МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ

Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ФІНАНСОВОГО ТЕХНІЧНОГО АНАЛІЗУ»**

Виконала: студентка 2 курсу, групи 8.1219

спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

освітньої програми 121 інженерія програмного забезпечення
(назва освітньої програми)

К. М. Гурєєва
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, к.ф.-м.н. Кудін О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри загальної математики, доцент, к.ф.-м.н. Стєганцев Є.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2020

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення

Освітня програма інженерія програмного забезпечення

(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.

(підпис)

« ____ » _____ 2020 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ (СТУДЕНТЦІ)

Гурєєвій Катерині Миколаївні

(прізвище, ім'я та по батькові)

1. Тема роботи Розробка програмного забезпечення фінансового технічного аналізу

керівник роботи Кудін Олексій Володимирович, к.ф.-м.н.

(прізвище, ім'я та по батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » травень 2020 року № 576-с

2. Строк подання студентом роботи 30.11.2020

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Постановка задачі.
2. Основні теоретичні відомості.
3. Реалізація програмного забезпечення
4. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	15.09.2020	Виконано
2.	Збір вихідних даних.	20.09.2020	Виконано
3.	Обробка методичних та теоретичних	29.10.2020	Виконано
	джерел.		
4.	Розробка першого та другого розділу.	07.11.2020	Виконано
5.	Розробка третього розділу.	21.11.2020	Виконано
6.	Оформлення та нормоконтроль	25.11.2020	Виконано
	кваліфікаційної роботи.		
7.	Захист кваліфікаційної роботи.	15.12.2020	Виконано

Студент _____ К.М. Гурєєва
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Кудін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О. В. Кудін
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Розробка програмного забезпечення фінансового технічного аналізу»: 65 с., 39 рис., 12 джерел.

АЛГОРИТМІЗАЦІЯ, ТОРГОВА СТРАТЕГІЯ, ТЕХНІЧНІ ІНДИКАТОРИ, ТЕХНІЧНИЙ АНАЛІЗ.

Об’єкт дослідження – процес обробки та прогнозування фінансових часових рядів.

Мета роботи: розробити бібліотеку прогнозування фінансових часових рядів за допомогою технічних індикаторів.

Методи дослідження – аналітичний, об’єктно-орієнтований аналіз, методи програмної інженерії.

В кваліфікаційній роботі наводиться аналітичний огляд ключових понять алгоритмічної торгівлі. Розглянуто компоненти торгової стратегії, фактори, що впливають на торги, теоретичну складову технічних індикаторів. Виконано проектування бібліотеки фінансового технічного аналізу. Реалізовано інструменти для завантаження, обробки і проведення обчислень над фінансовими даними.

SUMMARY

Master’s qualification thesis «Development of the Software for Financial Technical Analysis»: 65 pages, 39 figures, 12 references.

ALGORITHMIZATION, TRADING STRATEGY, TECHNICAL INDICATORS, TECHNICAL ANALYSIS.

The object of the study is the financial time series forecasting process.

The aim of the study is to develop a library for forecasting financial time series using technical indicators.

The methods of research are analytical, object-oriented analysis, methods of software engineering.

The qualifying paper provides an analytical overview of key concepts of algorithmic trading. The components of trading strategy, factors influencing trading, the theoretical component of technical indicators are considered. The design of the library of financial technical analysis is performed. Implemented tools for downloading, processing and performing calculations on financial data.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

PnL	Прибуток та втрати (англ. Profit and Loss)
FIFO	Перший зайшов, перший вийшов (англ. First In First Out)
SMA	Simple Moving Average
EMA	Exponential Moving Average
MACD	Moving Average Convergence Divergence
BBANDS	Bollinger Bands
RSI	Relative Strength Indicator
STDDEV	Standard Deviation
MOM	Momentum

ВСТУП

Алгоритмічна торгівля, або автоматизована торгівля, працює з програмою, яка містить набір інструкцій для торгових цілей. Порівняно з людиною-торговцем, ця торгівля може приносити прибуток і збитки з більшою швидкістю.

Об’єктом дослідження є процеси обробки та прогнозування фінансових часових рядів.

Предмет дослідження – методи прогнозування фінансових часових рядів за допомогою математичних функцій.

Метою кваліфікаційної роботи є розробка бібліотеки прогнозування фінансових часових рядів за допомогою технічних індикаторів.

Для досягнення поставленої мети сформульовано наступні задачі:

а) провести аналітичний огляд факторів та технічних індикаторів впливаючих на прогнозування фінансових часових рядів;

б) виконати проектування та розробку бібліотеки прогнозування часових рядів;

в) виконати тестування програмного засобу та провести ряд обчислювальних експериментів.

Новизна роботи полягає у застосуванні технічних індикаторів та різноманітних торгових стратегій з метою обробки фінансових даних та прогнозування подальшої поведінки графіку ціни.

1 ОГЛЯД ТЕОРЕТИЧНИХ ПОЛОЖЕНЬ

1.1 Ключові поняття алгоритмічної торгівлі

Історично торговці застосовували таку торгівлю на основі правил, щоб вручну вводити замовлення, займати позиції та отримувати прибуток або збитки протягом дня. З часом, з прогресом в технології, вони перейшли від крику у відділі біржі, щоб виконувати замовлення з іншими трейдерами, викликати брокера та вводити замовлення по телефону, зараз сучасні трейдери мають графічний інтерфейс програм, що дозволяє вводити замовлення через зручні інтерфейси.

Ручні підходи мають масу недоліків – люди повільно реагують на ринки, вони пропускають або повільно реагують на нову інформацію, вони не можуть добре масштабуватись або фокусуватись у декількох справах одночасно. Люди схильні робити помилки, вони відволікаються і відчувають страх втрати та радість заробітку. Усі ці недоліки можуть стати причиною відхилення від запланованої торгової стратегії, суворо обмежуючи її прибутковість.

Комп’ютери надзвичайно добре справляються з повторюваними завданнями на основі правил. Правильно запрограмовані, вони можуть надзвичайно швидко виконувати інструкції та алгоритми, і їх можна легко масштабувати та застосовувати на багатьох інструментах. Вони надзвичайно швидко реагують на ринкові дані, не відволікаються і не помиляються. У них немає емоцій, тому вони не відступають від запрограмованого алгоритму. Усі ці переваги роблять автоматизовані торгові системи вигідними, саме тут починається алгоритмічна торгівля.

Алгоритмічна торгівля – це процес виконання замовлень з використанням автоматизованих та запрограмованих торгових інструкцій з урахуванням таких змінних, як ціна, час та обсяг. Алгоритмічна торгівля використовує складні формули в поєднанні з математичними моделями та людським наглядом для прийняття рішень про купівлю або продаж фінансових цінних паперів на біржі. Алгоритмічні торговці часто використовують високочастотну технологію торгівлі, яка може дозволити

фірмі здійснювати десятки тисяч торгів у секунду. Алгоритмічна торгівля може бути використана в самих різних ситуаціях, включаючи виконання замовлень, арбітраж та стратегію торгівлі трендами [4].

Алгоритмічна торгівля в основному використовується інституційними інвесторами та великими брокерськими компаніями для скорочення витрат, пов’язаних з торгівлею. Згідно з дослідженнями, алгоритмічна торгівля особливо вигідна для великих розмірів замовлень, які можуть складати до 10% загального обсягу торгів. Зазвичай маркетологи використовують алгоритмічні торгові угоди для створення ліквідності [2].

Також, алгоритмічна торгівля дозволяє швидше і простіше виконувати замовлення, що робить її привабливою для бірж. У свою чергу, це означає, що трейдери та інвестори можуть швидко резервувати прибуток від невеликих змін ціни. У торговій стратегії скальпінгу зазвичай використовуються алгоритми, оскільки вона передбачає швидку купівлю та продаж цінних паперів з невеликим приростом ціни [6].

Швидкість виконання замовлення, що є перевагою за звичайних обставин, може стати проблемою, коли кілька замовлень виконуються одночасно без втручання людини. Ще одним недоліком алгоритмічних угод є те, що ліквідність, яка створюється за допомогою швидких замовлень на покупку і продаж, може зникнути в одну мить, що виключає можливість отримання трейдерами прибутку від зміни цін. Це також може призвести до миттєвої втрати ліквідності. Дослідження виявили, що алгоритмічна торгівля була основним фактором, що спричиняє втрату ліквідності на валютних ринках після того, як швейцарський франк припинив свою прив'язку до євро в 2015 році [2, 4].

Сьогодні велика частина торгівлі здійснюється в електронному вигляді за допомогою різних програмних додатків.

Обробники каналів ринкових даних (програмні додатки, які учасники ринку створюють з метою взаємодії з конкретним протоколом біржових ринкових даних) обробляють і розуміють ринкові дані, поширювані торговими біржами, щоб відображати справжній стан лімітної книги і ринкових цін (заявок і пропозицій).

Ринкові дані публікуються в спеціальному протоколі ринкових даних, попередньо узгодженому між біржею і учасниками ринку (FIX / FAST, ITCH і HSVF). Потім ті ж програмні додатки можуть передавати цю інформацію людям або самостійно приймати рішення алгоритмічно. Ці рішення потім знову передаються на біржу за допомогою аналогічного програмного додатка (шлюзів введення замовлень), який інформує біржу про зацікавленість в конкретному продукті і про зацікавленість в купівлі або продажу цього продукту за певними цінами, відправляючи певні типи замовлень (GTD, GTC, IOC і т. Д.). Це включає розуміння і взаємодія з біржею в протоколі введення замовлень на обмін, попередньо узгодженому між біржею і учасниками біржі (FIX, OMEX, OUCH) [1].

Після збігу з доступними учасниками ринку, цей збіг знову передається назад в програмний додаток через шлюзи введення замовлень і ретранслюється назад в торговий алгоритм або людей, тим самим завершуючи транзакцію, часто повністю в електронному вигляді. Швидкість цього прийому-передачі сильно залежить від ринку, учасників та самих алгоритмів.

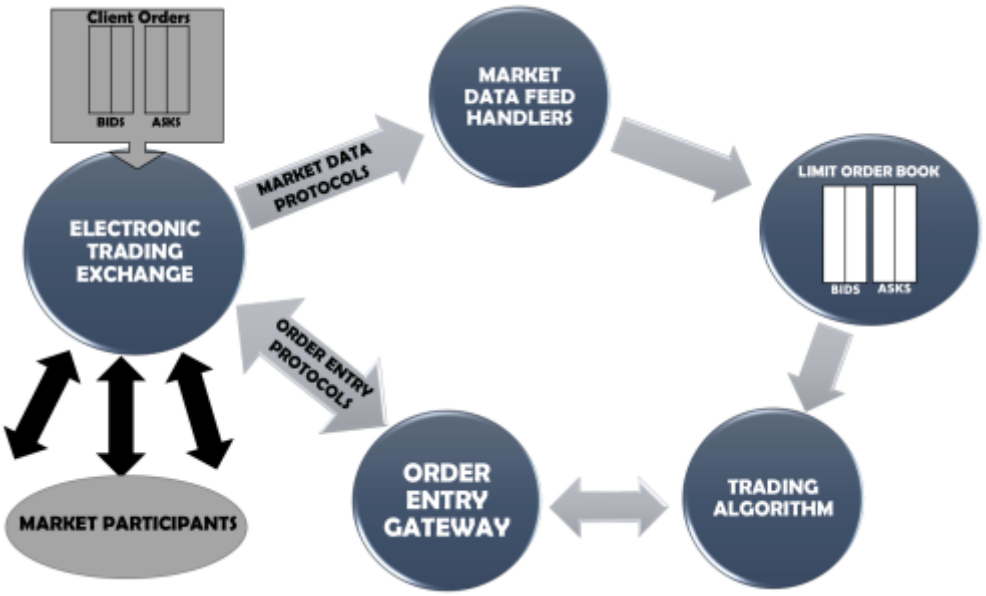

Рисунок 1.1 – Схема передачі даних у торговій системі

Як показано на попередній діаграмі (див. рис. 1.1), торгова біржа веде журнал клієнтських замовлень на покупку (бидов) і клієнтських запитів на покупку (запитує), а також публікує ринкові дані з використанням протоколів ринкових даних, щоб надати стан книги всім учасникам ринку. Обробники потоку ринкових даних на стороні клієнта декодують вхідний потік ринкових даних і створюють на своєму боці книгу лімітних заявок, щоб відображати стан книги заявок, яке бачить біржа. Потім він поширюється через торговий алгоритм клієнта, а потім проходить через шлюз введення замовлень для генерації вихідного потоку замовлень. Вихідний потік замовлень передається на біржу через протоколи введення замовлень. Це, в свою чергу, створить додатковий потік ринкових даних, і, отже, цикл торгової інформації продовжиться [1, 3].

