КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра інтелектуальних програмних систем

Кваліфікаційна робота
на здобуття освітнього рівня бакалавра
за спеціальністю 121 Інженерія програмного забезпечення
на тему:

РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ ТРЕЙДИНГУ

Виконав студент 4-го курсу
Гліб ПИЛИПЕЦЬ _____
(підпис)

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

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

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри інтелектуальних програмних систем
«____» _____ 2021 р.,
протокол № ___
Завідувач кафедри
Олександр ПРОВОТАР

_____
(підпис)

Київ – 2021

РЕФЕРАТ

Обсяг роботи 50 сторінок, 17 ілюстрацій, 4 таблиці, 30 джерел посилань.

АВТОМАТИЗОВАНА ТРЕЙДИНГ СИСТЕМА, АЛГОРИТМІЧНИЙ ТРЕЙДИНГ, БРОКЕР, ЕЛЕКТРОННА ТОРГІВЛЯ, СТРАТЕГІЯ ТОРГІВЛІ, ТЕХНІЧНИЙ ІНДИКАТОР, ТЕХНІЧНИЙ АНАЛІЗ, ТОРГОВИЙ РОБОТ, ТРЕЙДИНГ, ФОНДОВА БІРЖА, ФОРЕКС, ФОРЕКС РОБОТ, ФУНДАМЕНТАЛЬНИЙ АНАЛІЗ.

Об'єктом роботи є процеси біржової торгівлі та розробки програмного забезпечення для автоматизації процесу трейдингу на них.

Предметом роботи є автоматизована трейдинг система або торговий робот та група стратегій для алгоритмічної торгівлі за допомогою брокера FXCM.

Метою роботи є створення програмного застосунку для автоматизації трейдингу за допомогою брокера FXCM, систематизація матеріалів в області алгоритмічної торгівлі, розгляд основних аспектів алготрейдингу у порівнянні з ручним з метою відповіді на питання, як та чи можливо зробити роботизований трейдинг прибутковим.

Методи розроблення: системний підхід, ООП проектування та програмування, бектестинг, методологія розробки FDD (Feature driven development).

Інструменти розроблення: Spyder IDE (інтерактивна кросплатформна IDE для наукових обчислень та розрахунків на мові Python), середовище розробки Visual Studio 2019, Git (система контролю версій), мова програмування Python 3.9.5 з додатковими пакетами, серед основних – Backtesting, fxcmpy, stocktrends, statsmodels, numpy, pandas, matplotlib.

Результати роботи та їх новизна: виконано огляд основних аспектів алгоритмічної торгівлі, описано процес розробки торгового робота – від створення та тестування стратегії до її реалізації на торговому роботі та деплоймента, розроблено програмну систему “Трейдинг робот”, яка дозволяє торгувати за допомогою брокера FXCM за вибраною стратегією та конфігурацією, а також розширити функціонал, додавши нову стратегію.

Застосування та значущість результатів роботи: ця робота може використовуватися розробниками та/або іншими людьми, що хочуть отримати навички та розуміння процесів автоматизації трейдингу не тільки теоретично, а й на прикладі створення програмної системи, що його використовує. Створене ПЗ може бути розширене та адаптоване під інші типи стратегій чи ринків або використовуватися в наявному вигляді для торгівлі через брокера FXCM.

Висновки та пропозиції щодо розвитку об’єкта розроблення й доцільності продовження розробок: сфера трейдингу має високий поріг входу через інвестиційні ризики та недостатність матеріалів з описом його компонент та процесів – у цій роботі було описано основні частини алгоритмічного трейдингу, а також спроектовано та розроблено трейдинг робота. Напрямок трейдингу є відкритим в тому значенні, що створюються нові API для автоматизації, ML моделі, алгоритми для торгівлі, а стабільного рішення для заміни мануального трейдингу ще не створено, тому подальші розробки в цій сфері є актуальними.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

AI – Artificial Intelligence, штучний інтелект;

API – Application Programming Interface, інтерфейс прикладного програмування;

ATR – Average True Range індикатор;

ATS – Automated Trading System, автоматизована трейдинг система або торговий робот;

CFD – Contract For Difference, торгова угода, що є контрактом на різницю цін;

FDD – Feautre Driven Development, методологія розробки;

FIX протокол – Financial Information Exchange протокол, протокол обміну фінансовою інформацією;

FOREX – Foreign Exchange, ФОРЕКС, біржа валют;

FXCM – Forex Capital Markets, брокер для трейдингу на FOREX та CFD;

HFT – High Frequency Trading, високочастотний трейдинг;

IB – Interactive Brokers, брокер для торгівлі на фондових біржах США;

IDE – Integrated Development Environment, інтегроване середовище розробки;

KPI – Key Performance Indicators, основні показники ефективності;

ML – Machine Learning, машинне навчання;

NASDAQ – National Association of Securities Dealers Automated Quotation, Автоматизовані котирування Національної асоціації дилерів цінних паперів;

NYSY – New York Stock Exchange, Нью-Йоркська фондова біржа;

OHLC свічки – Open-High-Low-Close свічки, тобто дані беруться за певний період.

ВСТУП

Оцінка сучасного стану об'єкта розробки. Моментом зародження алгоритмічного торгівлі можна вважати появу першої автоматизованої системи біржової торгівлі – NASDAQ в 1971 [0]. Недостатність продуманості торгових систем та стратегій їх роботи за однією з версій призвело до найбільшого за всю історію торгівлі обвалу фондового ринку США в 1976 році, що назвали Black Monday. Ще тоді були розкритиковані торгові роботи, але час йшов і діджиталізація світу не могла залишити сферу трейдингу, тобто ринків з великим потенціалом і капіталізацією, без уваги.

З розвитком фондових ринків США чи валютник ринків, підвищувався і попит на рішення по прогнозуванню поведінки активів на таких ринках – адже знаючи наперед, де буде ціна активу, можна було заробити на різниці цін. Зокрема, з'являлися інструменти для аналізу графіків та інших факторів, що могли б вплинути на вартість активу. У першій половині XX століття такі інструменти були поділені на фундаментальний та технічний аналізи. Автоматизовану торгову систему або торгового робота було вперше описано Тушаром Чендом в 1997 році як рішення для торгівлі на Форекс в книзі "Beyond Technical Analysis" [0].

У 2021 алготрейдинг лише закріпив свої позиції – торгові роботи можуть розпізнавати значно більше чарт патернів, застосовувати комплексні аналітичні методи для виявлення та впровадження ефективних стратегій, надавати надійність в роботі, але пік розвитку ще не досягнуто. Новим ринком для торгівлі стали криптовалюти, що надають простоту в використанні – немає ні брокера, ні регулювань на державному рівні, тому одразу й набули популярності. Відносно новим рішенням у сферах розробки стратегій є застосування штучного інтелекту та глибокого навчання для передбачень цін на ринку.

Актуальність роботи та підстави для її виконання. Сферу алгоритмічного трейдингу на 2021 рік описує багато книг, серед яких я можу

рекомендувати такі як [2], [3], [4], [5], однак процес трейдингу та його автоматизації має високий поріг входу в плані вивчення та залишається відкритим з боку розробок. Він поєднує в собі математику, економіку та програмування – понад 90% відсотків людей після спроб трейдингу втрачають кошти [6] чи покидають цю діяльність, що лише підтверджує ствердження вище.

Стосовно відкритості в плані розробок, ядром торгового робота є трейдинг стратегія, яку він використовує, а проблемою на даний момент для стратегій є їх час життя, або прибуткової роботи. Тобто стратегії, що були ефективними та принесли мільйони тиждень/місяць/рік тому, є неприбутковими зараз – приклади таких стратегій можна знайти в книзі [7]. Жоден з учасників ринку не стане ділитися прибутковою трейдинг стратегією, бо як тільки трейдери почнуть її використовувати, то патерн зміни ціни активу відрізнятиметься, адже ціна формується внаслідок угод, які здійснюють учасники торгівлі.

На початку шляху в вивченні трейдингу я стикнувся з проблемою великої кількості літератури, але нерозумінням того, з чого почати його освоєння, а також малою кількістю матеріалів чи наявних відкритих рішень в напрямку автоматизації.

Бажання дослідити процеси трейдингу для розуміння аспектів інвестування, електронної торгівлі, фондової, валютної та крипто бірж, ідея використати всі переваги швидкості комп'ютеризованих систем у поєднанні з навичками програмування для автоматизації торгівлі, а також формування відповіді на питання можливості отримання прибутку від торгових роботів чи стратегій, які рекламуються в інтернеті, були підставами для виконання цієї роботи.

Мета й завдання роботи. Метою даної кваліфікаційної роботи є створення програмного забезпечення для автоматизації процесів трейдингу за допомогою брокера FXCM, аналіз та розв’язання основних проблем, що він включає на прикладі власної розробки з метою формування справедливої відповіді на питання можливості прибуткового використання систем

алгоритмічної торгівлі. Для реалізації поставленої мети в межах кваліфікаційної роботи було визначено наступні завдання:

1. Провести аналіз предметної області;
2. Описати основні проблеми та етапи в розробці торгового робота;
3. Спроектувати та розробити автоматизовану трейдинг систему, що дозволяла б використовувати одну з наявних торгових стратегій та визначити власну;
4. Реалізувати та автоматизувати торговий алгоритм на розробленій системі;

Об’єкт і методи розроблення. Об’єктами дослідження в даній роботі є алгоритмічна торгівля, торгові стратегії та біржова торгівля, об’єктом розроблення – автоматизована трейдинг система або торговий робот. Методами розроблення є ООП проектування та програмування з використанням методології розробок FDD.

Можливі сфери застосування. Ця робота зображує з практичної точки зору етапи та проблеми, з якими стикаються програмісти при автоматизації торгових рішень, а також розв’язує ці проблеми на прикладі розробки власного продукту. Після її освоєння можна отримати чітке розуміння природи торгових роботів та принципів їх роботи. Практична частина може використовуватися в наявному вигляді для тестування, автоматизації власних торгових стратегій чи як основа для розширення більш комплексними рішеннями.

Робота складається з трьох основних частин – теоретичне підґрунтя під алгоритмічним трейдингом, архітектура та використання розробленої торгової системи, автоматизація та тестування (бектестинг, демо-тестинг) торгової стратегії.

РОЗДІЛ 1. АЛГОРИТМІЧНИЙ ТРЕЙДИНГ

1.1 Що таке алгоритмічний трейдинг та автоматизована трейдинг система?

Під алгоритмічним трейдингом мається на увазі використання алгоритмів для відкриття торгових угод на електронних платформах. Тобто комп'ютеризоване виконання замовлень по визначених правилах та інструкціях у противагу використанню телефону чи факсу для зв'язку з брокером та виконанню ордерів. Є різні підкатегорії алгоритмічного трейдингу в залежності від роду виконуваних задач, але в межах даної роботи буде розглядатися автоматизована трейдинг система.

Автоматизована трейдинг система (ATS) – підмножина алгоритмічного трейдингу, комп'ютеризована система, що слідує певному алгоритму, або визначеному набору інструкцій, для визначення сприятливих умов та виконання buy/sell угод в автоматизованій манері з метою отримання прибутку. В простому випадку, ATS торгує на біржі через брокера, в складніших – з використанням FIX протоколу [8] без явного брокера. Переваги та недоліки алгоритмічної торгівлі можна знайти в розділі 1.4.

Компонентами такої системи є стратегія та брокер чи місце для здійснення торгових угод. Автоматизована торгівля може базуватися на будь-яких математичних моделях чи алгоритмах – зокрема, можна аналізувати такі параметри як час, кількість угод, сентимент ринку, економіку, показники компаній, ціну та інші. Більше детальніше про трейдингові стратегії можна знайти в розділі 3.1.

Згідно зі статистикою Kissell Research Group (рис. 1.1), частка об'єму торгових угод, що виконуються алгоритмами на фондових ринках США зросла з 0.1% у 2000 до 93% у 2019. Найбільш стрімкою була тенденція зростання в серпні 2007 року через обвал ринку та продовжилася в 2008-2009 через фінансовий кризис. Саме в ці періоди процеси мануального трейдингу були

нестабільними – постійна волатильність, різкі зміни цін, інсайди (витоки інформації), затримки часу виконання з метою отримання вигоди. Ці фактори змусили інвесторів прийняти рішення по переходу на більш надійний електронний та алгоритмічний трейдинг.

Рисунок 1.1 – Частка алготрейдингу на фондових біржах США. [5]

Важливо зазначити, що алгоритмічна торгівля не можлива на ринках (фондових та товарних) України, що можна пояснити низький рівнем розвитку біржової торгівлі, економіки.

