        data = await get_binance_data(crypto_symbol, user_id, lng)
    except:
```

```
        await message.answer_photo(photo=PHOTO_ID, caption=ERROR_CRYPTO[lng], reply_markup=markup)
        return

    if data is not None:

        cleaned_data = clean_data(data)

        model_scores = train_model(cleaned_data)

        print_model_scores(model_scores)

        predictions = predict_price(cleaned_data)

        csv_file_name = f'{crypto_symbol}.csv'
        save_predictions(predictions, crypto_symbol, csv_file_name)

        plot_file_name = f'{crypto_symbol}_predictions.png'
        plot_predictions(predictions, plot_file_name)

        await bot.send_message(message.chat.id, RESULT_CRYPTO[lng])
        await bot.send_document(message.chat.id, open(csv_file_name, 'rb'))
        await bot.send_photo(message.chat.id, open(plot_file_name, 'rb'))

        os.remove(csv_file_name)
        os.remove(plot_file_name)
    else:
        await message.answer(ERROR_SYMBOL[lng])
    await state.finish()
```

```
loop = asyncio.get_event_loop()
loop.run_until_complete(create_table_users())
loop.create_task(dp.start_polling())
loop.run_forever()
```

ДОДАТОК Б. Модуль клавіатури

```
from aiogram import types
from lng import *
from config import *
from sqlite import *


back_btn_text = '⬅'
back_to_main_page = types.InlineKeyboardButton(text=back_btn_text, callback_data='main_page')

def choice_lng_keyboard(back_btn_status=False):
    markup = types.InlineKeyboardMarkup(row_width=1).add(
    types.InlineKeyboardButton(text='UA Український', callback_data='choice_lng;uk'),
    types.InlineKeyboardButton(text='GB English', callback_data='choice_lng;eng'),
    types.InlineKeyboardButton(text='PL Polski', callback_data='choice_lng;pl'),
    )
    if back_btn_status:
        markup.add(back_to_main_page)
    return markup

def main_navigation_keyboard(lng):
    markup = types.ReplyKeyboardMarkup(resize_keyboard=True)
    markup.add(PROFILE_BUTTON_TEXT[lng])
    return markup

def main_keyboard(lng):
```

```
    markup = types.InlineKeyboardMarkup(row_width=2)
    markup.add(types.InlineKeyboardButton(text=CONVERTER[lng],
callback_data='converter'))
    markup.add(types.InlineKeyboardButton(text=FORECAST[lng],
callback_data='forecast'))
    markup.add(
        types.InlineKeyboardButton(text=CHANGE_LNG_BTN[lng],
callback_data='change_lng'),
        types.InlineKeyboardButton(text=SUPPORT_BTN[lng], url=SUPPORT_URL),
        )
    return markup



async def create_currency_selection_keyboard(exclude_currency=None):
    keyboard = types.InlineKeyboardMarkup()
    currencies = ['bitcoin', 'ethereum', 'litecoin', 'usd']
    for currency in currencies:
        if currency != exclude_currency:
            callback_data = f'select_source_{currency}'
            button = types.InlineKeyboardButton(text=currency.upper(),
callback_data=callback_data)
            keyboard.add(button)
    keyboard.add(back_to_main_page)
    return keyboard
```

ДОДАТОК В. Модуль конфігурації

```
API_TOKEN = '1710347244:A**********'
DB_PATH='C:/Users/Yeshenko/ Desktop/bot/main.db'
SUPPORT_URL = 'https://t.me/****'
SUPPORT_USERNAME = '@****'
PHOTO_ID='AgACAgIAAxkDAAIJ22Zaa1vVX_M7*****'
```

```
START_MESSAGE = {
                    'eng': 'ID: <code>{user_id}</code>\n\nName: <code>{name}</code>\n\nRegister date: <code>{register_date}</code>\n\n<a href="{referal_link}">Referral link</a>',
                    'pl': 'ID: <code>{user_id}</code>\n\nNazwa: <code>{name}</code>\n\nData rejestracji: <code>{register_date}</code>\n\n<a href="{referal_link}">Link polecający</a>',
                    'uk': 'ID: <code>{user_id}</code>\n\nІм`я: <code>{name}</code>\n\nДата реєстрації: <code>{register_date}</code>\n\n<a href="{referal_link}">Реферальне посилання</a>',
                    }


CHANGE_LNG_BTN = {
                    'eng': '🌐 Change language',
                    'pl': '🌐 Zmień język',
                    'uk': '🌐 Змінити мову',
                    }
PROFILE_BUTTON_TEXT = {
            'eng': 'Profile',
                    'pl': 'Profil',
                    'uk': 'Профіль',
}

ERROR_SYMBOL = {
                'eng': 'No data was found for the specified cryptocurrency symbol.',
```