Виконані замовлення призводять до того, що учасники ринку мають позиції в інструменті, який вони були виконані, на суму виконаного замовлення і за ціною виконання (лімітні замовленні можуть відповідати більш вигідними цінами, ніж вони були введені, але не гірше). Виконання на стороні покупки називається довгою позицією, а виконання на стороні продажу – короткою позицією. Коли у нас взагалі немає позиції, це називається флет. Довгі позиції приносять прибуток, коли ринкові ціни вище, ніж ціна позиції, і втрачають гроші, коли ринкові ціни нижчі, ніж ціна позиції. Короткі позиції, навпаки, заробляють гроші, коли ринкові ціни знижуються в порівнянні з ціною позиції, і втрачають гроші, коли ринкові ціни підвищуються в порівнянні з ціною позиції, звідси і добре відомі ідеї купувати за низькою ціною, продавати за високою і купувати за високою, продавати дорожче, і так далі [2, 3].

1.2 Компоненти стратегії

Повна система алгоритмічної торгівлі розділена на два розділи, як показано на наступній схемі (див. рис. 1.2):

- основна інфраструктура займається інтеграцією протоколів ринкових даних, обробниками ринкових даних, нормалізацією формату даних внутрішнього ринку, історичними записами даних, визначенням та розповсюдженням визначень приладів, протоколами введення біржових замовленнь, шлюзовими шлюзами, основними системами побічних ризиків, брокерськими заявки на виправлення, заявки на узгодження бек-офісу, задоволення вимог відповідності та інші;

- компоненти алгоритмічної торгової стратегії стосуються використання нормалізованих ринкових даних, побудови книг замовлень, генерування сигналів з вхідних ринкових даних та інформації про потоки замовлень, агрегування різних сигналів, ефективної логіки виконання, побудованої на основі статистичних прогнозних здібностей (альфа), управління позицією та PnL усередині стратегій, управління ризиками всередині стратегій, бектестування та історичні платформи досліджень та торгівлі [1, 4].

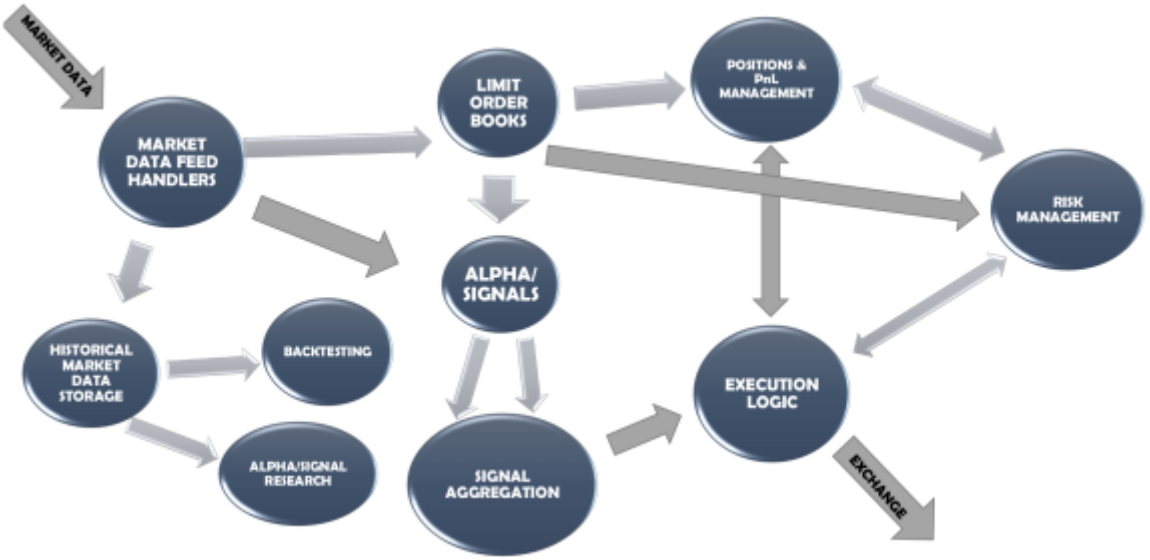

Рисунок 1.2 – Компоненти алгоритмічної торгової стратегії

Компоненти підписки на ринкові дані відповідають за взаємодію з компонентами обробника каналів, які публікують нормалізовані дані. Ці дані можуть бути доставлені по мережі або локально з використанням різних механізмів взаємодії між процесами (IPC) від обробників каналів.

Як тільки торгова стратегія отримує нормалізовані ринкові дані, вона використовує ці дані для створення і підтримки книги лімітних замовлень для кожного інструменту. Залежно від складності і складності книги лімітних замовленнь, вона може бути досить простою, щоб повідомляти нам, скільки учасників на кожній стороні, або досить складною, щоб відстежувати пріоритети учасників ринку, а також відстежувати наші власні замовлення в книзі лімітних замовленнь.

Після створення книги лімітних замовленнь, кожен раз, коли вона оновлюється в зв'язку з надходженням нової інформації про ринок, ми формуємо сигнали, використовуючи нову інформацію.

Сигнали називаються різними іменами – сигнали, індикатори, предиктори, калькулятори, функції, альфа і т. д. – але всі вони мають приблизно одне і те ж значення. Торговий сигнал – це чітко визначена інтелектуальна інформація, отримана з вхідної інформації про ринок, книги лімітних замовлень або торговельної інформації, яка дозволяє торгової стратегії отримати статистичну перевагу перед іншими учасниками ринку і, отже, підвищена рентабельність. Це одна з областей, на яку багато торгових команди приділяють багато часу і енергії. Ключ полягає в тому, щоб створити безліч сигналів, щоб мати перевагу над конкурентами, а також продовжувати адаптувати існуючі сигнали і додавати нові сигнали для обробки [7].

Часто велика кількість алгоритмічних торгових систем комбінують безліч різних сигналів, щоб отримати більшу перевагу, ніж дають окремі сигнали. Підхід полягає в тому, щоб по суті комбінувати сигнали, що мають різні передбачувані здатності/переваги в різних ринкових умовах. Є багато різних способів комбінувати окремі сигнали. Можна використовувати класичні методи статистичного навчання для генерації лінійних і нелінійних комбінацій для виведення значень класифікації або регресії, які являють собою комбінацію окремих сигналів.

Інший ключовий компонент алгоритмічної торгівлі – це швидке і ефективне управління замовленнями на основі сигналів з метою отримання переваги над конкурентами. Важливо швидко, але розумно реагувати на зміну ринкових даних, зміна значень сигналів. Часто швидкість і складність – дві конкуруючі цілі, і хороша логіка виконання намагатиметься збалансувати ці дві мети оптимальним чином. Також надзвичайно важливо приховувати свої наміри/інформацію від інших учасників ринку, щоб отримати найкраще можливе виконання [8].

Інші ринкові конкуренти можуть спостерігати, які замовлення відправляються на біржу, і оцінювати потенційний вплив, яке вони можуть надати, тому цей компонент повинен бути достатньо розумним, щоб не робити очевидним, що робить наша торгова стратегія. Прослизання і комісії також є дуже важливими факторами з точки зору розробки логіки виконання.

Прослизання визначається як різниця між очікуваною ціною угоди і ціною, по якій угода фактично виконується. Це може відбуватися переважно з двох причин [8]:

- якщо замовлення надходить на біржу пізніше, ніж очікувалося (затримка), то він може або взагалі не виконуватися, або виконуватися за нижчою ціною, ніж ви могли очікувати;
- якщо замовлення дуже великий і виконується за кількома цінами, тоді VWAP всього виконання може значно відрізнятися від ринкової ціни, що спостерігається при відправці замовлення.

Безсумнівно, прослизання призводять до збитків, які могли бути неправильно враховані, на додаток до труднощів при ліквідації позицій. У міру збільшення розмірів позицій для торгових алгоритмів прослизання стає більш серйозною проблемою.

Комісії – ще одна проблема, пов'язана з ефективним виконанням замовлень. Як правило, існують комісії за обмін і брокерські збори, пропорційні розміру замовленнь і загального обсягу торгів.

Знову ж таки, оскільки розміри позицій для торгових алгоритмів збільшуються, обсяги торгівлі зазвичай збільшуються, а разом з ними збільшуються

і комісії. Часто хороша торгова стратегія може виявитися неприбутковою, тому що вона занадто багато торгується і накопичує великі торгові комісії.

Хороша логіка виконання спрямована на мінімізацію сплачених комісій.

Усі алгоритмічні торгові стратегії повинні ефективно відстежувати свої позиції і PnL і управляти ними. Залежно від реальної торгової стратегії, вона часто може варіюватися по складності.

Для більш складних торгових стратегій, таких як парна торгівля (торгівля по кривій – ще одна аналогічна стратегія), необхідно відстежувати позиції і PnL за кількома інструментами, і часто ці позиції і PnL компенсують один одного і вносять складність/невизначеність щодо визначення дійсних позицій і PnLs [11].

Раціональне управління ризиками – один з наріжних каменів алгоритмічної торгівлі. Погана практика управління ризиками може перетворити потенційно прибуткові стратегії в неприбуткові. Існує ще більший ризик порушення правил і положень на торгових біржах, що часто може призвести до судових позовів і великих штрафів. Нарешті, один з найбільших ризиків високошвидкісний автоматичної алгоритмічної торгівлі полягає в тому, що погано запрограмоване комп'ютерне програмне забезпечення схильний до помилок і помилок. Отже, системи управління ризиками повинні бути надзвичайно надійними, функціональними та мати кілька рівнів надмірності. Також необхідний дуже високий рівень тестування, стрес-тестування і суворого управління змінами, щоб звести до мінімуму можливість відмови систем ризику [9].

При дослідженні автоматичної торгової стратегії на предмет очікуваної поведінки ключовим компонентом гарної системи дослідження алгоритмічної торгівлі є хороший бектестер. Бектестер використовується для моделювання поведінки автоматичної торгової стратегії і отримання статистики очікуваних прибутків і збитків, очікуваного ризику та інших показників на основі історично зареєстрованих ринкових даних. Це досягається за рахунок точної реєстрації історичних ринкових даних, наявності структури для її відтворення, наявності структури, яка може приймати модельований потік замовлень з потенційних торгових стратегій, і імітації того, як торгова біржа буде відповідати потоку

замовлень цієї стратегії в присутності інших учасників ринку. як зазначено в історичних ринкових даних. Це також гарне місце, щоб випробувати різні торгові стратегії, щоб побачити, які ідеї працюють, перш ніж впроваджувати їх на ринок. Створення та підтримка високоточного бектестера – одна з найскладніших завдань, пов'язаних з налаштуванням системи дослідження алгоритмічної торгівлі. Він повинен точно моделювати такі речі, як затримки програмного забезпечення, затримки в мережі, точні пріоритети FIFO для замовлень, прослизання, комісії і, в деяких випадках, також вплив на ринок, викликане потоком замовлень для розглянутої стратегії (тобто, як інші учасники ринку можуть відреагувати на наявність потоку замовлень і торгової активності цієї стратегії) [1, 2, 7].