1.2 Ринки для алгоритмічного трейдингу

Тепер з’ясуємо, на яких ринках можна здійснювати угоду в алгоритмічній манері та їх основні особливості. На травень 2021 для алгоритмічного трейдингу наявні 3 ринки:

1. Форекс (Forex Exchange) – міжнародний децентралізований ринок для торгівлі валютними парами. Це найбільший (2.409 квадрильйонів доларів США ринкова капіталізація) та найбільш ліквідний ринок (6.6 трильйонів доларів в середньому за день) серед усіх наявних [9]. Завдяки децентралізації працює ~24/5.

2. Криптобіржа або біржа цифрових валют (Crypto exchange) – це окремі сервіси чи платформи, що дозволяють користувачам торгувати цифровими валютами та обмінювати їх на інші цифрові валюти чи фіатні кошти. Криптобіржі не урегульовані на державному рівні, тому простіші у використані – працюють 24/7, 1179.06 мільярдів доларів США ринкова капіталізація, найбільш популярною є платформа Binance [10].
3. Фондова біржа (Stock exchange) – урегульована на рівні законодавства група ринків або акціонерне товариство, що являє собою ринок цінних паперів. Зокрема, фондова біржа поєднує попит (трейдери, сток брокери, інвестори) та пропозицію для цінних паперів. Зазвичай, мова йде про торгівлю акціями публічних компаній. Час роботи залежить від фондової біржі, але в середньому 05:30-20:00 Пн-Пт по локальному часу біржі. Ринкова капіталізація та ліквідність залежить від фондової біржі – наприклад, NYSE (25.62 трильйонів доларів), NASDAQ (19.51 трильйонів доларів з денним оборотом 258 мільярдів доларів) [11]. Рекомендованими брокерами є Interactive Brokers як міжнародний стандарт, або Фрідом Фінанс для України.
4. Контракти на різницю (CFD) – в окремий пункт можна виділити укладення контрактів на різницю або торгівлю CFD, яку надають багато брокерів, що дозволяє спекулювати та заробляти на зміні ціни активу, при цьому не здійснюючи його покупки/продажу – таким чином область трейдингу можна розширити на інші, окрім наявних на 3 ринках вище, активи. Прикладом брокера, що надає CFD є FXCM.

Якщо підсумувати, то купівля акцій на фондовій біржі є найбільш перспективною через саму суть стоків – отримання дивідендів, володіння часткою компанія, прозорість фундаментальних даних, ріст ціни портфоліо при успішній діяльності компанії. Але для України є суттєва проблема при роботі з акціями:

– Слабко розвинута власна фондова біржа, а тому українські трейдери торгують на ринках США;

– Рекомендовано щонайменше 3000$ на депозиті для початку торгівлі на фондовій біржі, для інтрадей (тобто більша кількість торгових угод в день) вимога U.S. для учасників щонайменше 25’000$ статків в наявності [12]. З деталей, абонплата брокеру IB (~20$), комісія за ордери (~1$), оплата податків (~18-20%) при виведенні коштів в Україну [13].

Найбільш зручним та прозорим серед зазначених ринків будуть криптобіржі через нерегульованість на рівні законодавства, а тому спрощена робота багатьох процесів. До недоліків можна віднести всі ризики, що пов’язані з нефіатними коштами та складність аналізу такого ринку.

Forex є невигідним для малих торгових позицій (табл. 1.1), бо зміна ціни валютних пар там відбувається найчастіше в тисячних чи менше від цілої та описується як піп.

Таблиця 1.1 – Приклад виконання угоди та отримання прибутку на ФОРЕКС.

Час	Тип	Валютна пара	Кількість	Ціна		Вартість позиції	Прибуток (USD)
1:54:31 PM	Sell/Close	GBP/USD	100,000	$1.3162	Bid	$131,620.00	$150.00
11:22:16 AM	Buy	GBP/USD	100,000	$1.3147	Ask	$131,470.00	-

Вище на таблиці 1.1 вище показано, що при вартості позиції більш як 100 тисяч доларів був зроблений прибуток в 150$ – тобто потенційний прибуток буде суттєво менше вартості позиції. Звісно, зазвичай очікується більша зміна ціни за пару валют та використовується торгове плече, але суть від цього не змінюється.

CFD представляє спрощену в використанні, але урізану в можливостях версію для торгівлі на фондовій/валютній/криптовалютній біржі, бо фактичних угод на них ви не здійснюєте – прибуток можливий лише з різниці цін для усіх типів угод.

1.3 На чому базується алгоритмічний трейдинг

Маючи розуміння того, що собою являє ATS та на яких ринках використовується, опишемо, завдяки чому алгоритмічний трейдинг є можливим, тобто які підходи використовуються в ньому. В даній роботі мова йтиме про внутрішньоденний (intraday або short-term) трейдинг, тобто час утримання однієї позиції від декількох хвилин до днів. Метою такого трейдингу є отримання прибутку на різниці цін, при цьому з меншим часом затримки капіталу – купити активи дешевше, щоб продати дорожче чи навпаки.

Трейдинг система, використовуючи стратегію, робить передбачення стосовно того, як зміниться ціна, та на основі цього приймається рішення про продаж/покупку активів. На 2021 рік для алгоритмічного трейдингу наявні 3 основні підходи [14] до передбачення зміни ціни активу:

- Фундаментальний аналіз;
- Технічний аналіз;
- Машине навчання та штучний інтелект;

Фундаментальний аналіз – це метод оцінки ціни активу шляхом вивчення суміжних економічних та фінансових факторів, що можуть на неї впливати. Фундаментальні аналітики роблять аналіз стану економіки, показників діяльності компанії, останніх новин, і т.д. з метою передбачення недооцінки чи переоцінки активу. Фундаментальний аналіз частіше використовується для інвестицій чи довготривалого трейдингу та є найбільш надійним серед 3 зазначених. Прикладами того на які параметри варто звертати увагу при фундаментальному аналізі, можна знайти в метриках, що були розроблені інвесторами та гуру фундаментального аналізу з його розвитком – Magic Formula (Джоел Грінблат), F-Score (Джозеф Піотроски), формула Бенджаміна Грехема та Z-Score (Едвард Альтман).

Технічний аналіз – метод передбачення та оцінки величини росту чи зниження ціни активу по історичних даних ринку. Він базується на припущенні,

що ринок рухається по встановлених трендах і патернах, що були вже раніше, бо всі чинники, що можуть вплинути на ціну вже враховані в ній. Типовим прикладом технічного аналізу є використання індикаторів та патернів таких як MA (Moving Average), RSI (Relative strength index), ATR (Average true range), Renko, MACD (MA convergence&divergence), Support, Fibonacci Retracements, Resistance, Boolinger Bands і їм подібних для генерації трейдинг сигналів – дивитися [25] з прикладами патернів на графіку. Такі індикатори базуються на математичному аналізі рядів та статистиці, детальніше про технічний аналіз та опис різних технічних патернів та індикаторів можна знайти за посиланням [15]. В певній мірі технічний аналіз є самореалізованим пророцтвом, бо трейдери помічають патерни та рухають ціну в очікуваному напрямку, укладаючи торгові угоди.

Машинне навчання та штучний інтелект – використання технік машинного навчання для передбачення ціни, що також комбінується з підбором параметрів стратегії, що буде адаптуватися до змін з часом. Прикладами можуть бути штучні нейронні мережі (ANN) для апроксимації функції ціни, дерева рішень (DT), що формуються для генерації торгових сигналів в залежності від стану параметрів активу, генетичні алгоритми (GA). Дослідження та пошук моделей з використанням машинного навчання для передбачення ціни активів це те, чим займаються Data Scientists в трейдингових компаніях – оптимізовані моделі на базі ML мають більшу ефективність у порівнянні з алгоритмами, що базуються виключно на технічному аналізі.

Нижче на рис. 1.2 наведено схематично основні аспекти технічного та фундаментального аналізів:

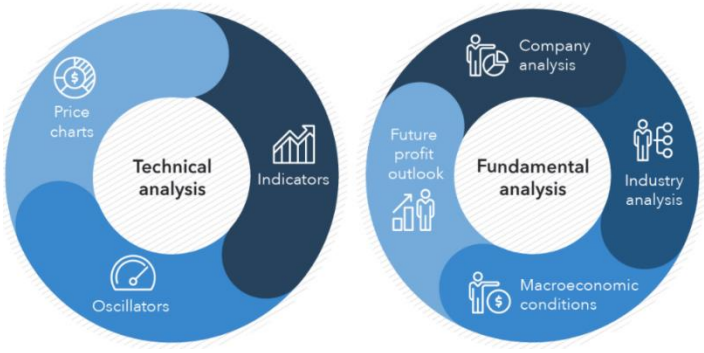

Рисунок 1.2 – Рисунок технічний та фундаментальний аналіз. [16]

Серед безкоштовних ресурсів для виконання фундаментального та технічного аналізу найкращими, на мій погляд, є TradingView та finviz, що часто використовуються у поєднанні:

– TradingView – платформа з торговою спільнотою 8+ мільйонів в Інтернеті, надає фундаментальні та технічні дані з графіками, а також спільноту з обговореннями, торговими ідеями, містить можливості додати наявні технічні індикатори чи створити нові, та багато іншого.
– Finviz (Financial Visualizations) – веб-сайт, що надає фінансовий аналіз, новини, дослідження та візуалізацію поведінки акцій на фондовому ринку.

1.4 Переваги та недоліки алгоритмічної торгівлі

На даному етапі було розглянуто основні поняття алготрейдингу, на чому він базується, а також ринки для торгівлі. Розглянемо, які є перспективи та потенціал використання автоматизованих трейдинг рішень. Професія трейдера вимагає терпіння та стійкості до стресів, сконцентрованості та уміння аналізувати статистичні дані, управляти ризиками – трейдер повинен слідкувати за змінами цін, робити математичні та аналітичні розрахунки для визначення умов входу та виходу з позицій – ATS намагається автоматизувати ці всі процеси зі зведенням участі людини в ньому до мінімуму. Такий підхід має наступні переваги над мануальним [29]:

– Швидкість роботи та надійність роботи – одна з найбільших переваг алготрейдингу це можливість комп’ютерів обчислювати різні параметри та технічні індикатори за мілісекунди та відправляти ордери на виконання як тільки досягнуті певні умови значно швидше за людину. Під надійністю мається на увазі спроектована, протестована та реалізована система на мові програмування, що виконує покрокові інструкції, а тому вибір активів, відкриття та закриття ордерів стають покроковими інструкціями, що робить процес трейдингу більш прозорим;

- Мінімальна участь людини – це дозволяє зменшити кількість помилок, в тому числі емоційних та дотримуватися чіткого стратегічного плану. Частими проблемами мануального трейдингу є торгівля не по плану через зміну стану ринку – сумніви, що ціна досягне бажаного рівня, бажання продати на піку ціни або боязнь продати на падінні, що часто призводить до втрати потенційного прибутку – в алготрейдингу такого бути не може, бо виставляються чіткі правила виходу з та входу в позиції, в залежності від вибраних стратегій ризику йде розприділення капіталу. Тобто стратегія залишається незмінною, незалежно від емоцій, страху, жадібності трейдерів. Не можуть бути також неправильно розпізнані технічні індикатори, бо алгоритм протестований.
- Можливість бектестингу (backtesting), демо-тестингу (paper trading) – бектестинг являє собою симуляцію виконання стратегії на великих об’ємах історичних даних з оптимізацією параметрів, а демо-тестинг це наступна стадія симуляції на реальних стрімінгових даних та демо-аккаунті. Обидва типи тестування стратегії показують очікувані основні показники діяльності(KPI) стратегії, щоб визначитися чи достатньо оптимальна стратегія для виходу на вибраний ринок торгівлі. Ефективний бектестинг робиться лише за допомогою алгоритмів через обробку усіх історичних або великої кількості з них. Більше детально на прикладі буде розглянуто ці процеси в розділах 3.5, 3.6;
- Диверсифікація та автопідбір портфоліо – під диверсифікацією мається на увазі можливість використання одночасно декількох торгових стратегій чи різних ринків, що значно ускладнюється при мануальній торгівлі, під автопідбором портфоліо мається на увазі ребаланс чи вибір нових активів в залежності від їх стану на ринку.

Розглянемо основні недоліки алгоритмічного трейдингу чи проблеми, пов’язані з ним:

- Втрата людського контролю – комп’ютери ще не достатньо розвинуті для прийняття рішень на рівні людини, тому для уникнення суттєвих втрат потрібно контролювати роботу роботів. Бот має бути запрограмований так, щоб його можна було зупинити з мінімальними втратами в ручному режимі по запиту чи автоматично при певних умовах;
- Залежність від технологій – подвійні ордери, відновлення стану торгівлі, пропущені ордери через втрату/затримки з’єднання, механічні пошкодження є можливими, тому це треба враховувати при побудові інфраструктури для трейдинг системи;
- Не всі стратегії можуть бути автоматизовані – в основному, через комплексність реалізації;
- Короткий час життя алгоритму – проблема алгоритмічного трейдингу, в тому, що стратегія може перестати просто працювати через зміну стану ринку, а тому її потрібно постійно оновлювати, адаптувати під зміни;

Враховуючи порівняння вище, можна відмітити, алгоритмічний трейдинг як і будь-який процес має як ряд переваг, так і недоліків, але моя думка, що з розвитком AI та процесу удосконалення цифрових технологій алгоритмічний трейдинг витіснятиме мануальний.