```
                'pl': 'Nie znaleziono danych dla podanego symbolu kryptowaluty.',
                'uk': 'Дані для цього символу криптовалюти не знайдені.',
}
SUPPORT_BTN = {
                'eng': '👩‍💻 Tech.Support',
                'pl': '👩‍💻 Wsparcie techniczne',
                'uk': '👩‍💻 Тех.Підтримка',
                }

CONVERTER = {
                'eng': '🔃 Converter',
                'pl': '🔃 Przetwornik',
                'uk': '🔃 Конвертер',
}

FORECAST = {
                'eng': '🔥 Forecast',
                'pl': '🔥 Prognoza',
                'uk': '🔥 Прогноз',
}

CHOICE_LNG_MESSAGE = {
                        'eng': '🌐 Choose language:',
                        'pl': '🌐 Wybierz język:',
                        'uk': '🌐 Оберіть мову:',
                        }
```

```
FORECAST_MESSAGE = {
                    'eng': '📝 Enter the symbol of the cryptocurrency (e.g., BTCUSDT):',
                    'pl': '📝 Wpisz symbol kryptowaluty (np. BTCUSDT):',
                    'uk': '📝 Введіть символ криптовалюти (наприклад, BTCUSDT):',
}

INVALID_NUM = {
                'eng': '❌ Invalid amount. Please enter a valid number.',
                    'pl': '❌ Nieprawidłowa kwota. Proszę wprowadzić poprawny numer.',
                    'uk': '❌ Помилка числа. Введіть число.',
}

CHOICE_CRYPTO = {
                'eng': '💲 Select a cryptocurrency pair:',
                    'pl': '💲 Wybierz parę kryptowalut:',
                    'uk': '💲Оберіть пару криптовалют:',
}



SELECTED_CRYPTO = {
                'eng': 'The first cryptocurrency is selected: {source_currency}\n\nSelect the target currency:',
```

'pl': 'Wybrana jest pierwsza kryptowaluta: {source_currency}\n\nWybierz walutę docelową:',

'uk': 'Вибрано першу криптовалюту: {source_currency}\n\nВиберіть цільову валюту:',

}

SELECTED_CRYPTO2 = {

'eng': 'The first cryptocurrency is selected: {source_currency}\nTarget cryptocurrency selected: {target_currency}\n\nEnter a number:',

'pl': 'Wybrana jest pierwsza kryptowaluta: {source_currency}\nWybrano docelową kryptowalutę: {target_currency}\n\nWpisz numer:',

'uk': 'Вибрано першу криптовалюту: {source_currency}\nВибрано цільову криптовалюту: {target_currency}\n\nВведіть число:',

}

ERROR_CRYPTO = {

'eng': '✕ Error entering cryptocurrency symbols',

'pl': '✕ Błąd przy wprowadzaniu symboli kryptowaluty',

'uk': '✕ Помилка введення символів криптовалют',

}

```
GET_DATA = {
                    'eng': '🔄 Getting {crypto_symbol} data from Binance...',
                                'pl': '🔄 Pobieranie danych {crypto_symbol} z Binance...',
                                'uk': '🔄 Отримання даних {crypto_symbol} з Binance...',

}


RESULT_CRYPTO = {
                    'eng': '✅ Data received successfully',
                                'pl': '✅ Dane odebrane pomyślnie',
                                'uk': '✅ Дані успешно отримано',
}START_MESSAGE = {
                                'eng': 'ID: <code>{user_id}</code>\n\nName: <code>{name}</code>\n\nRegister date: <code>{register_date}</code>\n\n<a href="{referal_link}">Referral link</a>',
                                'pl': 'ID: <code>{user_id}</code>\n\nNazwa: <code>{name}</code>\n\nData rejestracji: <code>{register_date}</code>\n\n<a href="{referal_link}">Link polecający</a>',
                                'uk': 'ID: <code>{user_id}</code>\n\nІм`я: <code>{name}</code>\n\nДата реєстрації: <code>{register_date}</code>\n\n<a href="{referal_link}">Реферальне посилання</a>',
                                }


CHANGE_LNG_BTN = {
```

```
                    'eng': '🌐 Change language',
                    'pl': '🌐 Zmień język',
                    'uk': '🌐 Змінити мову',
                    }
PROFILE_BUTTON_TEXT = {
            'eng': 'Profile',
                    'pl': 'Profil',
                    'uk': 'Профіль',
}

ERROR_SYMBOL = {
                'eng': 'No data was found for the specified cryptocurrency symbol.',
                    'pl': 'Nie znaleziono danych dla podanego symbolu kryptowaluty.',
                    'uk': 'Дані для цього символу криптовалюти не знайдені.',
}
SUPPORT_BTN = {
                    'eng': '👩‍💻 Tech.Support',
                    'pl': '👩‍💻 Wsparcie techniczne',
                    'uk': '👩‍💻 Тех.Підтримка',
                    }