1.3 Фактори

Алгоритмічні торгові стратегії засновані на сигналах, які вказують, коли купувати або продавати активи, щоб отримати позитивну прибуток в порівнянні з еталоном. Та частина прибутковості активу, який не пояснюється впливом еталонного показника, називається альфа-коефіцієнтом, тому ці сигнали також називаються альфа-факторами.

Альфа-фактори призначені для прогнозування руху цін на активи в інвестиційному середовищі на основі наявних ринкових, фундаментальних або альтернативних даних. Фактор може об'єднувати одну або кілька вхідних змінних, але приймає одне значення для кожного активу кожен раз, коли стратегія оцінює фактор. Торговельні рішення зазвичай ґрунтуються на відносній вартості активів. Торгові стратегії часто засновані на сигналах, що виходять з безлічі факторів, і очевидно, що моделі машинного навчання (ML) особливо добре підходять для ефективної інтеграції різних сигналів для більш точних прогнозів [9].

Альфа-фактори – це перетворення ринкових, фундаментальних і альтернативних даних, які містять прогнозні сигнали. Вони призначені для уловлювання ризиків, що впливають на прибутковість активів. Один набір факторів

описує фундаментальні змінні в масштабах всієї економіки, такі як зростання, інфляція, нестабільність, продуктивність і демографічний ризик. Інший набір складається з торгованих інвестиційних стилів, таких як ринковий портфель, інвестування з метою зростання вартості і імпульсне інвестування [10].

Існують також чинники, які пояснюють рух цін на основі економічних або інституційних умов фінансових ринків, або поведінки інвесторів, включаючи відомі упередження в цьому поведінці. Економічна теорія, що лежить в основі факторів, може бути раціональною, коли фактори мають високу прибутковість в довгостроковій перспективі, щоб компенсувати їх низьку прибутковість в погані часи, або поведінкової, коли премії за факторний ризик є результатом можливо упередженої або не зовсім раціональної поведінки агентів, яка не підлягає арбітражу [11].

Перетворення даних включають в себе просту арифметику, таку як абсолютні або відносні зміни змінної в часі, співвідношення між рядами даних або агрегування протягом тимчасового вікна, наприклад, просте або експоненціальне ковзне середнє. Вони також включають розрахунки, засновані на технічному аналізі цінових моделей, таких як індекс відносної сили попиту і пропозиції, і численні показники, відомі з фундаментального аналізу цінних паперів.

Фактори ризику імпульсу призначені для довгих активів, які показали хороші результати, і коротких активів з поганою продуктивністю протягом певного періоду.

Передумова стратегій, заснованих на цьому факторі, полягає в тому, що ціни на активи демонструють тенденцію, відтворену в позитивних серійних кореляціях. Такий ціновий імпульс кинув би виклик гіпотезі про ефективні ринки, в якій мовиться, що тільки минулі прибутковості цін не можуть передбачити майбутні результати [5].

Незважаючи на теоретичні аргументи про зворотне, стратегії цінового імпульсу принесли позитивну прибутковість за класами активів і є важливою частиною багатьох торгових стратегій.

Фактори імпульсу зазвичай виводяться зі змін у тимчасових рядах цін шляхом виявлення тенденцій і закономірностей. Вони можуть бути побудовані на основі абсолютної або відносної прибутковості, шляхом порівняння поперечного перерізу активів або аналізу часових рядів активів в рамках традиційних класів активів або між ними і на різних часових горизонтах.

Акції з низькими цінами по відношенню до їх фундаментальної вартості, як правило, приносять дохід, що перевищує орієнтований на капіталізацію орієнтир. Фактори вартості відображають цю кореляцію та призначені для подачі сигналів про покупку недооцінених активів, що є відносно дешевих, а також про продаж переоцінених та дорогих активів. З цієї причини в основі будь-якої стратегії створення вартості полягає модель оцінки, яка оцінює або прогнозує справедливу або фундаментальну вартість активу. Справедлива вартість може бути визначена як абсолютний рівень цін, спред по відношенню до інших активів або діапазон, в якому актив повинен торгувати (наприклад, два стандартних відхилення). Стратегії створення вартості засновані на повернення цін до справедливої вартості активу. Вони припускають, що ціни тільки тимчасово відхиляються від справедливої вартості через поведінкових ефектів, таких як надмірна реакція або стада, або ефектів ліквідності, таких як тимчасове вплив на ринок або довгострокові тертя між попитом і пропозицією. Оскільки фактори вартості покладаються на звернення до середнього, вони часто виявляють властивості, протилежні властивостям факторів імпульсу. Що стосується акцій, то протилежністю вартісних акцій є акції зростання з високою оцінкою через очікування зростання [10, 11].

Фактор низької волатильності відображає надлишкову прибутковість акцій з волатильністю, бета-коефіцієнтом або ідіосінкразіческім ризиком нижче середнього. Акції з більш високою ринковою капіталізацією, як правило, мають більш низьку волатильність, тому традиційний фактор розміру часто поєднується з більш новим фактором волатильності.

Аномалія низької волатильності – це емпірична загадка, яка суперечить основним принципам фінансів. Модель ціноутворення капітальних активів (CAPM) та інші моделі ціноутворення стверджують, що більш високий ризик повинен

приносити більш високу прибутковість, але на багатьох ринках і протягом тривалих періодів спостерігається зворотне, коли менш ризиковані активи перевищують своїх більш ризикованих аналогів [9].

Фактор якості спрямований на отримання додаткового прибутку від компаній, які є високоприбутковими, ефективними, безпечними, стабільними і добре керованими, інакше кажучи, високої якості в порівнянні з ринком. Ринки також, схоже, заохочують відносну впевненість у прибутках і карають акції з високою волатильністю прибутку. Орієнтація портфеля на бізнес з високою якістю вже давно підтримується збирачами акцій, які покладаються на фундаментальний аналіз, але це відносно нове явище в кількісних інвестиціях. Основна проблема полягає в тому, як послідовно і об'єктивно визначити фактор якості з використанням кількісних показників, враховуючи суб'єктивний характер якості [6].

Стратегії, засновані на окремих факторах якості, як правило, працюють протициклічно, оскільки інвестори платять надбавку, щоб мінімізувати ризики зниження вартості і підвищити оцінку. З цієї причини в багатофакторній стратегії чинники якості часто поєднуються з іншими факторами ризику, найчастіше з вартістю, щоб забезпечити якість за розумною ціною. Фактори якості для довгих і коротких позицій зазвичай мають негативну ринкову бета, тому що це довгі якісні акції, які також мають низьку волатильність, і короткі більш волатильні і низькоякісні акції. Отже, чинники якості часто позитивно корелюють з факторами низької волатильності і імпульсу і негативно корелюють з вартістю і широким впливом на ринок [7, 9].

1.4 Технічні індикатори

1.4.1 Simple moving average SMA

Проста змінна середня (SMA) – це арифметична змінна середня, що обчислюється шляхом додавання останніх цін і подальшого розподілу цієї цифри на

кількість періодів часу в розрахунковою середньою. Короткострокові середні значення швидко реагують на зміни ціни базової цінного паперу, в той час як довгострокові середні значення реагують повільніше.

$$SimpleMovingAverage = \frac{\sum_{i=1}^{N} Pi}{N},$$

де Pi – ціна у часовий проміжок i;

N – кількість цін складених разом або кількість часових проміжків.

Просте ковзне середнє можна налаштувати, оскільки воно може бути розраховане для різної кількості періодів часу. Це робиться шляхом додавання ціни закриття цінного паперу за кілька періодів часу і подальшого розподілу цієї суми на кількість періодів часу, що дає середню ціну цінного паперу за період часу. Проста змінна середня згладжує волатильність і спрощує перегляд цінової тенденції цінного паперу. Якщо проста змінна середня вказує вгору, це означає, що ціна цінного паперу зростає. Якщо він спрямований вниз, це означає, що ціна цінного паперу знижується. Чим довше тимчасові рамки для ковзної середньої, тим більш гладкою буде проста змінна середня. Короткострокова змінна середня більш волатильна, але її свідчення ближче до вихідних даних [1].

Ковзаючі середні – важливий аналітичний інструмент, який використовується для визначення поточних цінових тенденцій і потенціалу зміни встановленого тренда. Найпростіший спосіб використання SMA в технічному аналізі – це використовувати його для швидкого визначення того, чи знаходиться цінний папір у висхідному або низхідному тренді. Ще одне популярне, хоча і трохи більш складне, аналітичне використання – це порівняння пари простих ковзних середніх, кожна з яких охоплює різні часові рамки. Якщо короткострокова проста змінна середня вище довгострокової середньої, очікується висхідний тренд. З іншого боку, якщо довгострокове середнє значення вище короткострокового середнього, то очікуваним результатом може бути спадний тренд.

Невизначено, чи слід приділяти більше уваги самим останнім дням

періоду часу або більш віддаленим даними. Багато трейдерів вважають, що нові дані будуть краще відображати поточну тенденцію, в якій рухається цінний папір. У той же час інші трейдери вважають, що перевагу одних дат в порівнянні з іншими буде спотворювати тренд. Отже, SMA може занадто сильно покладатися на застарілі дані, оскільки він враховує вплив 10-го або 200-го дня так само, як і перший або другий.

Аналогічним чином SMA повністю покладається на історичні дані. Багато людей (включаючи економістів) вважають, що ринки ефективні, тобто поточні ринкові ціни вже відображають всю доступну інформацію. Якщо ринки дійсно ефективні, використання історичних даних нічого не має сказати нам про майбутній напрямок цін на активи [2].

1.4.2 Exponential moving average EMA

Експоненціальна змінна середня (EMA) є найвідомішим і широко використовуваним індикатором технічного аналізу для даних часових рядів. EMA схожа на просту ковзаючу середню, але замість того, щоб однаково зважувати всі ціни в історії, вона надає більшої ваги з останніми спостереженнями за цінами і менший – старим. Це спроба реалізувати інтуїтивно зрозумілу ідею про те, що нове спостереження за цінами містить більш свіжу інформацію, ніж ціни в минулому. Також можна надати більшої ваги старим спостереженнями за цінами і меншої – новим спостереженнями за цінами. Це спробувало б реалізувати ідею про те, що довгострокові тенденції містять більше інформації, ніж короткострокові волатильні руху цін.

Ваговий коефіцієнт залежить від обраного періоду часу EMA: чим коротше період часу, тим більше EMA реагує на нові спостереження цін. Іншими словами, EMA швидше сходиться до нових спостереженнями за цінами і швидше забуває старі спостереження, також звані Fast EMA. Чим довше період часу, тим менше реакція EMA на нові спостереження цін; тобто EMA повільніше сходиться до нових цінових спостережень і повільніше забуває старі спостереження, також звані повільними EMA [1].