РОЗДІЛ 2. АВТОМАТИЗОВАНА ТРЕЙДИНГ СИСТЕМА (ТОРГОВИЙ РОБОТ)

2.1 Визначення цілей розробки та вимог до системи

Ціллю проекту є розробка торгового робота, що буде підключатися до брокера за вказаним акаунтом та виконувати в автоматизованій манері вибрану одну з наперед визначених торгових стратегій – дані для акаунту та стратегії мають бути конфігурованими, брокер фіксований. Програмний код має бути написаний так, щоб була можливість розширення набору торгових стратегій власною.

Розглянемо основні питання, на які має дати відповідь специфікація вимог:

a) Дані: для кожної стратегії потрібні дані – це можуть бути дані для бектестингу, чи історичні з певним зсувом, чи стрімінгові для роботи стратегії. В деяких випадках потрібно платити для отримання даних з малою затримкою або для отримання великої кількості історичних даних. Основні питання: Як отримати такі дані, які дані потрібні? Де будуть зберігатися отримані дані – наприклад, БД?
b) Ринки торгівлі: від ринків торгівлі – акції, криптовалюта, валюта, CFD – залежить брокер, а від брокера дуже часто залежить архітектура системи. Тобто торгових роботів пишуть під конкретного брокера – не дивлячись на те, що функціонал у брокерів буде схожим, але часто інфраструктура API буде різною, тому зміна брокера вимагатиме змін коду.
c) Робота з ордерами: маючи торгову стратегію, що генерує сигнали, потрібно дати відповіді про можливість використання наступного функціоналу:
    1. Створення замовлення з нуля чи надати рішення по визначенню шаблонів ордерів;
    2. Модифікація та скасування існуючих невиконаних замовлень;
    3. Перевірка запланованих ордерів та відкритих позицій;

d) Надійність та хостинг: як гарантувати стабільність роботи? Чи буде робот запускатися локально чи деплоїтися на клауді? Чи потрібно роботу додаткові ресурси для роботи стратегій чи збереження стану – БД, файли?
e) Торгова стратегія та сигнали: які типи торгових стратегій будуть підтримувані роботом? Як будуть генеруватися сигнали торгових стратегій та на чому базуватися? Протягом якого часу запускати торгову стратегію, адже стратегія може бути розрахована під конкретні години роботи ринку.
f) Мова програмування: в релізному випадку для трейдинг роботів майже завжди використовуються різні готові бібліотеки, що інкапсулюють логіку рутингу замовлень, стрімінгу даних, можливо, надають графічний інтерфейс, електронну систему для здійснення замовлень і інші. Такі бібліотеки залежать від ринку торгівлі та мов програмування – тому мова програмування є важливою.

2.2 Специфікація вимог до проекту та декомпозиція задач

Спочатку зазначимо, що в межах розробки всі використані технології є безкоштовними – також буде зазначено, як можна оптимізувати систему з використанням платних рішень в висновках.

Вимоги до торгового робота:

- Обрахунок різних технічних індикаторів з можливістю додавання нових;
- Отримання історичних даних – OHLC свічки, кількість угод(volume) – для вибраного асету та гранулярності;
- Отримання стрімінгових даних (час, ціна, кількість активів в угоді) за тік для вибраного асету;
- Відкриття та закриття buy/sell торгових угод;
- Менеджмент даних по акаунту (позицій, замовлень, активності, отримання актуальних bid/ask оферів, рахунку і т.д.);
- Конфігурованість параметрів та даних для використовуваного акаунта та стратегії;

В окремі пункти варто винести наступні вимоги до системи:

– Створення прикладів торгових стратегій – зокрема, будуть реалізовані 3 стратегії: Сітчата торгівля, Ренко графік/лінії Боллінджера у поєднанні з індикатором MACD;
– Надання архітектури для створення власної торгової стратегії шляхом комбінації наявних технічних індикаторів чи інших інструментів;

Специфікація вибраних рішень для реалізації представлена в табл. 2.1.

Таблиця 2.1 – Специфікація вимог ATS

Дані	FXCM – для отримання стрімінгових + інтрадей даних в режимі роботи; Dukascopy – для отримання історичних даних для бектестингу;
Ринки торгівлі	FXCM брокер – Форекс та CFD, демо-аккаунт;
Робота з ордерами	Можливість створення шаблонів + модифікація, відміна, перегляд існуючих ордерів/позицій;
Надійність та хостинг	Локально(AWS EC2)
Мова програмування	Python3
Торгова стратегія та сигнали	Сигнали базуються на технічних індикаторах чи статистичних моделях, ітеративна робота стратегії протягом заданого часу. Детальніше дивитися розділ 3.

Якщо прокоментувати прийняті рішення вище, то FXCM це єдиний брокер серед фондових бірж та Форексу, що надає можливість безкоштовного отримання стрімінгових, історичних даних з демо-аккаунтом.

Alpaca API, що позиціонується як API для алгоритмічного трейдингу, у випадку безкоштовної підписки надає дані з 1-годинною затримкою від останніх актуальних, що унеможливлює його використання.

Alpha Vintage є безкоштовним дата провайдером історичних та стрімінгових даних для стоків компаній, в тому числі інтрадей до 30 днів, але сильно урізаний максимальний період історичних інтрадей даних та необхідність брокеру роблять це API непотрібним для даної системи.

Python3 взятий для імплементації через широкий спектр додаткових бібліотек, що суттєво спрощує роботу з мережею, математичними даними та зменшує кількість написаного коду.

2.3 Архітектура системи

В цілому, архітектура системи бути поділена на 2 основні компоненти – модуль з торговим роботом та модуль зі стратегіями, що використовує торгового робота. Тому нижче буде представлено та описано діаграм класів для ATS з двох перспектив – торгового робота (рис. 2.1) та з боку стратегій (рис. 2.2).

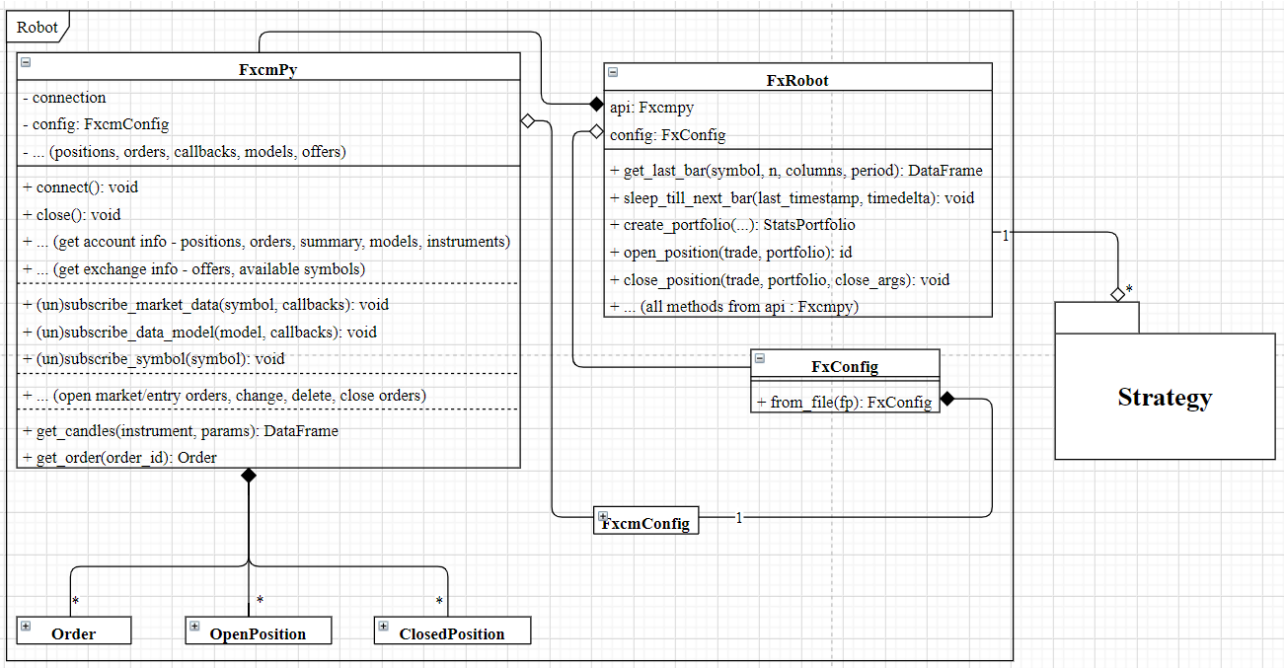

Рисунок 2.1 – Архітектура ATS з перспективи торгового робота

Найважливішим компонентом серед наявних є FxRobot, що включає FxcmPy:

FxRobot – робот, що приймає конфігурації акаунта на FXCM брокері та містить необхідний функціонал для використання стратегіями, взаємодії з FXCM API [17]. Детальніше: має доступ до відкритих/закритих позицій за сесію, даних акаунту, надає інструменти для використання publisher/subscriber патерна з сервером, отримання історичних свічок для символа, очікування наступної свічки, створення та відправлення ордерів на біржу, створення портфоліо.

FxcmPy є ядром FxRobot – власною модифікацією пакета fxcmpy, що використовує з'єднання по веб-сокету з сервером. Для з'єднання використовується саме веб-сокет для забезпечення можливості стрімінгу даних з боку-сервера. FXCM API надає наступні типи підписок на стрімінгові моделі

даних – {'Offer', 'Account', 'Order', 'OpenPosition', 'ClosedPosition', 'Summary'}. Клієнт через FxRobot може підписатися на одну з таких моделей та перевизначити хендлер обробки відповідних даних при отриманні від сервера чи робити запит на снепшоти, які зберігаються за умовчанням після підписки локально. Окрім моделей вище, FXCM API, а відповідно FxRobot, надає можливість підписки на отримання торгових тіків (даних при мінімальній зміні ціни асету) для вибраного символа – це є необхідним для стратегій, які використовують меншу гранулярність аніж свічки. Окрім publisher/subscriber патерна, доступні й звичайні методи, що звертаються до локальних даних чи сервера – наприклад get_order(…) візьме дані від локального об'єкта, що зберігає активні/закриті позиції, а get_candles(…) відправить запит на сервер.

Якщо підсумувати, то клієнт може взаємодіяти з FXCM API трьома способами – стрімінг тіків чи моделей, відправлення запитів на сервер по REST та звичайні методи. FxRobot кешує дані за останню сесію локально. Детальніше приклад роботи з FxRobot буде розглянуто в розділі 2.4.

Окрім FxRobot є також класи Order, OpenPosition, ClosedPosition, де Order являє собою замовлення, що ще не виконане – може бути entry чи market ордер. OpenPosition\ClosedPosition це відкриті та закриті позиції по певних символах. Відкрита позиція утворюється після виконання ордеру на біржі та описує в якій кількості та якими активами власник акаунта наразі володіє – містить такі дані як айді ордеру, акаунта, дата відкриття/закриття, ціна, параметри зупинки(take_profit, stop_loss), об'єм та тип угоди та інші.