CONVERTER = {
                'eng': '🔃 Converter',
                    'pl': '🔃 Przetwornik',
                    'uk': '🔃 Конвертер',
}
```

```
FORECAST = {
        'eng': '🔥 Forecast',
        'pl': '🔥 Prognoza',
        'uk': '🔥 Прогноз',
}

CHOICE_LNG_MESSAGE = {
        'eng': '🌐 Choose language:',
        'pl': '🌐 Wybierz język:',
        'uk': '🌐 Оберіть мову:',
        }

FORECAST_MESSAGE = {
        'eng': '📝 Enter the symbol of the cryptocurrency (e.g., BTCUSDT):',
        'pl': '📝 Wpisz symbol kryptowaluty (np. BTCUSDT):',
        'uk': '📝 Введіть символ криптовалюти (наприклад, BTCUSDT):',
}

INVALID_NUM = {
        'eng': '❌ Invalid amount. Please enter a valid number.',
        'pl': '❌ Nieprawidłowa kwota. Proszę wprowadzić poprawny numer.',
        'uk': '❌ Помилка числа. Введіть число.',
}
```

```
CHOICE_CRYPTO = {
                    'eng': '💲 Select a cryptocurrency pair:',
                        'pl': '💲 Wybierz parę kryptowalut:',
                        'uk': '💲Оберіть пару криптовалют:',
}



SELECTED_CRYPTO = {
                    'eng': 'The first cryptocurrency is selected: {source_currency}\n\nSelect the target currency:',
                        'pl': 'Wybrana jest pierwsza kryptowaluta: {source_currency}\n\nWybierz walutę docelową:',
                        'uk': 'Вибрано першу криптовалюту: {source_currency}\n\nВиберіть цільову валюту:',
}

SELECTED_CRYPTO2 = {
                    'eng': 'The first cryptocurrency is selected: {source_currency}\nTarget cryptocurrency selected: {target_currency}\n\nEnter a number:',
                        'pl': 'Wybrana jest pierwsza kryptowaluta: {source_currency}\nWybrano docelową kryptowalutę: {target_currency}\n\nWpisz numer:',
                        'uk': 'Вибрано першу криптовалюту: {source_currency}\nВибрано цільову криптовалюту: {target_currency}\n\nВведіть число:',
}
```

```
ERROR_CRYPTO = {
                    'eng': '❌ Error entering cryptocurrency symbols',
                        'pl': '❌ Błąd przy wprowadzaniu symboli kryptowaluty',
                        'uk': '❌ Помилка введення символів криптовалют',

}


GET_DATA = {
                'eng': '🔄 Getting {crypto_symbol} data from Binance...',
                        'pl': '🔄 Pobieranie danych {crypto_symbol} z Binance...',
                        'uk': '🔄 Отримання даних {crypto_symbol} з Binance...',

}


RESULT_CRYPTO = {
                'eng': '✅ Data received successfully',
                        'pl': '✅ Dane odebrane pomyślnie',
                        'uk': '✅ Дані успешно отримано',
}
```

ДОДАТОК Ґ. Модуль бази даних

```
import aiosqlite
import asyncio
from datetime import datetime, timedelta
from config import DB_PATH


async def create_table_users():
    async with aiosqlite.connect(DB_PATH) as db:
        cursor = await db.cursor()
        await cursor.execute('CREATE TABLE IF NOT EXISTS Users(user_id INTEGER, username TEXT, first_name TEXT, register_date DATE, referal_id INTEGER,lng TEXT)')
        await db.commit()

async def get_user_info(user_id, params):
    async with aiosqlite.connect(DB_PATH) as db:
        cursor = await db.cursor()
        sql_request = f'SELECT {params} FROM Users WHERE user_id = ?'
        await cursor.execute(sql_request, (user_id,))
        response = await cursor.fetchone()
        if ',' in params:
            return response
        else:
            return response[0]


async def update_user_info(user_id, column, param):
```

```
    async with aiosqlite.connect(DB_PATH) as db:
        cursor = await db.cursor()
        sql_request = f'UPDATE users SET {column} = ? WHERE user_id = ?'
        await cursor.execute(sql_request, (param, user_id))
        await db.commit()


async def get_user_exist(user_id):
    async with aiosqlite.connect(DB_PATH) as db:
        cursor = await db.cursor()
        await cursor.execute('SELECT Count() FROM Users WHERE user_id = ?', (user_id,))
        return (await cursor.fetchone())[0]


async def add_user(user_id, username, name, referal_id, register_date):
    async with aiosqlite.connect(DB_PATH) as db:
        cursor = await db.cursor()
        await cursor.execute('INSERT INTO Users(user_id,username,first_name,referal_id,register_date) SELECT ?,?,?,?,? WHERE NOT EXISTS (SELECT 1 FROM users WHERE user_id = ?) ', (user_id, username, name, referal_id, register_date, user_id))
        await db.commit()
```