Грунтуючись на описі EMA, він формулюється як ваговий коефіцієнт, застосовуваний до нових цінових спостереженнями, і ваговий коефіцієнт, застосовуваний до поточного значення EMA, щоб отримати нове значення EMA. Оскільки сума ваг повинна бути дорівнює 1, щоб одиниці EMA залишалися такими ж, як одиниці ціни, тобто $s, ваговий коефіцієнт, застосовуваний до значень EMA, виявляється p-1. Отже, ми отримуємо наступні два формулювання нових значень EMA, засновані на старих значеннях EMA і нових цінових спостереженнях, які є одними і тими ж термінами, написаними в двох різних формах (див. рис. 1.4) [2]:

$$EMA = (P - EMA_{old}) \times \mu + EMA_{old},$$

$$EMA = P \times \mu + (1 - \mu) \times EMA_{old},$$

де P −поточна ціна;

EMA_{old} −попереднє значення EMA;

μ −константа згладжування, як правило має значення $\frac{2}{(n+1)}$;

N −кількість часових проміжків.

1.4.3 Moving average convergence divergence MACD

Дивергенція конвергенції ковзаючих середніх (MACD) – ще один в класі індикаторів, які будують поверх ковзаючих середніх цін.

$$MACD = EMA_{Fast} - EMA_{Slow},$$

$$MACD_{Signal} = EMA_{MACD},$$

$$MACD_{Histogram} = MACD - MACD_{Signal},$$

Дивергенція конвергенції ковзають середніх була створена Джеральдом Аппелем. він встановлює різницю між швидкою експоненційної ковзної середньої і повільної експоненційної ковзної середньої. Однак в разі MACD ми застосовуємо згладжену експонентну ковзаючу середню до самого значенням MACD, щоб отримати остаточний вихідний сигнал від індикатора MACD. Правильно налаштований сигнал MACD може успішно фіксувати напрямок, величину і тривалість тренда ціни інструменту [1].

Одна з основних проблем з дивергенцией полягає в тому, що вона часто може сигналізувати про можливий розворот, але тоді насправді розвороту не відбувається – це дає помилкове спрацьовування. Інша проблема в тому, що дивергенція не передвіщає всіх розворотів. Іншими словами, він пророкує занадто багато розворотів, які не відбуваються, і недостатньо реальних розворотів ціни.

«Хибнопозитивна» дивергенція часто виникає, коли ціна активу рухається убік, наприклад, в діапазоні або трикутнику після тренда. Уповільнення імпульсу – бічний рух або повільний рух тренда-ціни змусить MACD відірватися від своїх попередніх екстремумів і тяжіти до нульових лініях навіть за відсутності справжнього розвороту.

1.4.4 Bollinger bands BBANDS

Смуги Боллинджера (BBANDS) також будуються на основі ковзних середніх, але враховують недавню волатильність цін, яка робить індикатор більш адаптивним до різних ринкових умов.

Смуги Боллинджера – це добре відомий індикатор технічного аналізу, розроблений Джоном Боллинджер. Він обчислює ковзаючу середню цін. Крім того, він обчислює стандартне відхилення цін в період ретроспективного аналізу, розглядаючи ковзаючу середню як середню ціну. Потім він створює верхню смугу, яка є ковзної середньої плюс кілька стандартних відхилень ціни, і нижню смугу, яка представляє собою ковзаючу середню за вирахуванням декількох стандартних

відхилень ціни. Ця смуга являє собою очікувану волатильність цін, якщо розглядати ковзаючу середню ціни як довідкову. Тепер, коли ціни виходять за межі цих смуг, це можна інтерпретувати як сигнал прориву/тренда або сигнал повернення до середнього значення перекупленності/продажу [1, 2].

$$BBAND_{Middle} = SMA_{n-periods'}$$

$$BBAND_{Upper} = BBAND_{Middle} + (\beta * \delta),$$

$$BBAND_{Lower} = BBAND_{Middle} - (\beta * \delta),$$

де β −стандартний фактор відхилення;

δ −стандартне відхилення, обчислюється як $\sqrt{\frac{\sum_{i=1}^{N}(Pi-SMA)^2}{n}}$.

Смуги Боллинджера – дуже популярна техніка. Багато трейдерів вважають, що чим ближче ціна рухається до верхньої смузі, тим більше ринок перекуплений і чим ближче ціна рухається до нижньої смузі, тим більше ринок перепроданий.

Стиснення – це центральна концепція смуг Боллинджера. Коли смуги зближуються, обмежуючи ковзаючу середню, це називається стисненням. Стиснення сигналізує про період низької волатильності і розглядається трейдерами як потенційний ознака майбутньої підвищеної волатильності і можливих торгових можливостей. І навпаки, чим ширше переміщаються смуги, тим вище ймовірність зниження волатильності і тим вище ймовірність виходу з угоди. Однак ці умови не є торговими сигналами. Смуги не вказують, коли може відбутися зміна або в якому напрямку ціна може рухатися.

Приблизно 90% руху ціни відбувається між двома смугами. Будь який прорив вище або нижче смуг – серйозна подія. Пробій не є торговим сигналом.

Помилка більшості людей полягає в тому, що вони вважають, що ціна, що досягає або перевищує одну зі смуг, є сигналом до купівлі або продажу. Прориви не дають підказки щодо спрямування і ступеня майбутнього цінового руху.

Смуги Болліндже́ра не є окремою торговою системою. Це просто один індикатор, призначений для надання трейдерам інформації про волатильність цін. Джон Боллинджер пропонує використовувати їх з двома або трьома іншими некоррельорованними індикаторами, які дають більш прямі ринкові сигнали. Він вважає дуже важливим використовувати індикатори, засновані на різних типах даних. Деякі з його улюблених технічних методів – це дивергенція/конвергенція ковзаючих середніх (MACD), балансовий обсяг і індекс відносної сили (RSI) [2].

1.4.5 Relative strength indicator RSI

Індикатор відносної сили (RSI) сильно відрізняється від попередніх індикаторів, які ґрунтувалися на ковзних середніх цін. Це засновано на зміні ціни за періоди, щоб зафіксувати силу/величину цінових рухів.

Індикатор відносної сили був розроблений Дж. Уеллсом Уайлдером. Він включає період ретроспективного аналізу, який використовується для обчислення величини середнього прибутку/зростання цін за цей період, а також величини середніх значень збитків/зниження ціни за цей період. Потім він обчислює значення RSI, яке нормалізує значення сигналу, щоб воно залишалося в діапазоні від 0 до 100, і намагається визначити, чи було набагато більше виграшів в порівнянні з втратами або було набагато більше втрат щодо прибутків. Значення RSI більше 50% вказують на висхідний тренд, а значення RSI нижче 50% вказують на спадний тренд [1].

$$RelativeStrength(RS) = \frac{\sum|GainsOverLastNPeriods|}{\sum|LossesOverLastNPeriods|},$$

$$RelativeStrengthIndicator(RSI) = 100 - \frac{100}{(1+RS)},$$

Традиційна інтерпретація і використання RSI полягає в тому, що значення 70 або вище вказують на те, що цінний папір стає перекупленності або переоціненою і може бути націлена на розворот тренда або коригувальний відкат ціни. Значення RSI 30 або нижче вказує на стан перепроданості або недооцінки.

RSI буде рости в міру збільшення кількості та розміру позитивних закриттів і падати у міру збільшення кількості та розміру збитків. Друга частина розрахунку згладжує результат, тому RSI буде тільки близько 100 або 0 на ринку з сильним трендом [2].

RSI порівнює “бичачий” і “ведмежий” цінової імпульс і відображає результати в осцилляторе, який можна розмістити під графіком ціни. Як і більшість технічних індикаторів, його сигнали найбільш надійні, коли вони відповідають довгостроковим тренду.

Оскільки індикатор показує імпульс, він може залишатися перекупленим або перепроданим протягом тривалого часу, коли актив має значний імпульс в будь-якому напрямку. Отже, RSI найбільш корисний на хиткому ринку, де ціна активу чергується між “бичачим” і “ведмежим” рухами.

1.4.6 Standard deviation STDDEV

Стандартне відхилення (STDEV) є основним заходом волатильності цін, яка використовується в поєднанні з безліччю інших індикаторів технічного аналізу для їх поліпшення.

Стандартне відхилення – це стандартна міра, яка обчислюється шляхом вимірювання квадрата відхилення окремих цін від середньої ціни і подальшого перебування середнього значення всіх цих значень квадратичного відхилення. Це значення відомо як дисперсія, а стандартне відхилення виходить шляхом вилучення квадратного кореня з дисперсії. Більші STDEV є ознакою більш волатильних ринків або більш значних очікуваних цінових рухів, тому торгові стратегії повинні враховувати цю підвищену волатильність в оцінках ризиків і іншому торговельному поведінці [1, 2].

Щоб обчислити стандартне відхилення, спочатку ми обчислюємо дисперсію:

$$\sigma^2 = \frac{\sum_{i=1}^{n} (Pi - SMA)^2}{n},$$

Стандартне відхилення – це статистика, яка вимірює дисперсію набору даних щодо його середнього значення і розраховується як квадратний корінь з дисперсії. Стандартне відхилення розраховується як квадратний корінь з дисперсії шляхом визначення відхилення кожної точки даних щодо середнього значення. Якщо точки даних знаходяться далі від середнього значення, в наборі даних є більше відхилення; отже, чим більше розкид даних, тим вище стандартне відхилення.

Стандартне відхилення – це статистичний показник в фінансах, який в застосуванні до річній нормі прибутковості інвестицій проливає світло на історичну волатильність цих інвестицій. Чим більше стандартне відхилення цінних паперів, тим більша різниця між кожною ціною і середнім значенням, яке показує більший діапазон цін. Наприклад, волатильні акції мають високу стандартне відхилення, в той час як відхилення стабільних блакитних фішок зазвичай досить низька.

Стандартне відхилення – особливо корисний інструмент в інвестиційних і торгових стратегіях, оскільки він допомагає вимірювати волатильність ринку і цінних паперів, а також прогнозувати тенденції продуктивності. Наприклад, що стосується інвестування, індексний фонд, ймовірно, буде мати низьку стандартне відхилення в порівнянні з його еталонним індексом, оскільки мета фонду – відтворити індекс [2].

З іншого боку, можна очікувати, що фонди агресивного зростання матимуть високу стандартне відхилення від відносних фондових індексів, оскільки їх керуючі портфелями роблять агресивні ставки для отримання прибутку вище середньої.

Нижчий стандартне відхилення не обов'язково переважно. Все залежить від вкладень і готовності інвестора прийняти на себе ризик. Маючи справу з

величиною відхилень в своїх портфелях, інвестори повинні враховувати свою терпимість до волатильності і свої загальні інвестиційні цілі. Більш агресивні інвестори можуть бути задоволені інвестиційною стратегією, яка вибирає транспортні засоби з волатильністю вище середнього, в той час як більш консервативні інвестори можуть не робити цього.

Стандартне відхилення – один з ключових фундаментальних показників ризику, який використовують аналітики, що управляють портфелем, консультанти. Інвестиційні фірми повідомляють про стандартному відхиленні своїх пайових інвестиційних фондів та інших продуктів. Великий розкид показує, наскільки прибутковість фонду відхиляється від очікуваної нормальної прибутковості.

1.4.7 Momentum MOM

Імпульс, також званий MOM, є важливим показником швидкості і величини цінових рухів. Часто це ключовий індикатор торгових алгоритмів на основі тренда/прориву.

У своїй простій формі імпульс – це просто різниця між поточною ціною і ціною деяких фіксованих періодів часу в минулому. Послідовні періоди позитивних значень імпульсу вказують на висхідний тренд і навпаки, якщо імпульс послідовно негативний, це вказує на низхідний тренд. Часто ми використовуємо прості/експоненціальні ковзаючі середні індикатора MOM, як показано на рисунку 1.9, для виявлення стійких тенденцій [1]:

$$MOM = Price_t - Price_{t-n},$$

де $Price_t$ −значення ціни у час t;

$Price_{t-n}$ −значення n часових проміжків до часу t.