Тепер розглянемо архітектуру ATS з боку стратегії:

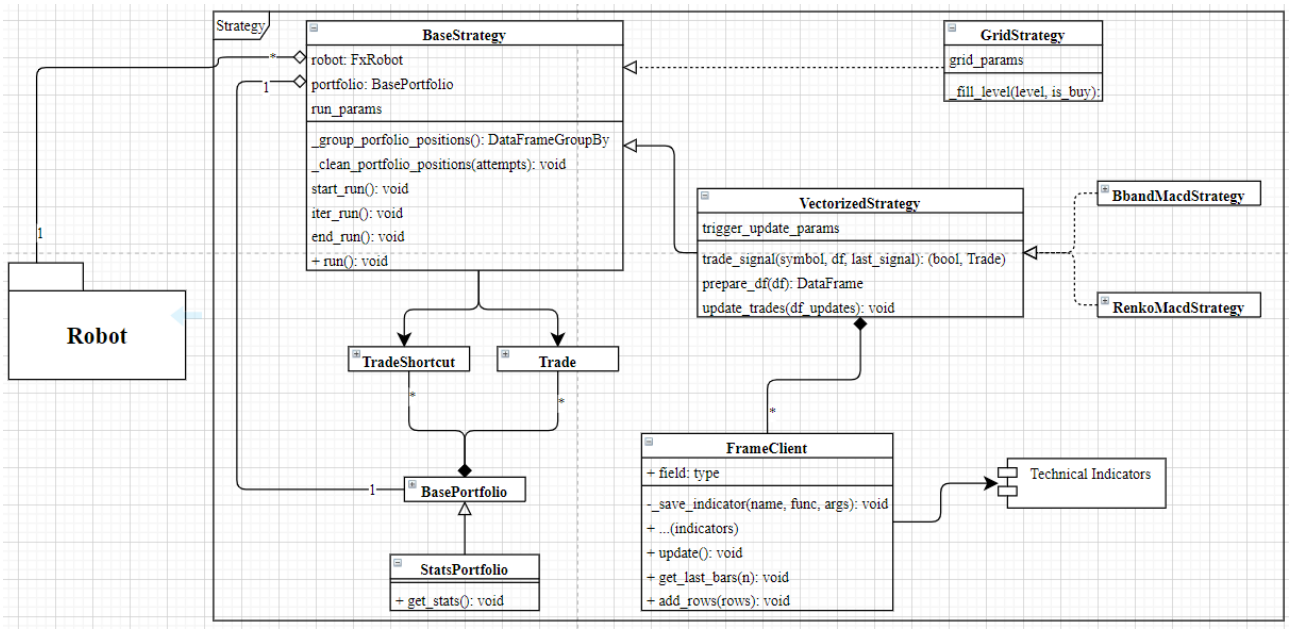

Рисунок 2.2 – Архітектура ATS з перспективи стратегії

Основні компоненти:

– BaseStrategy – абстрактний клас стратегії, що може бути використана клієнтом. Клас надає 3 основні методи на перевизначення потомкам – start_run(ініціює запуск стратегії), iter_run(виконується в циклі протягом заданого часу), end_run(завершує запуск стратегії). Для запуску стратегії використовується метод run(), в скороченому вигляді клас містить функціонал як нижче.

```
class BaseStrategy:
    def __init__(self, robot : FxRobot, portfolio : Portfolio, run_for : pd.
Timedelta, ...):
        ...
    def run(self):
        self.start_run()
        ...
        while time.time() < run_until:
            try:
                ...
                self.iter_run()
            ...
        self.end_run()
```

В розділі 3.3 буде розглянуто приклад використання класу BaseStrategy для визначення класу GridStrategy в межах стратегії Сітчата торгівля.

– TradeShortcut та Trade – шаблон для створення трейдів та трейд відповідно. Клас Trade становить собою ще не відправлений на виконання Order та містить інформацію таку як символ, кількість, ціна, тип угоди, stop_loss, take_profit та інші параметри – використовується для відкриття/закриття позицій через FxRobot.
– BasePortfolio та StatsPortfolio – інкапсулюють ідею портфоліо, що містить дані про відкриті/закриті в межах нього позиції, а також надають статистику по ньому. FxRobot містить методи open_trade, close_trade, що приймають Portfolio та Trade для відкриття та закриття позиції.
– FrameClient – врепер над pandas.DataFrame, що дозволяє застосовувати різного роду технічні індикатори поверх нього, використовуючи модуль Techical Indicators. Такий DataFrame буде містити свічки даних по символу, що представляються OHLC дані + об'єм угод;
– VectorizedStrategy – абстрактний клас, що розширює BaseStrategy та інкапсулює логіку стратегій, що отримують та обробляють OHLC свічки послідовно. Для кожного символу з Portfolio відводиться відповідний FrameClient, який оновлюється та використовується для генерації сигналів.

```
class VectorizedStrategy(BaseStrategy):
    def __init__(self, update_period, bars_period, update_bars_cnt = 1,
                 trigger_frame_size = 0, max_frame_size = 500,
                 init_bars_cnt = 0, ...)
```

VectorizedStrategy приймає параметрами для роботи частоту апдейтів(update_period), період свічок(bars_period), кількість свічок для апдейту(update_bars_cnt), кількість свічок для тригеру сигналу(trigger_frame_size) та початкову кількість свічок(init_bars_cnt). При отриманні від FxRobot наступної OHLC свічки робиться оновлення FrameClient, та при виконанні відповідних умов викликається метод update_trades, що у свою чергу викликає trade_signal для отримання та виконання наступних ордерів чи зміни попередніх.

В розділах 3.4, 3.6 буде розглянуто стратегії RenkoMacd, BbandMacd, що працюють за описаною вище логікою.

– Модуль Technical Indicators містить реалізацію різних технічних індикаторів, які використовуються над pandas.DataFrame, pandas.DataSeries – ATR, RenkoDF, slope, BollingerBands, MACD. Технічні індикатори використовуються стратегіями як складові для прийняття рішень про відкриття/закриття позицій та генерації сигналів.

2.4 Робота та можливості використання

Для кращого розуміння того, з якими даними працює описана система та який функціонал надає, перед автоматизацією стратегій покажемо декілька прикладів роботи з ATS.

Маючи, конфігураційний файл (рис. 2.3), його потрібно вказати для створення класу FxRobot:

```
config = FxConfig.from_file("config/init_config.ini")
bot = FxRobot(config)
```

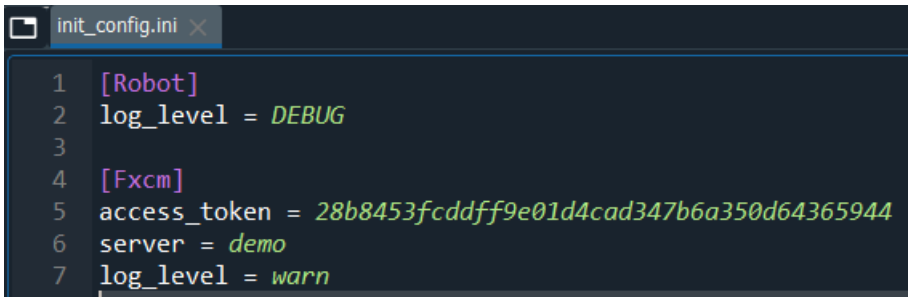

Рисунок 2.3 – Конфігурація класу FxRobot

FxRobot надає змогу отримати дані за тік чи OHLC свічки, які можуть використовуватися торговими стратегіями:

– Ohlc свічки (рис. 2.4) зберігають open, high, low, close значення ціни для ask/bid, а також кількість куплених активів за вибраний період.

```
df = bot.get_last_bar('EUR/USD', n = 100, columns=[], period = 'H4')
```

date	open	close	high	low	askopen	askclose	askhigh	asklow	volume
2021-04-29 17:00:00	1.21173	1.21175	1.21288	1.2112	1.21184	1.21239	1.21301	1.21131	17382
2021-04-29 21:00:00	1.21175	1.21214	1.21255	1.21151	1.21239	1.21226	1.2129	1.21182	11108
2021-04-30 01:00:00	1.21214	1.21165	1.2126	1.21147	1.21226	1.21176	1.21273	1.2116	16412
2021-04-30 05:00:00	1.21165	1.20945	1.21193	1.20937	1.21176	1.20958	1.21206	1.2095	31101
2021-04-30 09:00:00	1.20945	1.20831	1.20977	1.20767	1.20958	1.20843	1.2099	1.20778	37261
2021-04-30 13:00:00	1.20831	1.20261	1.20915	1.20234	1.20843	1.20272	1.20926	1.20246	78548
2021-04-30 17:00:00	1.20261	1.20155	1.20298	1.20155	1.20272	1.20248	1.2031	1.20172	25364
2021-05-02 17:00:00	1.20155	1.20251	1.20347	1.20241	1.20248	1.20324	1.20393	1.2032	155
2021-05-02 21:00:00	1.20313	1.20332	1.20333	1.20195	1.20356	1.20344	1.20367	1.20208	8746
2021-05-03 01:00:00	1.20332	1.20205	1.20349	1.20164	1.20344	1.20216	1.20361	1.20177	12421
2021-05-03 05:00:00	1.20205	1.20467	1.20491	1.20125	1.20216	1.20479	1.20502	1.20136	33029
2021-05-03 09:00:00	1.20467	1.20424	1.20571	1.20412	1.20479	1.20436	1.20583	1.20424	33965

Рисунок 2.4 – Приклад OHLC свічок

На рисунку 2.4 зображено приклад таких свічок в pandas.DataFrame – кількість (n = 100) з періодом (period = ‘H4’), тобто свічка сформована кожні 4 години для пари ‘EUR/USD’. Схожим чином клас FrameClient інкапсулює збереження OHLC свічок та дозволяє накладати різні технічні індикатори поверх нього.

– Дані за тік – для прикладу підпишемося на акцію, індекс та форекс бакет:

```
bot.subscribe_instrument(['PLTR.us', 'SPX500', USDOLLAR])
bot.subscribe_market_data(['PLTR.us', 'SPX500', USDOLLAR], (log_method,))
```

Тепер FxRobot буде від FXCM серверу отримувати дані в найменшій доступній гранулярності – тіках – по вибраних підписках, зберігати дані в pandas.DataFrame та викликати хендлер на кожен такий тік (рис. 2.5).

```
In [3]: bot.subscribe_instrument(['PLTR.us', 'USDOLLAR', 'SPX500'])
   ...: bot.subscribe_market_data(['PLTR.us', 'USDOLLAR', 'SPX500'], (log_method,))
  2 | SPX500 | 2021-05-25 22:07:09.649000, 4192.99, 4193.41, 4194.98, 4191.92
  2 | USDOLLAR | 2021-05-25 22:07:10.203000, 11677.8, 11679.8, 11683, 11674.2
  3 | USDOLLAR | 2021-05-25 22:07:10.402000, 11677.8, 11679.8, 11683, 11674.2
  4 | USDOLLAR | 2021-05-25 22:07:11.304000, 11677.9, 11679.9, 11683, 11674.2
  3 | SPX500 | 2021-05-25 22:07:11.597000, 4193.1, 4193.52, 4194.98, 4191.92
  5 | USDOLLAR | 2021-05-25 22:07:11.714000, 11677.8, 11679.8, 11683, 11674.2
  6 | USDOLLAR | 2021-05-25 22:07:12.303000, 11677.9, 11679.9, 11683, 11674.2
  7 | USDOLLAR | 2021-05-25 22:07:12.715000, 11677.9, 11679.9, 11683, 11674.2
  4 | SPX500 | 2021-05-25 22:07:13.493000, 4193.15, 4193.57, 4194.98, 4191.92
  8 | USDOLLAR | 2021-05-25 22:07:13.803000, 11677.9, 11679.9, 11683, 11674.2
```

Рисунок 2.5 – Отримання даних за тік.

Стосовно створення ордерів та відкриття позицій, стратегії, що використовують FxRobot, можуть створювати замовлення в межах BasePortfolio та без:

– Через портфоліо: `order = robot.open_trade(trade, portfolio)`
– Без портфоліо:

```
order = robot.create_entry_order(symbol='GBP/CAD', is_buy=True,
amount=150, limit=1.7,    is_in_pips = True,    time_in_force='GTC', rate=1.7,
stop=-8, trailing_step=1)
```

На прикладах вище буде створено Order об’єкт з визначеними параметрами, що можна побачити через FXCM Trading desktop та метод get_orders (рис. 2.6):

```
bot.get_orders()
Out[8]:
   t  ratePrecision    orderId  ... trailingStop trailing range
0  3              5  129134388  ...      dynamic      0.1     0
```

Рисунок 2.6 – Приклад створення ордеру на FXCM

Торгова стратегія буде генерувати прибуток відкриваючи та закриваючи позиції по активах, використовуючи торгового робота, тобто в спрощеному вигляді схема генерації прибутку наступна:

– BaseStrategy->Сигнал ->Trade->Order->OpenPosition;
– BaseStrategy->Сигнал->Trade->Order->ClosedPosition->Прибуток;

Торгова стратегія періодично генеруватиме сигнали на створення Trade, або торгових угод, які хоче виконати, що можуть бути створені з TradeShortcut або без. Далі, перед відправленням торгової угоди на біржу, Trade стане Order, після виконання ордеру FxRobot отримає від серверу FXCM апдейт та сформує відкриту позицію, яку можна буде переглянути зі стратегії чи іншим клієнтом методом get_open_position.

Відкрита позиція може бути закрита автоматично на рівні біржі, якщо були встановлені та досягнуті такі параметри як stop_loss, take_profit, що відповідають за автоматичне закриття угоди при досягненні ціною певних значень або по сигналу від стратегії – тоді діям вище передуватиме формування Trade, виконання Order. Після закриття FxRobot отримає апдейт та сформує ClosedPosition.

РОЗДІЛ 3. АВТОМАТИЗАЦІЯ ТОРГОВОЇ СТРАТЕГІЇ

3.1 Трейдинг стратегія та їх типи

Трейдинг стратегія це алгоритм, що генерує сигнали для покупки/продажу активів на ринку та базується на визначених правилах для прийняття рішень з метою отримання прибутку на різниці цін активу. Трейдинг стратегії можуть використовувати один або декілька з методів передбачення ціни, що були зазначені в розділі 1.3. Розробка прибуткової торгової стратегії є найскладнішим завданням та слабким місцем в ефективному використанні трейдинг роботів.

Схематично всі трейдинг стратегії можна категоризувати як на рисунку 3.1, [14]:

Рисунок 3.1 – Типи трейдинг стратегій

Стратегії суто на технічному аналізі будують технічні індикатори, що являють собою математичні функції, на основі історичних даних активу, періодично обновляючи їх, а далі комбінують ці індикатори з метою отримання сигналів на купівлю/продаж активу. Статистичні моделі включають сітчату торгівлю, арбітражні стратегії та інші.

Стратегії на основі машинного навчання поєднують фундаментальні та технічні дані – вони можуть використовуватися для визначення та реалізації параметризованих стратегій чи як моделі для передбачень.

В даній роботі буде автоматизовано стратегії, що базуються на технічному аналізі та статистичній моделі та оцінено ефективність деяких з них:

1. Сітчата торгівля (Grid Trading) – вважається однією з найкращих та найбільш надійних стратегій для волатильних активів на визначених інтревалах – складна в реалізації на практиці. В спрощеному вигляді, сітчата торгівля ділить очікувану область волатильності ціни активу на рівні та формує sell/buy угоди на встановлених інтервалах симетрично відносно ціни, далі прибуток формується після досягнення ціною відповідного рівня, а сітка оновлюється для підтримки рівнів [22].
2. Стратегії Фібоначчі – стратегія, що використовує рівні корекції Фібоначчі [28] для виявлення потенційних ліній support та resistance, на основі яких прогнозуються тренди та очікувані величини для значень stop_loss та take_profit.

Дивитися [22], [28] з зображенням сітчатої торгівлі та стратегією Фібоначчі.

3. Трендові стратегії та стратегії прориву [26], [27] – торгові стратегії, що базуються на виявленні висхідних/спадних трендів та ідентифікації зміни попереднього тренду/прориву уявних ліній support, resistance відповідно. Такі стратегії простіші в реалізації у порівнянні з двома категоріями вище, бо зводяться до застосування математичних функцій на наборі даних, тобто програмування технічних індикаторів, а також їх комбінування для отримання торгових сигналів.

Дивитися [26], [27], з зображенням трендової торгівлі та стратегії прориву.

4. Інші ідеї стратегій, що можуть бути вже навіть протестованими можна знайти в енциклопедії стратегій алгоритмічного трейдингу [18]. Альтернативним джерелом для стратегій є опис використаних алгоритмів від самих же авторів трейдинг роботів, що вже є на ринку – зокрема, список на Forex Robot Trader [19]. Варто звернути увагу, що цей ресурс демонструє роботу реальних

стратегій на штучних даних (це видно з детального аналізу місячних логів), тому його варто використовувати як ілюстрацію ідей того, як працюють реальні роботи, але не як робота стратегій на реальному ринку.

Якщо приводити конкретні приклади в минулому прибуткових інтрадей або короткочасних трейдинг стратегій, то, посилаючись на книгу [7], можна виділити Double 7s, The TRIN Strategy, VIX Stretches, VIX RSI, Cumulative RSI-3.

3.2 Етапи автоматизації трейдинг стратегії

Трейдинг стратегія є найбільш важливим компонентом торгового робота, адже саме від неї залежить прибуток та стабільність доходу. Автоматизацію алгоритмічної торгової стратегії заведено робити в 5 етапів (рис. 3.2), [20]:

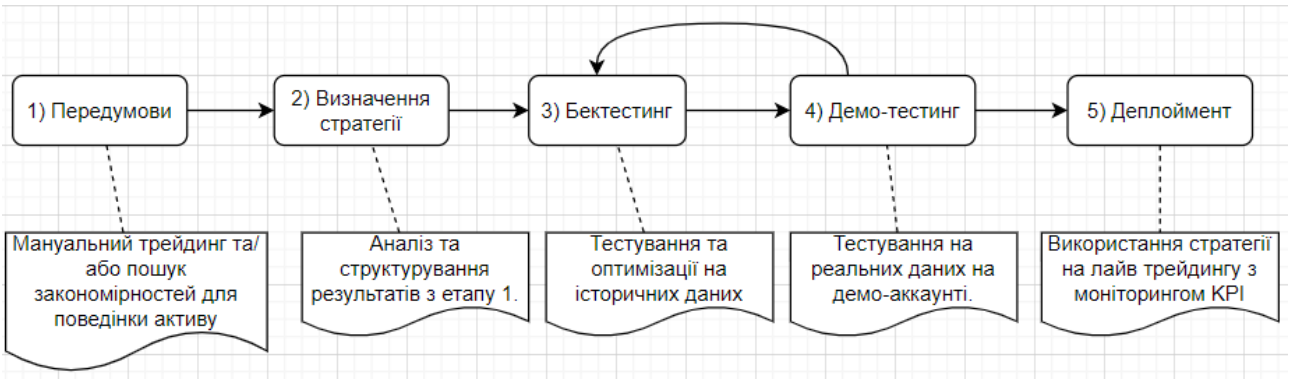

Рисунок 3.2 – Етапи автоматизації торгової стратегії.

Зупинимося детальніше на кожному з етапів:

1. Передумови – це виконання ордерів за допомогою пейпер [21] чи реального трейдингу, перевірка припущень на графіку зміни ціни, пошук матеріалів та написання заміток з метою виявлення закономірностей в русі параметрів активу в залежності від комбінацій даних – ціни, об’єму, новин, часу торгівлі, сентименту. Тобто на цьому етапі йде пошук потенційних стратегій для автоматизації.
2. Визначення стратегії – аналіз результатів з першого етапу для виявлення найбільш перспективних кандидатів по стратегіях. На даному етапі виділяють

основні характеристики стратегій та аналізують складність автоматизації, активи, на яких стратегія може використовуватися та KPI, що можуть бути підраховані для портфоліо після здійснення замовлень. До основних метрик відносяться win/loss коефіцієнт, CAGR, volatility, sharpe/sortino коефіцієнти, calmar, максимальне падіння (maximum drawdown), кількість прибутків та втрат, середній прибуток, середня втрата, прибуток за час роботи та інші – на прикладі будуть оцінені в розділі 3.5.

3. Бектестинг – це етап оптимізації параметрів стратегії та вимірювання метрик з попереднього етапу на великих об’ємах даних. На даному етапі стратегія ще не повністю реалізована. Ідея полягає в імітації поведінки стратегії, де замість реальних даних ітеративно беруться історичні – при цьому стратегія тестується на різних проміжках історії, гранулярності, тобто періоді свічок, різних активах, портфоліо, різних параметрах з метою визначення оптимальних.
4. Демо-тестинг – це етап, на якому стратегія з попереднього етапу тестується на демо-аккаунті в автоматизованому режимі. Зазвичай, робиться тестування за допомогою того ж брокеру та API, де і планується реальна сесія торгівлі, тому стратегія має бути повністю або більшою мірою реалізованою. За допомогою демо-аккаунту стратегія працює на звичайних потокових даних ринку, але при цьому витрати та прибутки є штучними. Також важливо розуміти, що ваші ордери з демо-аккаунту не будуть впливати на рух ціни як було б у випадку реального, але при цьому таке тестування є важливим через наступні причини – високі KPI стратегії на історичних даних ще не означають такі ж метрики на реальних даних, важливо перевірити реалізоване рішення на відсутність помилок чи неврахованих моментів на потокових даних, перевірка брокера, API та всієї системи на відповідність вимогам та цілям.
5. Деплоймент – запуск торгового робота в продакшені, тобто на реальному ринку.

В розділах 3.3-3.4 буде представлено реалізацію деяких з трейдинг стратегій з описом їх основних компонентів. Для кожної торгової стратегії дуже важливим компонентом є керування ризиками, тобто визначення розміру позицій, враховуючи наявний капітал, виставлення stop_loss, take_profit при відкритті торгових угод. Нижче в реалізаціях цей момент може бути упущений з опису, але тим не менш в повному коді це враховується.

3.3 Автоматизація стратегії Сітчата торгівля

Основними компонентами сітчатої торгівлі є сигнал трігер на початок торгівлі, початкова структура сітки, відкриття ордерів, апдейт сітки, сигнал трігер на зупинку, зупинка [22]. Розглянемо сітчату стратегію в спрощеному вигляді на прикладі (табл. 3.1).

Таблиця 3.1 – Сітчата торгівля на прикладі.

Ціна	Тип ордеру
1020$	Продаж
970$	Продаж
920$	Купівля
870$	Купівля
820$	Купівля
Крок 1	

Ціна	Тип ордеру
1020$	Продаж
970$	Продаж
920$	-
870$	Купівля
820$	Купівля
Крок 2	

Ціна	Тип ордеру
1020$	Продаж
970$	-
920$	Купівля
870$	Купівля
820$	Купівля
Крок 3	

Ціна	Тип ордеру
1020$	Продаж
970$	Продаж
920$	Продаж
870$	-
820$	Купівля
Крок 4	

Візьмемо, що початкове значення ціни активу 940$, на кроці 1 формуємо ентрі ордери на купівлю та продаж відносно початкової ціни – нехай рівнів буде 6 та крок 50$. На кроці 2 ціна падає до 920$, тоді виконується buy ордер на цьому рівні. На кроці 3 ціна піднімається до 970$ – виконується sell ордер та встановлюється новий buy ордер на рівні 920$. На кроці 4 ціна падає до 870$ – виконуються ордери на купівлю на рівнях 920$, 870$, встановлюється sell ордер на рівні 870$. І так далі.

Тепер розглянемо основні дії для автоматизації такої стратегії на розробленій в розділі 2 ATS – клас GridStrategy.

Параметрами стратегії є частота апдейтів (update_period), верхня та нижня межі сітки, ціна інтервалу, початкова ціна, кількість рівнів сітки та частота апдейтів. Багато з цих параметрів є опціональними, тому якщо не будуть передані, то будуть виведені з актуальних даних – оферів на ринку для base_price, ATR для interval_price.

```
class GridStrategy(BaseStrategy):
    def __init__(self, update_period : pd.Timedelta = None,
                 lower_price = None, upper_price = None,
                 interval_price = None, base_price = None,
                 grid_levels = 5, moving_grid = False,
                 **kwargs):
```

Як було зазначено в розділі 2.3, для визначення класу BaseStrategy, важливо визначити методи start_run, iter_run, end_run.

start_run відповідає за ініціалізацію початкової структури сітки – відправляються ліміт ордери на buy та sell симетрично з визначеним price_interval відносно base_price:

```
# Виведення base_price, interval_price, кількості levels, якщо необхідно
for level in range(self.last_level - 1, -1, -1):
    self._fill_level(level, is_buy=True)
for level in range(self.last_level + 1, len(self.grid)):
    self._fill_level(level, is_buy=False)
```

end_run відповідає за закриття відкритих та ще не заповнених позицій, тому залишається без змін з BaseStrategy.

iter_run робить перевірку статусів ордерів у визначеному порядку та на основі цього здійснює подальші дії по оновленню сітки, тобто створення нових buy/sell ордерів у протилежному напрямку, якщо ціна залишилася по очікуваному напрямку руху або у тому ж, якщо це був перетин рівня з подальшою зміною тренду:

```
if order.get_isBuy():
    if order.get_buy() < offer_price['sell']:
        self._fill_level(level+1, is_buy=False)
    else:
        self._fill_level(level, is_buy=True)
else:
    if order.get_sell() > offer_price['buy']:
        self._fill_level(level-1, is_buy=True)
    else:
        self._fill_level(level, is_buy=False)
```

З реалізації видно, що Сітчата торгівля в спрощеному вигляді не потребує технічних індикаторів для роботи, але їх можна використовувати для визначення автоматичних умов виходу з позицій – take_profit, stop_loss, чи визначення умов виходу зі стратегії, а також як сигнали для отримання більш чітких умов купівлі чи продажу. Визначивши клас GridStrategy описаним вище чином, ми зможемо алгоритмічно виконувати стратегію Сітчата торгівля – дивитися розділ 3.6.

3.4 Автоматизація стратегії Ренко + MACD

Розглянемо тепер автоматизацію стратегії, що базується суто на технічних індикаторах та визначає клас VectorizedStrategy, тобто трейдинг сигнали створюються на основі FrameClient, що оновлюється свічками потрібного періоду.

Одна з таких комбінує Ренко діаграму (рис. 3.3) та MACD індикатор (рис. 3.4). Для цієї стратегії варто вибирати активи з великою активністю, волатильністю та об’ємами. Значення “великою” буде залежати від торгуємих об’ємів, тобто розміру позиції, але більше торгових сигналів буде від активів, що часто змінюють тренди.