Імпульс – це швидкість прискорення ціни або обсягу цінних паперів, тобто швидкість, з якою змінюється ціна. Простіше кажучи, це відноситься до

швидкості зміни руху ціни на конкретний актив і зазвичай визначається як швидкість. У технічному аналізі імпульс вважається осциллятором і використовується для визначення тенденцій.

Коли застосовується, інвестор може купувати або продавати в залежності від сили тенденцій в ціні активу. Якщо трейдер хоче використовувати стратегію, засновану на імпульсі, він відкриває довгу позицію по акції або активу, які мають висхідний тренд. Якщо акція рухається вниз, він відкриває коротку позицію. Замість традиційної філософії торгівлі – купувати дешево, продавати дорого – імпульсивне інвестування прагне продавати дешево і купувати дешевше або купувати дорого і продавати дорожче. Замість того, щоб визначати модель продовження або розвороту, імпульсні інвестори зосереджуються на тенденції, створеної самим останнім проривом ціни.

Деякі інструменти для імпульсних інвесторів допомагають визначити тренд, наприклад, лінія тренда. Лінія тренда – це лінія, проведена від максимальної ціни до мінімальної або навпаки за певний період часу. Якщо лінія висхідна, тренд висхідний, і імпульсивний інвестор купує акції. Якщо лінія тренда спадна, тренд спадний, і імпульсний інвестор продає акції.

Отже, імпульсне інвестування – це чисто технічний індикатор. Хоча «імпульс» може ставитися до фундаментальних показників ефективності, таким як виручка і прибуток, він найчастіше використовується в відношенні історичних цін на активи в якості технічного індикатора [2].

Як і будь-який інший стиль торгівлі, є ризики, пов'язані з імпульсною торгівлею. Використовуючи цю техніку, треба звертати особливу увагу на те, що тенденції цін ніколи не гарантовані і неочікувані розвороти або виправлення можуть статися через несподівані новин або зміни настроїв інвесторів на ринку.

2 ПРОЕКТУВАННЯ БІБЛІОТЕКИ ПРОГНОЗУВАННЯ ФІНАНСОВИХ ЧАСОВИХ РЯДІВ

Етап проектування містить визначення функціональних, логічних та структурних особливостей і обмежень системи. У цьому розділі розглядаються діаграми в нотації UML, що дозволяє наочно зобразити структуру і взаємодію різних компонентів системи. За призначенням, UML діаграми можна розділити на наступні класи: діаграми поведінки і структурні.

2.1 Діаграми поведінки

Цей клас включає діаграми варіантів використання (англ. Use Case Diagrams) – для моделювання бізнес-процесів і функціональних вимог до системи що розробляється [3].

На діаграмі використання показані різні функціональні можливості системи. Завдяки кольоровому оформленню, компоненти логічно поділені за сферами використання, в якості акторів винесені зовнішні бібліотеки. Користувачеві надається можливість взаємодії лише з двома високорівневими класами, які, в свою чергу, містять всі необхідні методи для подальшої більш низкорівневої взаємодії з допоміжними класами.

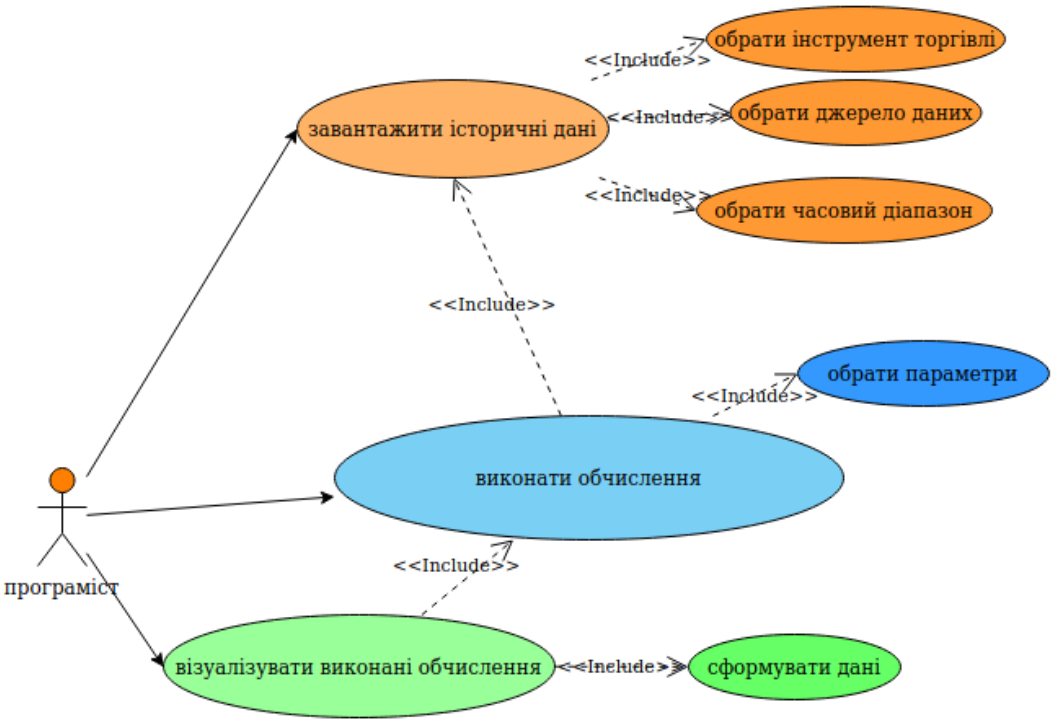

Рисунок 2.1 – Діаграма використання

2.2 Структурні діаграми

Даний клас діаграм включає в себе наступні діаграми: класів – визначають типи класів системи, компонентів – моделюють фізичний рівень системи, розміщення – відображають фізичні взаємозв'язку між програмними та апаратними компонентами системи.

На рисунках 2.2–2.4 показані діаграми класів, що логічно розділені за функціональними задачами. Кожна група класів відповідає за певну задачу.

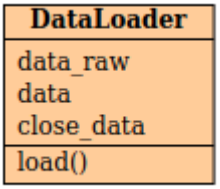

Рисунок 2.2 – Діаграма классу DataLoader

Клас DataLoader відповідає за завантаження історичних даних з зовнішнього ресурсу. Також функціонал даного класу дозволяє записувати отримані дані в файл. Як параметри можна задати початкову дату завантаження, кінцеву дату завантаження, перелік символів торгових інструментів і ресурс, з якого будуть завантажуватися дані. Результатом роботи даного класу є масив даних, який зберігає ціну на момент закриття біржі в десяткового формі. Довжина масиву дорівнює кількості днів, що входять в обраний часовий проміжок.

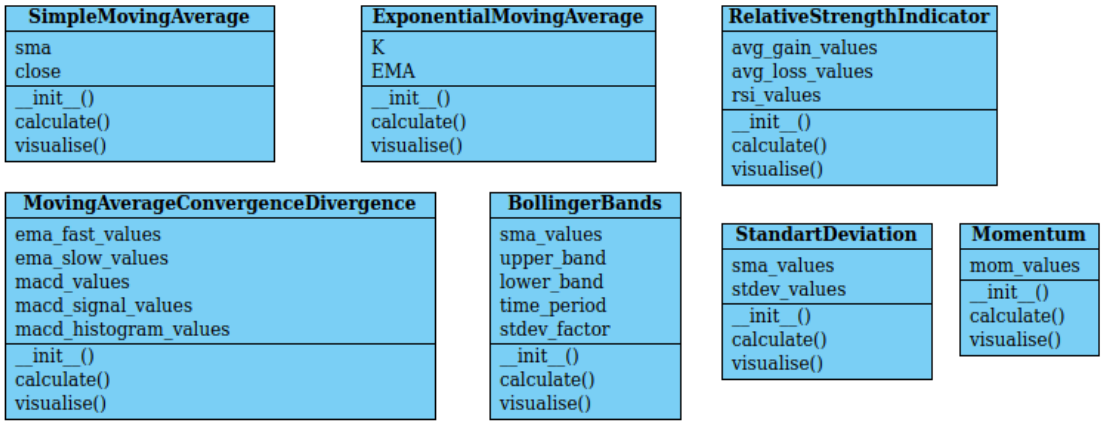

Рисунок 2.3 – Діаграма класів технічних індикаторів

Дана безліч класів являє собою програмні реалізації обраних технічних індикаторів. За своєю структурою всі класи схожі: метод ініціалізації відповідає за початкове завантаження даних і передачу значення констант, необхідних при обчисленнях; метод calculate() робить необхідні обчислення на основі історичних даних, результатом його роботи є числовий масив; за візуалізацію

результатів обчислень відповідає метод visualise(), він перетворює отримані дані в більш зручний формат для уявлення і будує графіки.

Кожен клас в якості параметрів приймає різну кількість різних параметрів, в залежності від формули технічного індикатора.

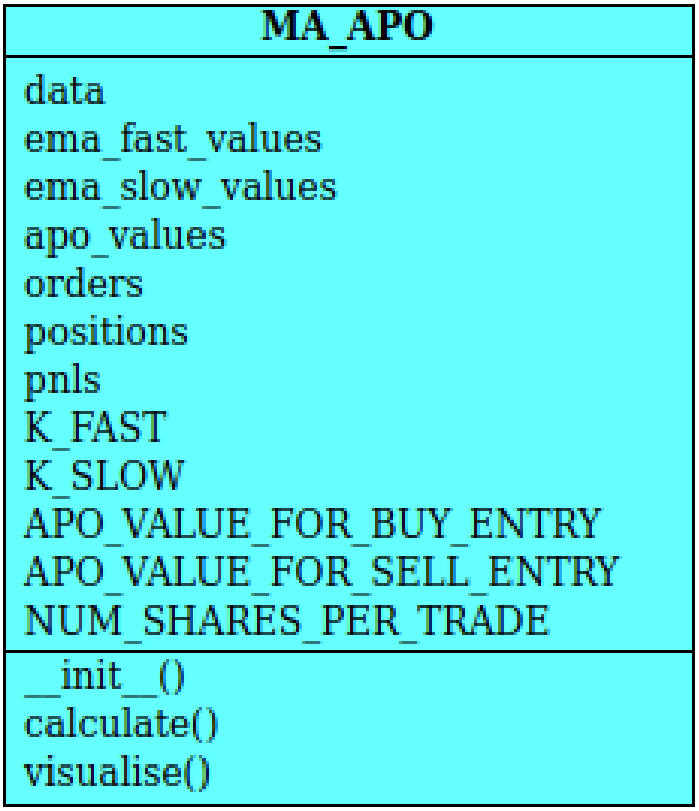

Рисунок 2.4 – Діаграма класу MA_APO

Даний клас є прикладом реалізації торгової стратегії. За своєю структурою вона схожа з класами, що реалізують технічні індикатори, відмінність полягає в кількості параметрів і констант, необхідних для обчислень.