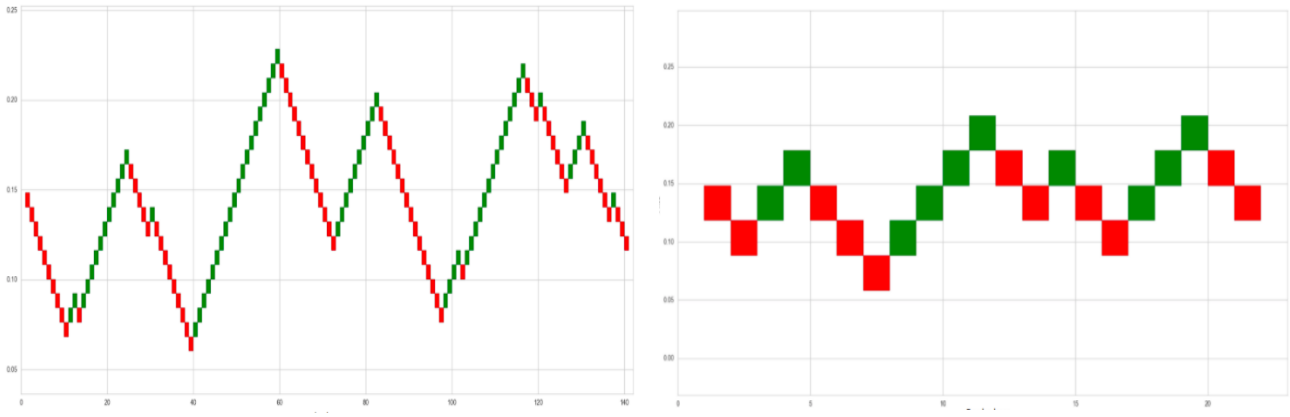

Рисунок 3.3 - Ренко діаграма(ціна/к-сть Ренко блоків) – розміри цегли 0.008 та 0.03 відповідно. [23]

Ренко діаграма [23] це тип фінансової діаграми, що використовується для аналізу рухів ціни активу з фільтрацією шумів, тобто незначних її коливань.

Замість свічкової діаграми ціна/час, ми отримуємо діаграму ціна/кількість Ренко блоків, де Ренко блок додається лише після руху ціни на вибрану величину та зображується під кутом 45 градусів (рис. 3.3) – рух може бути фіксованим або динамічним як ATR за певний період. Перевагою Ренко діаграми є розуміння ринків з великою волатильністю, а також більш наглядного представлення трендів та рівнів support/resistance.

Рисунок 3.4 – MACD індикатор [24]

MACD (Moving Average Convergence/Divergence) індикатор, що слідує за трендами та представляє сходження/розбіжність середніх ковзних (рис. 3.4). Рахується в модулі TechnicalIndicators наступним чином – спочатку беруться дві середні ковзні ціни (часто з періодами 12 та 26), потім береться середня ковзна їх різниці (часто з періодом 9).

```
df["macd_fast"] = df["close"].ewm(span=fast, min_periods=fast).mean()
df["macd_slow"] = df["close"].ewm(span=slow, min_periods=slow).mean()
df["macd"]= df["macd_fast"] - df["macd_slow"]
df["macd_signal"] = df["macd"].ewm(
      span=macd_period, min_periods=macd_period).mean()
```

Таким чином, якщо відбулася зміна напрямку тренду зі спадного на висхідний чи навпаки, то відбудеться перетин macd та macd_signal ліній, але їх перетин ще не гарантує зміну тренду. Якщо враховувати ще кути нахилу між macd та macd_signal, то можна програмно визначати, якому тренду слідує зараз ціна – crossover == 2 (висхідний) та crossover == -2 (спадний):

```
crossover = (np.sign(df["macd"][-1] - df["macd_signal"][-1]) +
      np.sign(df["macd_slope"][-1] - df["macd_signal_slope"][-1]))
```

Визначимо сигнали, які генеруватиме стратегія:

– Сигнал купівлі/продажу: кількість Ренко блоків у висхідному/спадному напрямку >= 2 та macd_crossover == 2/-2.
– Сигнал виходу з купівлі/продажу: macd_crossover змінив напрямок, тобто став -2/2 відповідно.

Окрім вищезазначених сигналів можна також додатково виставляти take_profit, stop_loss межі, коли позиція буде автоматично закрита. Варто зазначити, що технічні індикатори не гарантують руху ціни у відповідному напрямку, а є лише одним з показників, на які варто звертати увагу – тобто, можливі також помилкові сигнали.

З розділу 2, клас RenkoMacdStrategy визначає методи prepare_df(..., df), trade_signal(..., symbol, df, last_signal) абстрактного класу VectorizedStrategy.

prepare_df перетворює вхідний DataFrame свічок на Ренко діаграму з підрахунком підряд стоячих Ренко блоків(bar_num) та накладає поверх нього індикатори MACD, slope, ATR. Оскільки Renko діаграма замість часової шкали має кількість блоків, то це треба буде врахувати при реалізації для приведення даних до однієї шкали:

```
renko_df = indicators.RenkoDF(df)
df = df.merge(renko_df.loc[:,["date", "bar_num"]], how="outer", on="date")
df["bar_num"].fillna(method = 'ffill', inplace = True)
df.set_index('date', inplace=True)

# ... (Додавання MACD, ATR, slope)
return client.get_df().dropna()
```

trade_signal генерує один із сигналів описаних вище, якщо зазначені умови виконані, та створює Trade:

```
if last_signal is None:
    if bar_num >= 2 and crossover == 2: is_buy = True
    elif bar_num <= -2 and crossover == -2: is_buy = False
elif last_signal == "long":
    if bar_num <= -2 and crossover == -2: close, is_buy = True, False
    elif crossover == -2: close = True
elif last_signal == "short":
    if bar_num >= 2 and crossover == 2: close, is_buy = True, True
```

```
    elif crossover == 2: close = True
# ... (Створення Trade на закриття чи відкриття, враховуючи close, is_buy)
```

Приклади роботи RenkoMacdStrategy та GridStrategy буде показано в розділі 3.6. Тепер перейдемо до оцінки ефективності автоматизованих стратегій через бектестинг.

3.5 Бектестинг

Як уже було зазначено, на цьому етапі проводиться оцінка ефективності роботи стратегії на історичних даних з метою оптимізації та підбору її параметрів. Для прикладу, зробимо бектестинг стратегії, що є також реалізованою в рамках цієї роботи та використовує такі технічні індикатори як лінії Болліджера та MACD індикатор.

Для бектестингу не потрібен брокер, ринок торгівлі, капітал, він не має фактичних ризиків, бо здійснення ордерів відбувається локально у симуляційній манері на історичних даних – все, що потрібно, це дані та логіка стратегії. Для більш комплексного бектестингу заведено використовувати існуючі бібліотеки чи їх розширення.

Визначимо основні критерії, по яких виберемо задовільне рішення для бектестингу нашої стратегії:

1. Враховування комісій за здійснення торгових угод – бектестинг без комісій може суттєво змінити результати в кращу сторону, що потім призведе до меншої ефективності при реальному запуску;
2. Підрахунок основної статистики, тобто KPI;
3. Візуалізація роботи стратегії, тобто в який період були здійсненні ордери, щоб оцінити, за яких умов стратегія показує себе гірше/краще;
4. Доступ до відкритих позицій та мануальне їх закриття;
5. Виставлення параметрів для автоматичного закриття позицій – тобто take_profit, stop_loss, trailing_step;
6. Оптимізація параметрів роботи;

В даній роботі буде використовуватися пакет для Python Backtesting, що задовольняє визначені вище критерії. Для бектестингу опишемо клас BBandMacdSystem з методами init та next.

```
class BbandMacdSystem(Strategy):
    # Параметри стратегії
    def init(self):
        # Ініціалізація ...
    def next(self):
        # Викликається на кожній ітерації по історичних даних ...
```

Для бектестингу нам потрібні дані, і безкоштовно знайти провайдера інтрадей даних, тобто даних зі свічками менше 1-го дня за весь час торгівлі активом є досить складно – в межах даної роботи будемо використовувати dukascopy. Наведемо приклад бектестингу на акціях Ebay, параметри стратегії – капітал та розмір позиції в USD:

```
cash = 25000
pos_size = int(cash * 0.05)
```

В init зробимо ініціалізацію сигналів та графіків, які хочемо побачити.

```
def init(self):
    ...
    DF = indicators.MACD(DF)
    DF = indicators.BollingerBands(DF)
    self.data.df['macd_cross'] = (np.sign(DF["macd"] - DF["macd_signal"]) +
                                  np.sign(DF["macd_slope"] - DF["macd_signal_slope"
]))
    self.data.df['bbands'] = DF['bbands_percent']
    ...
```

В next робимо логіку по створенню ордерів на основі індикаторів:

```
def next(self):
    ...
    if atr > 0.2:
        if bbands_percent <= 0.6 and crossover == 2:
            self.buy(size=pos_size//price, tp=price+1.5*atr, sl=price-1.5*atr)
        elif bbands_percent >= 0.4 and crossover == -2:
            self.sell(size=pos_size//price, tp=price-1.5*atr, sl=price+1.5*atr)
```

В якості take_profit та stop_loss, будемо використовувати ATR індикатор, завантажуємо історичні дані з dukascopy для роботи тестування (рис. 3.5).

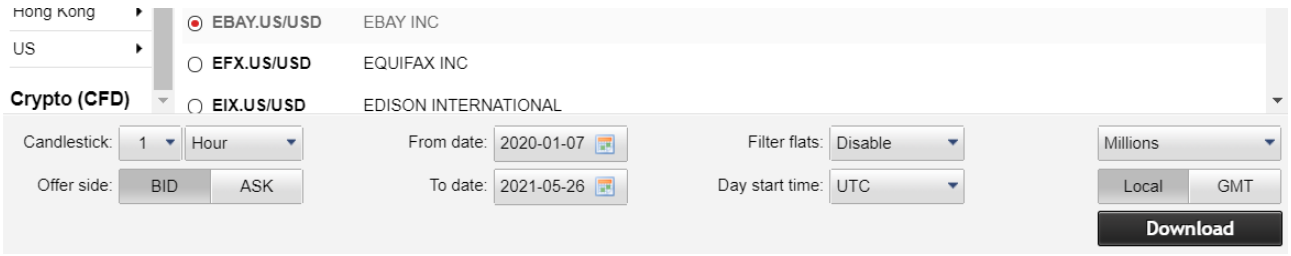

Рисунок 3.5 – Отримання даних для бектестингу.

Для прикладу, зобразимо поведінку стратегії за останні 1.5 роки на свічках з періодом 1 година. Прочитаємо їх в pandas.DataFrame, ініціюємо та виконаємо сам процес тестування:

```
df = pd.read_csv(fp, parse_dates = ["Local time"],
                 date_parser=lambda x: pd.to_datetime(x, utc=True,
dayfirst=True))
df = dukascopy_filter(df)
backtest = Backtest(df, BbandMacdSystem, cash=BbandMacdSystem.cash,
                    commission=.002, hedging=True)
print(backtest.run())
```

Після запуску стратегії на таких даних отримуємо біля нульового прибутку від такої стратегії – табл. 3.2:

Таблиця 3.2 – KPI стратегії Bollinger Bands та Macd індикатора

Start	2020-01-07 02:00...	Max. Drawdown Duration	433 days 16:00:00
End	2021-05-27 01:00...	Avg. Drawdown Duration	94 days 14:00:00
Duration	505 days 23:00:00	# Trades	175
Exposure Time [%]	8.475554	Win Rate [%]	43.428571
Equity Final [$]	24925.80158	Best Trade [%]	8.826397
Equity Peak [$]	25402.28531	Worst Trade [%]	-3.029983
Return [%]	-0.296794	Avg. Trade [%]	-0.079156
Buy & Hold Return [%]	71.714461	Max. Trade Duration	3 days 17:00:00
Return (Ann.) [%]	-0.213757	Avg. Trade Duration	0 days 19:00:00
Volatility (Ann.) [%]	1.085504	Profit Factor	0.903325
Sharpe Ratio	0	Expectancy [%]	-0.062564
Sortino Ratio	0	SQN	-0.409007
Calmar Ratio	0	_strategy	BbandMacdSystem
Max. Drawdown [%]	-2.132036	Max. Drawdown Duration	433 days 16:00:00
Avg. Drawdown [%]	-0.458092	Avg. Drawdown Duration	94 days 14:00:00

Протягом 505 днів було здійснено 175 торгових угод, 43.4% з яких були прибутковими, з 25000$ було отримано 24925. Як видно, з Equity Peak та Win Rate параметрів, були проміжки, на яких робот генерував прибуток, але через

примітивність цієї стратегії на довгих періодах роботи робот не отримав ніякого прибутку. Якщо взяти свічки з іншим часовим періодом/інші активи та розрідити дані бек-тестингу, то стратегія покаже себе гірше/краще, тобто при комбінації параметрів та додаванні trailing_step коефіцієнт WinRate можна довести до 50%, що все одно є недостатнім.

Метою надання такої неприбуткової статистики було показати, що саме торгова стратегія є тим, що перешкоджає стабільному доходу від торгового робота, тому при розробці варто фокусуватися на ній, бо прості чи загальновідомі рішення не будуть ефективними. Нижче представлено графічне представлення результатів бектестингу (рис. 3.6):

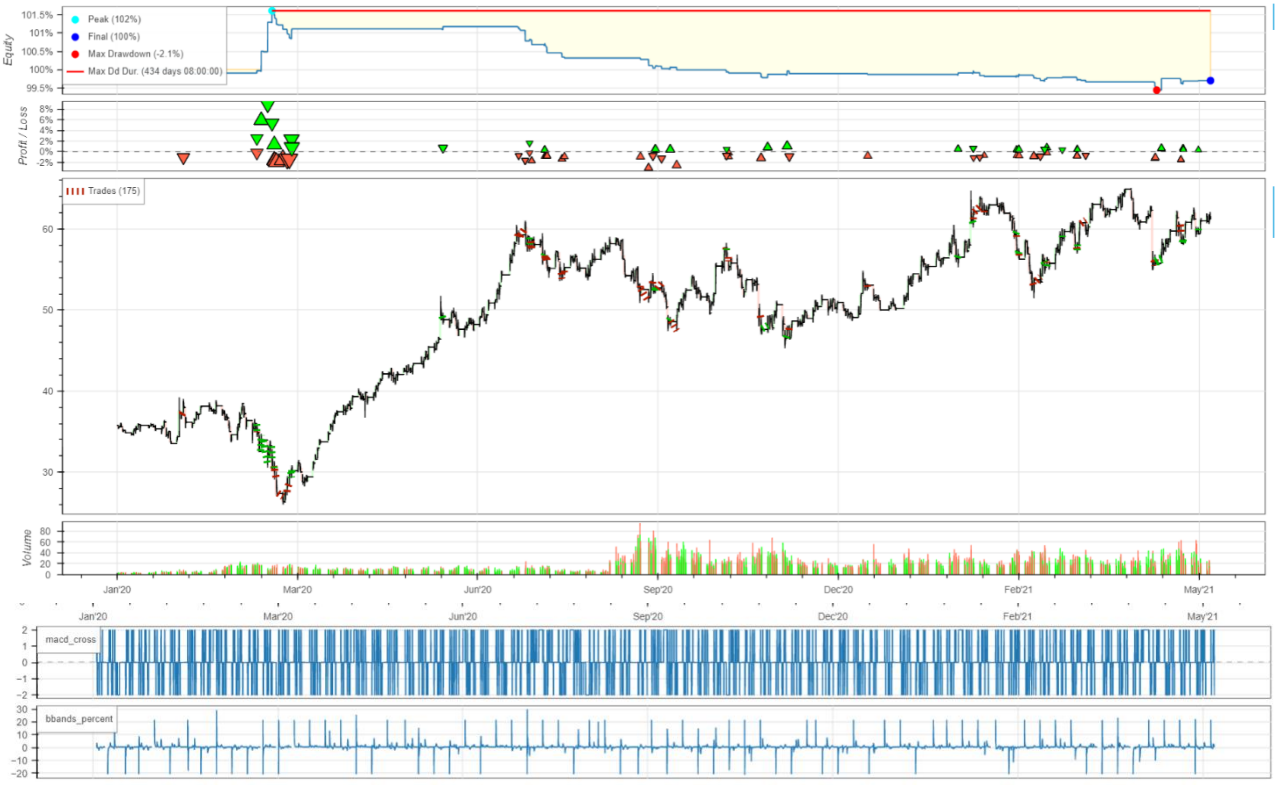

Рисунок 3.6 – KPI стратегії ліній Болліджера та індикатора MACD в графічному представленні.

Якщо розглянути графічне представлення результатів (рис. 3.6), то більш чітко буде видно, як саме торгував робот – спочатку в плюс, а потім в мінус. Графіки macd_cross та bbands_percent показують значення параметрів, з яких

формувалися buy/sell сигнали для основної стратегії. Трикутники вверх/вниз це buy/sell угоди та зелений/червоний кольори показують прибутки чи втрати.

Якщо зробити оцінку KPI на цих же даних на стратегії, що поєднує Ренко діаграму та індикатор MACD (рис 3.7), то WinRate так само не буде більше ніж 50%.

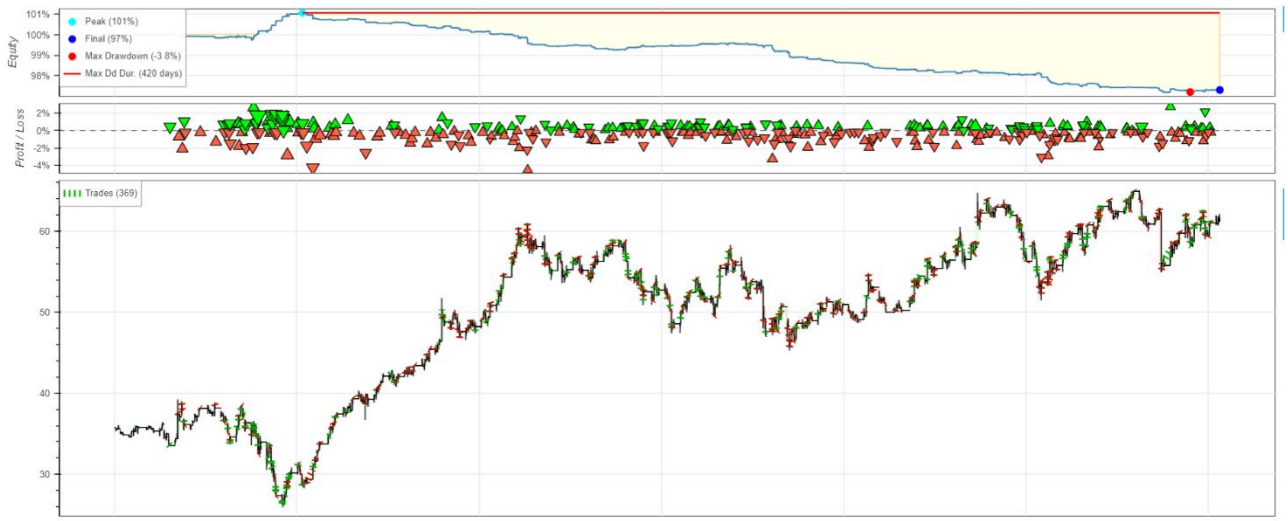

Рисунок 3.7 – Графічне представлення KPI стратегії на індикаторах Renko та MACD

Підводячи підсумки етапу бектестингу, хочеться зазначити, що цей процес є комплексним та триває не один день перед тим, як стратегія дасть необхідні показники для переходу на наступний етап тестування. На стратегіях, що в простому вигляді комбінують технічні індикатори, на довгих періодах прибутковим використання роботу не буде, а тому розробка ефективної стратегії є найскладнішим етапом.

3.6 Демо-тестинг та деплоймент

Наразі ми маємо автоматизовану трейдинг систему з трьома стратегіями – Ренко діаграма/лінії Болліндджера у поєднанні з індикатором MACD та Сітчата торгівля. Покажемо, як виглядає запуск двох з них на демо-сервері на реальних даних:

```
portfolio = bot.create_portfolio(['G
BP/JPY'], lot_size=15)
strategy = GridStrategy(
    robot = bot,
    portfolio= portfolio,
    run_for = pd.Timedelta(8, unit='
hours'),
    lower_price = 1.156,
    upper_price = 1.5346,
    grid_levels = 5,
    moving_grid=True,
    update_period = pd.Timedelta(60,
 unit='sec')
)
strategy.run()
```

```
portfolio = bot.create_portfolio(['E
UR/USD',
    'GBP/CAD'],
{'EUR/USD' : 15, 'GBP/CAD' : 12})
strategy = RenkoMacdStrategy(
    robot = bot,
    portfolio = portfolio,
    bars_period = 'm1',
    update_period = pd.Timedelta(95,
unit='sec'),
    init_bars_cnt = 300,
    trigger_frame_size = 300,
    run_for = pd.Timedelta(8, unit='
hours')
)
strategy.run()
```

Для обох стратегій перед кодом вище ми створюємо бота з конфігураційним файлом, як зазначено в розділі 2.4. Після цього стратегії необхідно надати портфоліо, що описує розміри позицій та символи для трейдингу – об’єкт portfolio відповідно. Стратегія запускається методом run() та при створенні приймає параметри роботи, спільним з яких є run_for, що є очікуваним часом роботи стратегії.

З логів роботи RenkoMacdStrategy (рис. 3.7) можна бачити приклад створення та через 13 хвилин закриття позиції по сигналам,

```
2021-05-26 03:12:35,013:run:81:INFO:FxRobot:Running iteration at 2021-05-26 00:12:35.013961+00:00
2021-05-26 03:12:35,015:iter_run:49:INFO:FxRobot:Processing GBP/CAD update
2021-05-26 03:12:35,016:get_last_bar:33:DEBUG:FxRobot:Querying last 1 bars for GBP/CAD
2021-05-26 03:12:35,238:iter_run:58:DEBUG:FxRobot:Data update happened for GBP/CAD
2021-05-26 03:12:36,725:update_trades:103:DEBUG:FxRobot:Method update_trades called for GBP/CAD with 263 items
2021-05-26 03:12:36,728:trade_signal:66:INFO:FxRobot:Close[False], Signal[Buy] found
2021-05-26 03:12:36,729:open_trade:69:INFO:FxRobot:Opening new trade for portfolio(2797605338464):
Trade({'order_id': None, 'symbol': 'GBP/CAD', 'amount': 120, 'rate': 0, 'is_buy': True, 'is_in_pips': True,
'session': 'GTC', 'order_type': 'AtMarket', 'stop': -10, 'trailing_step': 1})
2021-05-26 03:12:36,920:open_trade:77:INFO:FxRobot:Adding new trade(70532354) to the portfolio(2797605338464)
...
2021-05-26 03:24:35,004:run:81:INFO:FxRobot:Running iteration at 2021-05-26 00:24:35.004276+00:00
2021-05-26 03:24:35,005:iter_run:49:INFO:FxRobot:Processing GBP/CAD update
2021-05-26 03:24:35,006:get_last_bar:33:DEBUG:FxRobot:Querying last 1 bars for GBP/CAD
2021-05-26 03:24:35,216:iter_run:58:DEBUG:FxRobot:Data update happened for GBP/CAD
2021-05-26 03:24:36,271:update_trades:103:DEBUG:FxRobot:Method update_trades called for GBP/CAD with 263 items
2021-05-26 03:24:36,273:trade_signal:68:INFO:FxRobot:Close[True], Signal[Sell] found
2021-05-26 03:24:36,275:close_trade:83:INFO:FxRobot:Closing position(70532354) for currency(GBP/CAD) in the portfolio(2797605338464)
with args {'time_in_force': 'GTC', 'order_type': 'AtMarket', 'amount': 120}
2021-05-26 03:24:36,524:sleep_till_next_bar:51:DEBUG:FxRobot:Sleeping till next data update for 58.475716 seconds
```

Рисунок 3.7 – Робота автоматизованої торгової стратегії.

Варто також зазначити про необхідність та можливість у використанні хмарного деплоймента для виконання торгових стратегій:

– Необхідність: насправді, загальноприйняте рішення в трейдинг компаніях це використання надійних локальних серверів для хостингу, що забезпечують більший контроль та приватність у порівнянні з віддаленими. Локальний постачальник даних, бази даних, VPN, дешевша обчислювальна потужність є перевагами такого рішення;
– Можливість: AWS/Azure/Firebase/Heroku/Netlify – у випадку хмарного хостингу нам потрібно мати доступ та контроль над запущеним торговим роботом. Перевагами є можливість розташування такої віддаленої машини в локації, що є ближче до постачальника даних/брокера, наприклад в одному зі штатів USA, та мати менші затримки в мережевій комунікації. Найкраще серед безкоштовних рішень підходить AWS EC2, що надає протягом року роботи віддалену Linux машину в вибраній локації – підключення по SSH, запуск сервера через AWS консоль (рис. 3.8). Після підключення необхідно налаштувати середовище необхідне для запуску, тобто Python та всі необхідні залежності проекту.