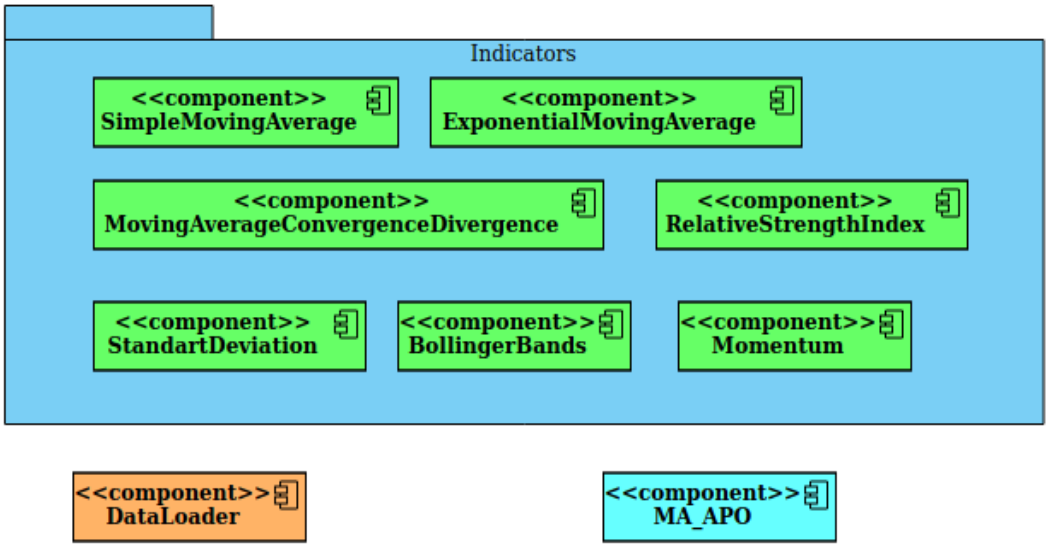

Рисунок 2.5 – Діаграма компонентів

На цій діаграмі компонентів (див. рис. 2.5) зображені компоненти програмного забезпечення. Кожен клас системи перетворюється в компонент вихідного коду і є виконуваним. Також окремо виділяються зовнішні бібліотеки. Між окремими компонентами зображено залежності, відповідні залежностям на етапі компіляції або виконання програми [3].

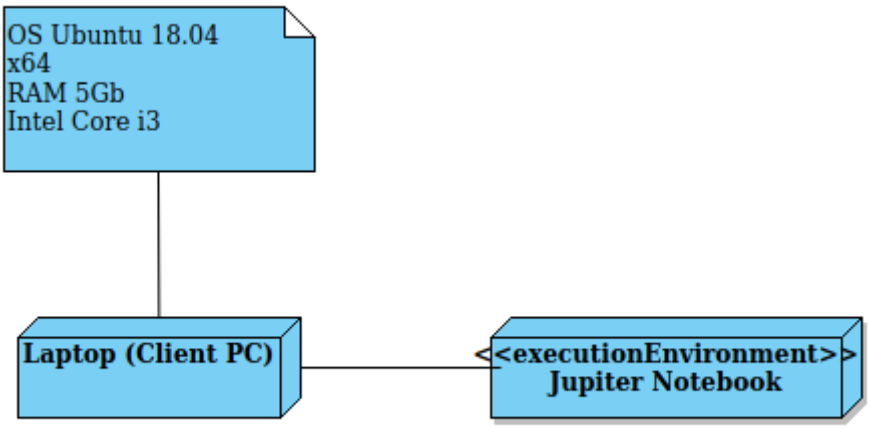

Рисунок 2.6 – Діаграма розгортання

Діаграма розгортання (див. рис. 2.6) демонструє розміщення об'єктів і компонентів в розподіленій системі. Показано вузли, що представляють собою певний тип обчислювального пристрою. Також вказані характеристики середовища виконання коду [3].

2.3 Діаграми взаємодії

Даний клас діаграм включає в себе діаграми послідовності і кооперації – для моделювання процесу обміну повідомленнями між об'єктами [3].

Діаграми послідовності відображають тимчасову послідовність подій, що відбувається в рамках варіанту використання.

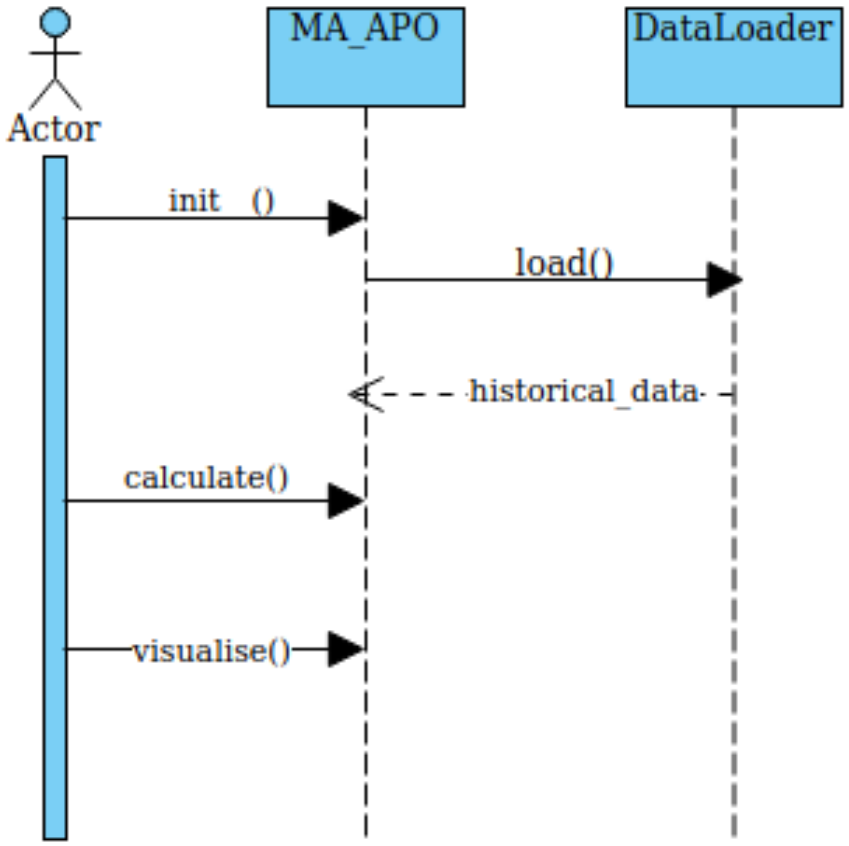

Рисунок 2.7 – Діаграма послідовності

На рисунку 2.7 показана тимчасова послідовність взаємодії між усіма класами, які беруть участь у виконанні торгової стратегії. Користувач в першу чергу створює новий екземпляр класу MA_APO, при його ініціалізації відбувається завантаження історичних даних за допомогою класу DataLoader. Далі послідовно відбувається звернення до методів calculate() і visualise(). Результатом виконання даних дій будуть графіки всіх індикаторів, що беруть участь в роботі стратегії, відображення реалізованих позицій, результат проведених торгів у вигляді графіка прибутку і втрат.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Реалізація методів завантаження даних

Клас DataLoader відповідає за завантаження і попередню підготовку історичних даних акцій Google. Джерелом даних є сервіс Yahoo Finance.

```
class DataLoader:

    data_raw = []
    data = []
    close_data = []

    def load(self, start_date = '2014-01-01', end_date = '2018-01-01', symbols = ['GOOG'], source = 'yahoo'):
        self.data = data.DataReader(symbols, source, start_date, end_date)
        self.close_data = pd.DataFrame(index=self.data.index)
        self.close_data['price'] = self.data['Adj Close']
        self.close_data['series'] = pd.Series(self.close_data['price'], index=self.data.index)
```

Рисунок 3.1 – Реалізація классу DataLoader

В результаті роботи клас повертає масив даних типу Data Frame:

	price	series
Date		
2014-01-02	554.481689	554.481689
2014-01-03	550.436829	550.436829
2014-01-06	556.573853	556.573853
2014-01-07	567.303589	567.303589
2014-01-08	568.484192	568.484192

Рисунок 3.2 – Приклад початкових даних

Набір даних містить наступні поля:

- Date – дата торгів
- Price – цифрове поле, містить вартість валюти в USD на момент закриття біржі
- Series – цифрове поле, містить вартість валюти в USD на момент закриття біржі в форматі Series

3.2 Реалізація технічних індикаторів

3.2.1 Simple Moving Average

Клас SimpleMovingAverage відповідає за реалізацію відповідного індикатора. Спочатку при створенні класу відбувається ініціалізація змінних і завантаження історичних даних.

У методі calculate є можливість вибору тимчасового вікна на якому буде працювати функція. Серед історичної інформації виділяються останні N днів, з яких в подальшому знаходиться середнє значення, яке і є результатом згладжування. З подібних значень і складається результат роботи функції SimpleMovingAverage.

Функція visualise відповідає за візуальне представлення результатів обчислень. Для наочності графік ціни закриття і графік Simple Moving Average представлені на одній площині.

```
class SimpleMovingAverage:

    sma = []
    close = []


    def __init__(self):
        self.dataLoader = DataLoader()
        self.dataLoader.load()
        self.close = self.dataLoader.close_data['series']


    def calculate(self, time_period = 20):
        history = []
        for close_price in self.close:
            history.append(close_price)
            if len(history) > time_period:
                del(history[0])

            self.sma.append(stats.mean(history))

    def visualise(self ):
        sma_series = pd.Series(self.sma, index=self.dataLoader.close_data.index)
        fig = plt.figure()
        ax1 = fig.add_subplot(111, ylabel='GOOG')
        self.close.plot(ax=ax1, color='g', lw=2., legend=True)
        sma_series.plot(ax=ax1, color='r', lw=2., legend=True)
        plt.show()
```

Рисунок 3.3 – Реалізація классу SimpleMovingAverage

На рисунку 3.4 видно, що 20-денна SMA надає намічений ефект згладжування і вирівнює мікроволатільность фактичної ціни акцій, забезпечуючи більш стабільну криву ціни.

Рисунок 3.4 – Результат роботи классу SimpleMovingAverage

3.2.2 Exponential Moving Average

При реализации ExponentialMovingAverage використовується коефіцієнт згладжування за замовчуванням 2 / (n + 1). Подібно SMA, EMA також вирівнює звичайні денні ціни. Перевага EMA полягає в тому, що є можливість зважувати останні ціни з більш високою вагою, ніж SMA, яка виконує рівномірне зважування.

За аналогією з попереднім класом, процес ініціалізації включає в себе привласнення базових значень ключовим змінним і параметрам, завантаження історичних даних.

У методі calculate відбувається обчислення кожного значення EMA відповідно до формули і заданим коефіцієнтом згладжування.

```
class ExponentialMovingAverage:

    K = 0
    ema = []


    def __init__(self, num_periods = 20):
        self.dataLoader = DataLoader()
        self.dataLoader.load()
        self.close = self.dataLoader.close_data['series']

        self.K = 2 / (num_periods + 1)


    def calculate(self):
        ema_previous = 0

        for close_price in self.close:
            if (ema_previous == 0):
                ema_previous = close_price
            else:
                ema_previous = (close_price - ema_previous) * self.K + ema_previous

            self.ema.append(ema_previous)


    def visualise(self):
        ema_series = pd.Series(self.ema, index=self.dataLoader.close_data.index)
        fig = plt.figure()
        ax1 = fig.add_subplot(111, ylabel='GOOG')
        self.close.plot(ax=ax1, color='g', lw=2., legend=True)
        ema_series.plot(ax=ax1, color='r', lw=2., legend=True)
        plt.show()
```

Рисунок 3.5 – Реалізація классу ExponentialMovingAverage

Функція visualise() відповідає за візуальне представлення результатів обчислень. Для наочності графік ціни закриття і графік Exponential Moving Average представлені на одній площині.

З графіка видно, що EMA має дуже схожий ефект згладжування на SMA, як і очікувалося, і знижує шум у вихідних цінах. Однак додатковий параметр, доступний в EMA на додаток до параметру, дозволяє нам контролювати відносну вагу, якої надає нового цінового спостереження, в порівнянні з більш старими ціновими спостереженнями. Це дозволяє нам будувати різні варіанти EMA, варіюючи параметр, щоб створювати швидкі і повільні EMA навіть для одного і того ж параметра.

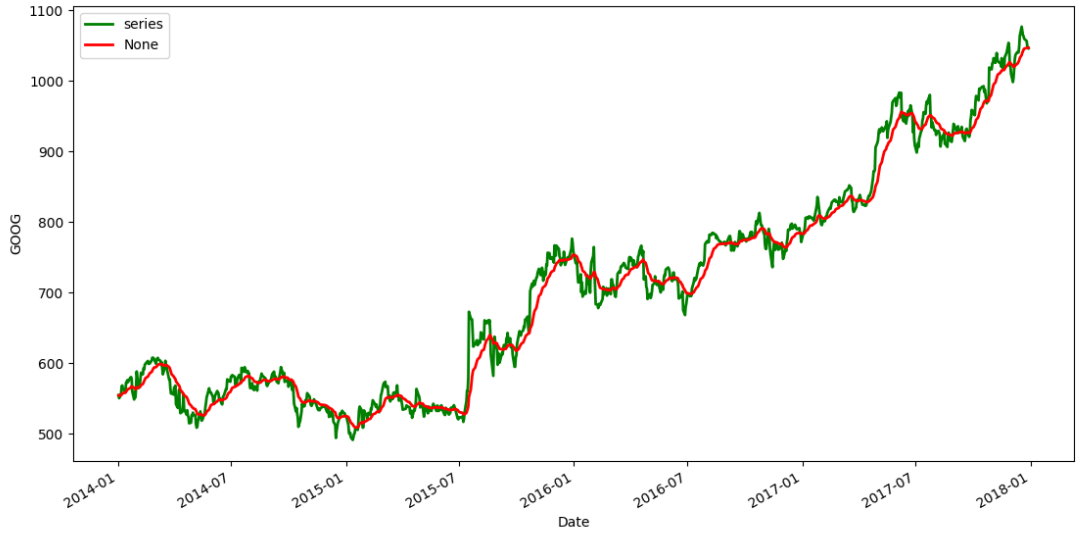

Рисунок 3.6 – Результат роботи классу ExponentialMovingAverage

3.2.3 Moving Average Convergence Divergence

У даній імплементації MovingAverageConvergenceDivergence будується на основі двох експоненційних ковзають середніх: швидка EMA обчислюється на 10-денному періоді, повільна – на періоді в 40 днів.

В якості стандартних згладжують факторів для ковзних середніх прийняті значення в 2/11 і 2/41 відповідно. Тимчасове вікно і коефіцієнт згладжування для MACD рівні 20 днів і 2/21 відповідно.

```
class MovingAverageConvergenceDivergence:

    ema_fast_values = []
    ema_slow_values = []
    macd_values = []
    macd_signal_values = []
    macd_histogram_values = []

    def __init__(self, num_periods_fast = 10, num_periods_slow = 40, num_periods_macd = 20):
        self.dataLoader = DataLoader()
        self.dataLoader.load()
        self.close = self.dataLoader.close_data['series']

        self.K_fast = 2 / (num_periods_fast + 1)
        self.K_slow = 2 / (num_periods_slow + 1)
        self.K_macd = 2 / (num_periods_macd + 1)
```

Рисунок 3.7 – Реалізація методу __init__() классу MACD

У методі calculate на кожній ітерації обчислюються значення змінних середніх. На їх основі виробляється обчислення значень MACD.

```
def calculate(self):
    ema_fast = 0
    ema_slow = 0
    ema_macd = 0
    macd = 0

    for close_price in self.close:
        if (ema_fast == 0):
            ema_fast = close_price
            ema_slow = close_price
        else:
            ema_fast = (close_price - ema_fast) * self.K_fast + ema_fast
            ema_slow = (close_price - ema_slow) * self.K_slow + ema_slow

        self.ema_fast_values.append(ema_fast)
        self.ema_slow_values.append(ema_slow)
        macd = ema_fast - ema_slow

        if (ema_macd == 0):
            ema_macd = macd
        else:
            ema_macd = (macd - ema_macd) * self.K_slow + ema_macd

        self.macd_values.append(macd)
        self.macd_signal_values.append(ema_macd)
        self.macd_histogram_values.append(macd - ema_macd)
```

Рисунок 3.8 – Реалізація методу calculate() классу MACD

Функція visualise() відповідає за візуальне представлення результатів обчислень. Для наочності графік ціни закриття і графік MovingAverageConvergenceDivergence представлені на одній площині.

На представленні видно, що EMA МАСД показує більш плавне згладжування в порівнянні з чистими даними МАСД, що дозволяє виключити зайвий шум в даних МАСД. Гістограма, яка є різницею між легкими середніми, дозволяє вловити часовий період попередній розвороту ціни, а також показує силу поточних трендів, залишаючись в позитивному або негативному діапазоні.

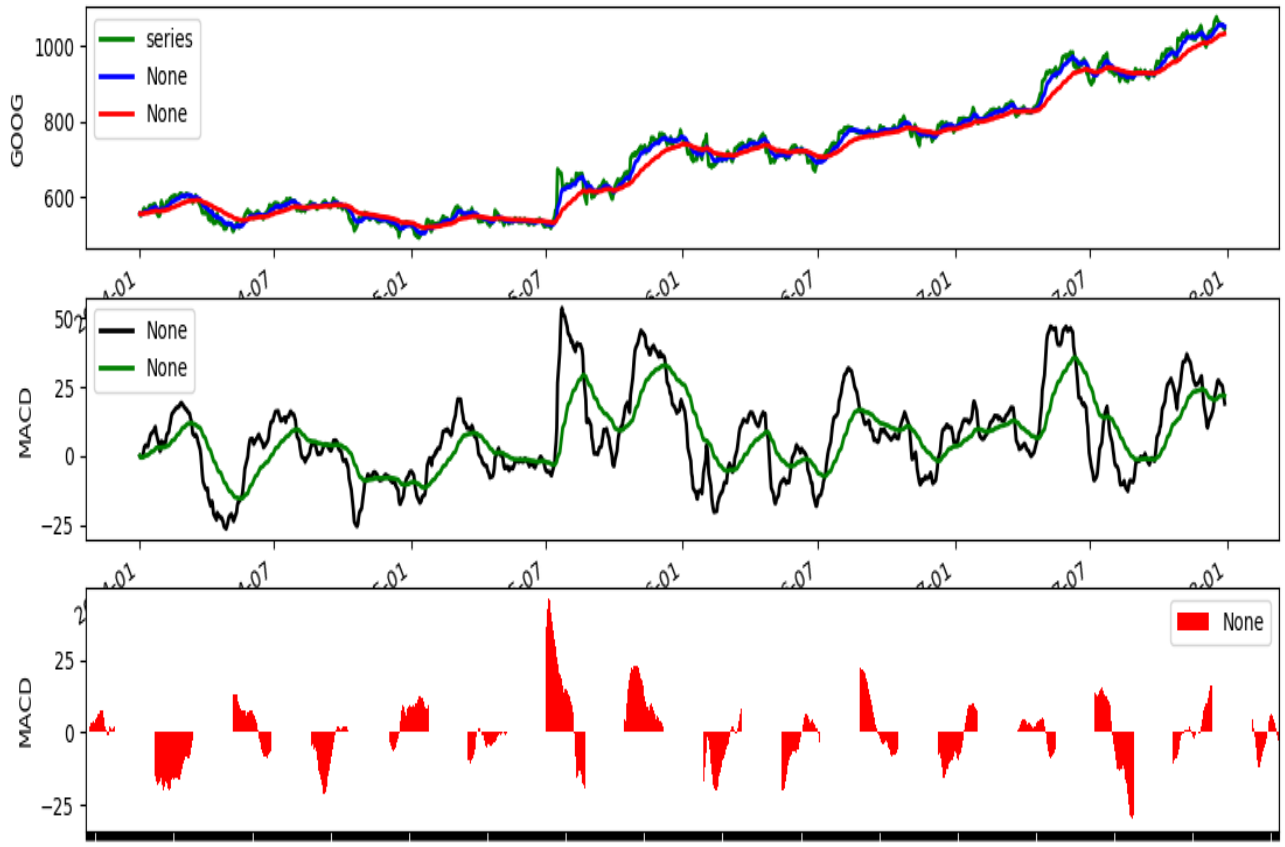

Рисунок 3.9 – Результат роботи классу MACD

3.2.4 Bollinger Bands

Для реалізації BollingerBands була взята проста змінна середня з періодом в 20 днів. Також був використаний stdev фактор рівний 2 для обчислення верхньої та нижньої межі. Значення stdev обчислюється в процесі.

У методі calculate() на основі змін ціни за останні 20 днів обчислюються значення простий ковзної середньої, стандартного відхилення та дисперсії.

```
class BollingerBands:

    sma_values = []
    upper_band = []
    lower_band = []
    time_period = 0
    stdev_factor = 0

    def __init__(self,time_period = 20, stdev_factor = 2):
        self.dataLoader = DataLoader()
        self.dataLoader.load()
        self.close = self.dataLoader.close_data['series']

        self.time_period = time_period
        self.stdev_factor = stdev_factor


    def calculate(self):
        history = []
        for close_price in self.close:
            history.append(close_price)
            if (len(history) > self.time_period):
                del(history[0])

            sma = stats.mean(history)
            self.sma_values.append(sma)

            variance = 0
            for hist_price in history:
                variance = variance + ((hist_price - sma) ** 2)

            stdev = math.sqrt(variance / len(history))

            self.upper_band.append(sma + self.stdev_factor * stdev)
            self.lower_band.append(sma - self.stdev_factor * stdev)
```

Рисунок 3.10 – Реалізація класу BollingerBands

Дивлячись на графік BollingerBands можна винести кілька висновків: якщо ціна залишається в межах верхньої і нижньої межі, то ніяких суттєвих змін очікувати не доводиться; якщо ж графік ціни перетинає верхню межу, одним з варіантів буде продовження руху ціни вгору, або це може свідчити про перекупленності ринку і поверненню ціни в межі кордонів. Коли графік ціни перетинає нижню межу в разі перепроданості ринку очікується повернення ціни в колишнє русло, або продовження руху вниз.