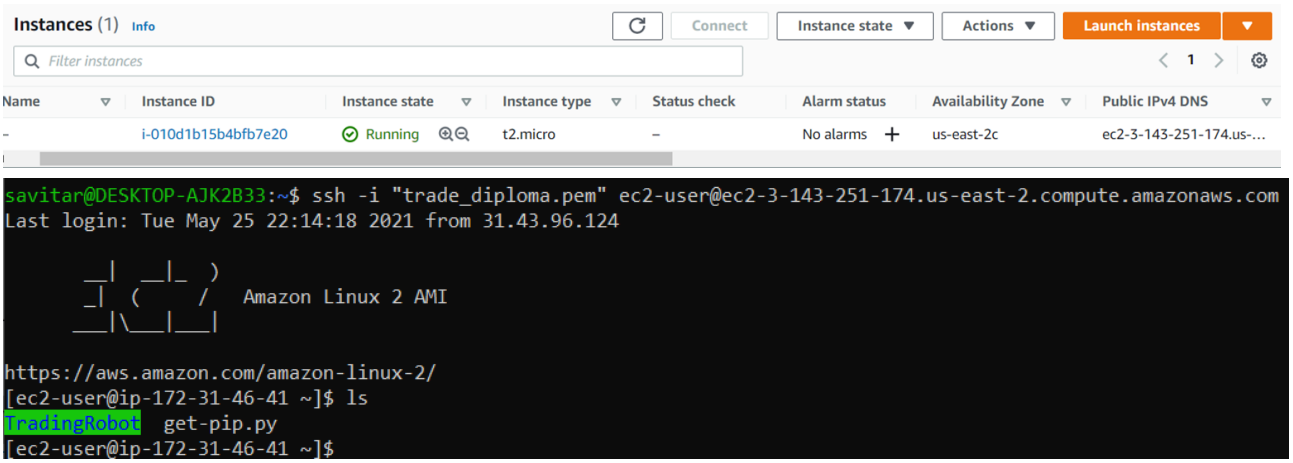

Рисунок 3.8 – Запуск та підключення до віддаленої AWS машини

ВИСНОВКИ

Таким чином у ході цієї роботи було розглянуто процес розробки автоматизованої трейдинг системи та створено прототип такої системи, використовуючи брокера FXCM, мову програмування Python. З використанням такої системи було автоматизованому стратегії з використанням технічних індикаторів – Ренко діаграми/ліній Боллінджера та MACD індикатора, а також стратегії на основі статистичної моделі – Сітчата торгівля. Від основ алгоритмічного трейдингу та визначення вимог до системи до розробки торгової системи та автоматизації торгових стратегій – кожен з цих процесів було описано в межах цієї роботи.

Матеріалів по автоматизації трейдингу, відкритих рішень по торговим роботам та стратегіям дуже мало, а прибуткових не мало б існувати, в принципі, бо їх розробникам просто невигідно ділитися доступом до необмежених прибутків. Ціллю даної роботи не було створити робота, що був би конкурентом до наявних у HFT компаній, а надати фундамент для розробників і дослідників, що починають свій шлях в алгоритмічній торгівлі та стикнулися з проблемою її вивчення, тобто зобразити на прикладі розробки, що собою являють торгові роботи, алгоритми, а також їх переваги над мануальним трейдингом, що і було зроблено.

Універсальних стабільних рішень, що дозволяли б покласти капітал на рахунок робота та лише примножувати його на постійній основі не існує через змінність ринку та відповідно необхідність модифікації алгоритму – стратегія, що була прибутковою ще вчора може перестати працювати вже сьогодні. Проте наразі найбільш ефективними є стратегії, що використовують моделі на основі машинного навчання та штучного інтелекту через вміння адаптуватися до змін ринку. За цим посиланням [30] можна знайти open-source ML моделі, що можуть використовуватися торговим роботом. Робота з біржами (крипто, фондові, валютні) це можливість заробити мільйони на інвестиціях чи трейдингу, але

одночасно й можливість стати банкрутом, бо біржова торгівля це zero-sum game – якщо якийсь учасник заробляє, то інший втрачає, адже ціни рухають учасники ринку, а прибутки формуються на різниці цін. Досить часто в інтернеті ми стикаємося з відеоматеріалами, де продають прибуткові сигнали чи торгові стратегії, але в певній мірі це скам, бо реальність в тому, що продавати сигнали (частина з яких може бути неуспішними) чи стратегії (які можуть бути й неприбутковими чи перестати працювати) значно вигідніше, аніж самим по них торгувати. Успішні трейдери чи торгові стратегії можуть показувати 80% WinRate на піку ефективності.

А все-таки отримання прибутку торговими роботами є можливим – прикладами можуть бути такі відомі компанії як Hudson River Trading, Jump One Trading, Jane Street, Squarepoint Capital, а також банки та багато інших непублічних/публічних хедж-фондів. Ще одним підтвердженням цього є графік зміни частки торгових угод, що здійснені в автоматизованій манері з часом (рисунок 1.1). Розробка прибуткового торгового робота це завдання не для одного розробника – загальноприйнятий процес автоматизації має вигляд Трейдер -> Data scientists/Quantitative researchers -> Програмісти, де перші приносять ідеї торгових стратегій чи наробітки по них, другі займаються оптимізаціями та тестуванням, а останні автоматизацією рішення. Стратегія, на прикладі якої був зображений бектестинг, виявилася неприбутковою на довгому періоді часу, що було очікувано, але це не означає, що прибуткових не існує, а лише стимулює проведення досліджень в цій області.

Продовження розробок у сфері розробки автоматизованих торгових систем на основі цієї роботи є актуальним та може бути здійснене в одному з наступних напрямків:

- Розробка прибуткової торгової стратегії з використанням штучного інтелекту;
- Оцінка величини впливу різних факторів (новин, сентименту, технічних) на зміни цін активу з метою розробки методик по виявленню стратегій;
- Розробка рішення для бектестингу торгових стратегій;

- Розробка high-speed інфраструктури для автоматизації трейдингу на одному чи декількох брокерах;
- Порівняння роботи трейдинг стратегій та торгового робота на різних ринках торгівлі/активах;
- Використання та оцінка ефективності наявних у відкритому доступі торгових моделей на основі машинного навчання з реалізацією на C++/C#;

Окрім цього, варто зазначити можливі оптимізації чи технології, що активно використовуються в сучасних ринкових торгових роботах:

- Мова програмування – C++/C#. Для трейдинг систем ефективність є дуже суттєвою, тому затримки навіть менше секунд є неприпустимими – через це вибирається сучасна мова програмування, що дозволяла б задовольнити таким вимогам. C# хоч і показує гірші результати по часу виконання, але є більш зручною у використанні;
- Трейдинг API – використання REST API робить суттєві для трейдингу затримки, тому його намагаються замінити більш ефективними рішеннями. Прикладами таких є Takion, Lightspeed, Sterling, PPro8 чи Interactive Brokers;
- Торгові стратегії – моделі на основі машинного навчання чи стратегії, що запрограмовані після визначення та оптимізації параметрів на основі бектестингу як тимчасові прибуткові рішення;

З одного боку перевагами алгоритмічного трейдингу є швидкість та надійність через відсутність людини, що проявляється у потенційно швидшому аналізі ринку, здійсненні угод, можливості автоматизації, надійності здійснення угод, але з іншого боку недоліком є саме відсутність людини через недостатність розвитку програмних систем на 2021 рік – моя думка, що з часом переваги будуть ставати все більш суттєвими. В Україні трейдинг як процес та професія ще не повністю урегульований, а компаній, що на цьому спецификуються досить мало, тому потенціал росту у сфері є великим.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Kravets F. How Algo trading works on exchanges - the essence, types and examples [Електронний ресурс] / Fyodor Kravets. – 2021. – Режим доступу до ресурсу: https://equity.today/algotrejding.html
2. Velu R. Algorithmic Trading and Quantitative Strategies / R. Velu, M. Hardy, D. Nehren., 2020. – 451 с. – (Taylor & Francis Group, LLC).
3. Davey K. Building Winning Algorithmic Trading Systems / Kevin Davey. – New Jersey, 2014. – 284 с. – (John Wiley & Sons, Inc., Hoboken).
4. Quantitative Trading: Algorithms, Analytics, Data, Models, Optimization / X.Guo, H. Shek, T. Lai, T. Wong., 2017. – 369 с. – (Taylor & Francis Group).
5. Kissell R. Algorithmic Trading Methods: Applications Using Advanced Statistics, Optimization, and Machine Learning Techniques / Robert Kissell., 2021. – 614 с. – (Elsevier).
6. Heitkoetter M. Why 90% of Traders Lose Money: Top Trading Mistakes to Avoid [Електронний ресурс] / Markus Heitkoetter. – 2021. – Режим доступу до ресурсу: https://www.moneyshow.com/articles/tebiwkly08-56338/.
7. Connors L. Short Term Trading Strategies That Work / L. Connors, C. Alvarez., 2008. – 122 с. – (TradingMarkets).
8. FIX Implementation Guide [Електронний ресурс] – Режим доступу до ресурсу: https://www.fixtrading.org/implementation-guide.
9. Grossbard J. Forex Trading Statistics [Електронний ресурс] / Justin Grossbard. – 2021. – Режим доступу до ресурсу: https://www.compareforexbrokers.com/forex-trading/statistics/.
10. Cryptocurrency Market Size, Share and COVID-19 Impact Industry Analysis [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.fortunebusinessinsights.com/industry-reports/cryptocurrency-market-100149.

11. Largest stock exchange operators worldwide as of February 2021, by market capitalization of listed companies. // Statista Research Department. – 2021.
12. Margin Rules for Day Trading [Електронний ресурс]. – 2011. – Режим доступу до ресурсу: https://www.sec.gov/files/daytrading.pdf.
13. Остапів Л. Податки з іноземних акцій та ETF [Електронний ресурс] / Любомир Остапів. – 2019. – Режим доступу до ресурсу: https://simeinyi-budzhet.ua/personaltaxes/податки-з-іноземних-акцій-та-etf/.
14. Hagmann A. Algorithmic Trading A-Z with Python, Machine Learning [Електронний ресурс] / Alexander Hagmann. – 2021. – Режим доступу до ресурсу: https://www.udemy.com/course/algorithmic-trading-with-python-and-machine-learning/.
15. Bulkowski T. Encyclopedia of Chart Patterns / Thomas Bulkowski., 2021. – 1312 с. – (Wiley Trading).
16. Beginners' guide to fundamental analysis [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://www.ig.com/en/trading-strategies/beginners-guide-to-fundamental-analysis-190503.
17. FXCM Socket REST API User Guide [Електронний ресурс] // FXCM Group – Режим доступу до ресурсу: https://apiwiki.fxcorporate.com/api/RestAPI/Socket%20REST%20API%20Specs.pdf.
18. The Encyclopedia of Algorithmic and Quantitative Trading Strategies [Електронний ресурс] – Режим доступу до ресурсу: https://quantpedia.com/.
19. Forex Robots [Електронний ресурс] – Режим доступу до ресурсу: https://www.forexrobottrader.com/forex-robots/.
20. Rasu M. Algorithmic Trading & Quantitative Analysis Using Python [Електронний ресурс] / Mayank Rasu. – 2021. – Режим доступу до ресурсу: Algorithmic Trading & Quantitative Analysis Using Python.
21. Majaski C. Paper Trade [Електронний ресурс] / Christina Majaski. – 2020. – Режим доступу до ресурсу: https://www.investopedia.com/terms/p/papertrade.asp.

22. What is Grid Trading? [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://www.binance.com/en/support/faq/f4c453bab89648beb722aa26634120c3.
23. Malchevskiy S. Renko Brick Size Optimization [Електронний ресурс] / Sergey Malchevskiy. – 2018. – Режим доступу до ресурсу: https://towardsdatascience.com/renko-brick-size-optimization-34d64400f60e.
24. Fernando J. Moving Average Convergence Divergence (MACD) [Електронний ресурс] / Jason Fernando. – 2021. – Режим доступу до ресурсу: https://www.investopedia.com/terms/m/macd.asp.
25. Technical Analysis Series — Article #3: Introduction to Pattern Trading [Електронний ресурс] // Junior Economist. – 2019. – Режим доступу до ресурсу: https://medium.com/@je_12591/technical-analysis-series-article-3-introduction-to-pattern-trading-71febc1f5bec.
26. Trend trading vs counter trend trading [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://fbs.com/analytics/tips/trend-trading-vs-counter-trend-trading-12689.
27. Bennett J. The Forex Breakout Strategy You Need to Master in 2020 [Електронний ресурс] / Justin Bennett. – 2019. – Режим доступу до ресурсу: https://dailypriceaction.com/blog/forex-breakout-strategy/.
28. Mitchell C. Fibonacci Retracement Levels [Електронний ресурс] / Cory Mitchell. – 2021. – Режим доступу до ресурсу: https://www.investopedia.com/terms/f/fibonacciretracement.asp.
29. 10 Pros and Cons of Algo Trading [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://therobusttrader.com/pros-and-cons-of-algo-trading/.
30. Stock-Prediction-Models [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/huseinzol05/Stock-Prediction-Models.