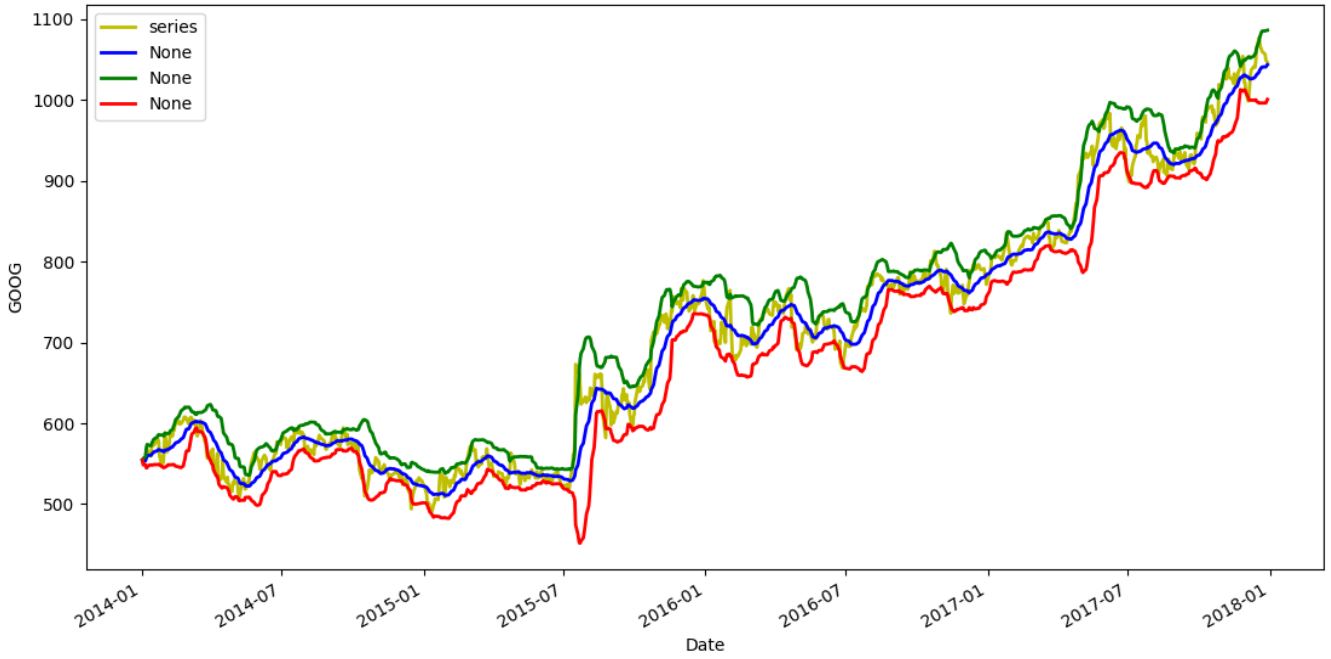

Рисунок 3.11 – Результат роботи классу BollingerBands

3.2.5 Relative Strength Index

Для даної імплементації RelativeStrengthIndex взято тимчасове вікно в 20 днів. В ході виконання методу calculate() проводиться запис історії значень прибутку і втрат а також обчислення їх середнього значення. Ці дані в подальшому були нормалізовані і приведені до значень від 0 до 100.

```
def __init__(self,time_period = 20):
    self.dataLoader = DataLoader()
    self.dataLoader.load()
    self.close = self.dataLoader.close_data['series']

    self.time_period = time_period

def calculate(self):
    gain_history = []
    loss_history = []
    last_price = 0

    for close_price in self.close:
        if last_price == 0:
            last_price = close_price

        gain_history.append(max(0, close_price - last_price))
        loss_history.append(max(0, last_price - close_price))
        last_price = close_price

        if (len(gain_history) > self.time_period):
            del(gain_history[0])
            del(loss_history[0])

        avg_gain = stats.mean(gain_history)
        avg_loss = stats.mean(loss_history)

        self.avg_gain_values.append(avg_gain)
        self.avg_loss_values.append(avg_loss)

        rs = 0
        if avg_loss > 0:
            rs = avg_gain / avg_loss

        rsi = 100 - (100 / (1 + rs))
        self.rsi_values.append(rs)
```

Рисунок 3.12 – Реалізація класу RelativeStrengthIndex

Візуалізація даних вказує на те, що в разі, коли середня прибуток перевищує середнє значення втрат – графік буде перевищувати значення 50 і свідчити про стійке зростання ціни. В іншому випадку він буде залишатися нижче цієї межі або незмінно повертатися в її нижні межі.

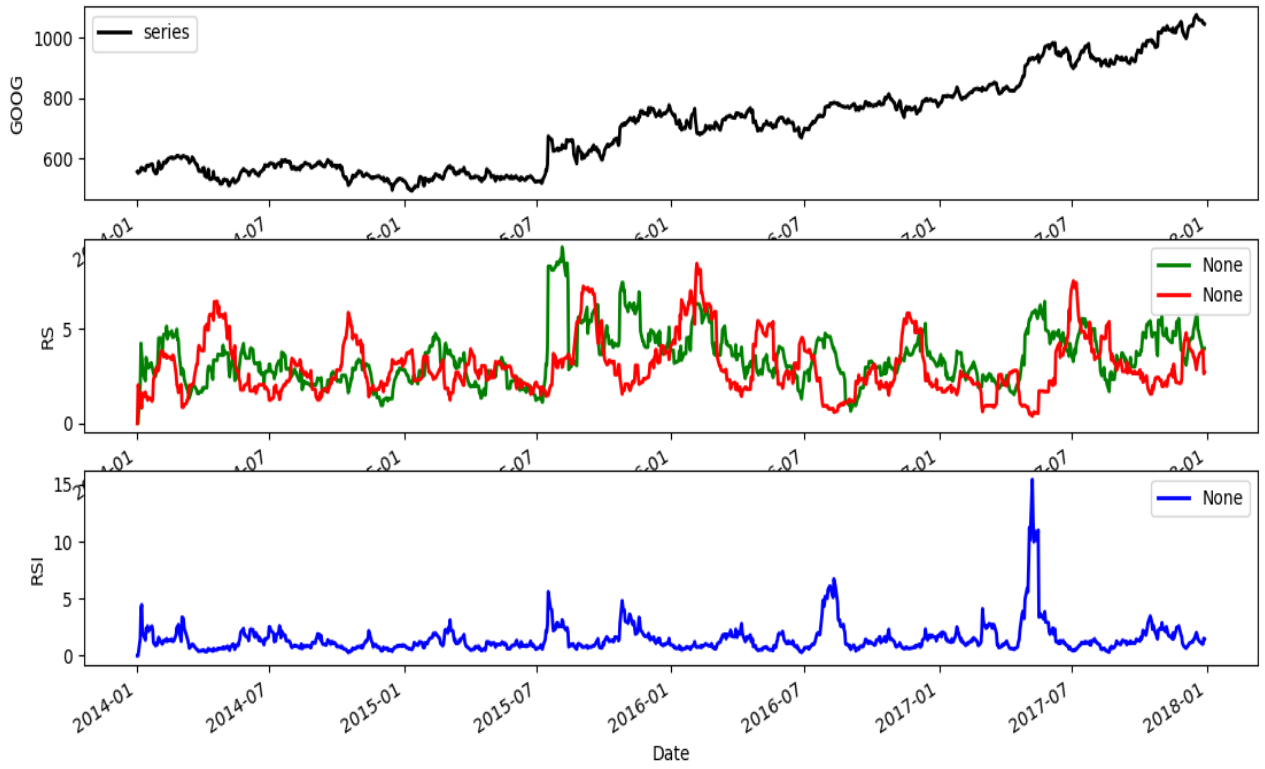

Рисунок 3.13 – Результат роботи классу RelativeStrengthIndex

3.2.6 Standard Deviation

У реалізації StandardDeviation в якості стандартного значення періоду зворотного відгуку використовується значення в 20 днів. На основі цього параметра обчислюються значення стандартної ковзної середньої. У свою чергу для кожного значення SMA обчислюється стандартне відхилення і значення дисперсії.

```
class StandartDeviation:

    sma_values = []
    stdev_values = []


    def __init__(self,time_period = 20):
        self.dataLoader = DataLoader()
        self.dataLoader.load()
        self.close = self.dataLoader.close_data['series']

        self.time_period = time_period


    def calculate(self):
        history = []

        for close_price in self.close:
            history.append(close_price)
            if len(history) > self.time_period:
                del(history[0])

            sma = stats.mean(history)
            self.sma_values.append(sma)

            variance = 0

            for hist_price in history:
                variance = variance + ((hist_price - sma) ** 2)

            stdev = math.sqrt(variance / len(history))
            self.stdev_values.append(stdev)
```

Рисунок 3.14 – Реалізація класу StandardDeviation

На графіку видно, що стандартне відхилення кількісно характеризує волатильність руху ціни за останні 20 днів. Волатильність різко зростає, коли ціни на акції різко підвищуються або падають, або зазнають великі зміни за останні 20 днів.

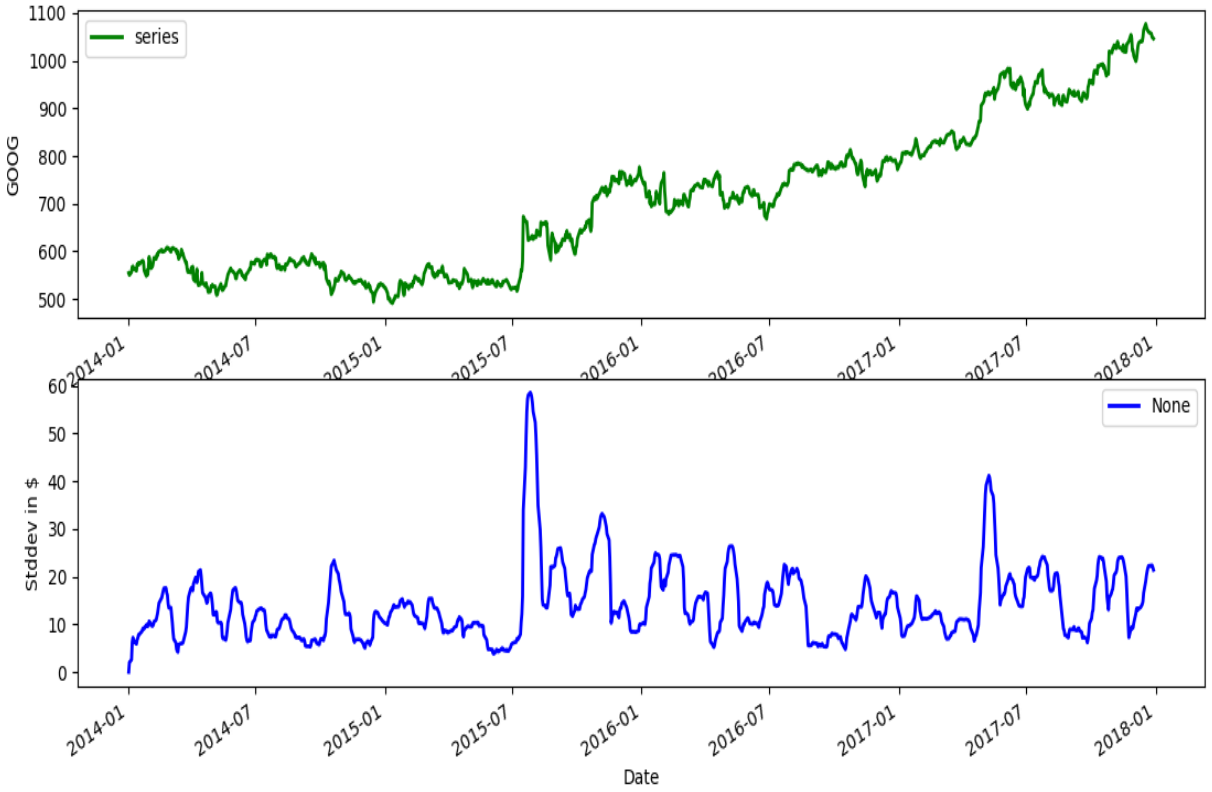

Рисунок 3.15 – Результат роботи классу StandartDeviation

3.2.7 Momentum

У простій формі імпульс є різницею між поточною ціною і ціною в певний час в минулому. Послідовні періоди позитивної динаміки позначають висхідний тренд і навпаки, якщо імпульс послідовно негативний – тренд є низхідним. Часто використовуються елементарні або експоненціальні ковзаючі середні на основі індикатора імпульсу для визначення стійкого тренда.

```
class Momentum:

    mom_values = []


    def __init__(self,time_period = 20):
        self.dataLoader = DataLoader()
        self.dataLoader.load()
        self.close = self.dataLoader.close_data['series']

        self.time_period = time_period

    def calculate(self):
        history = []

        for close_price in self.close:
            history.append(close_price)
            if len(history) > self.time_period:
                del(history[0])

            mom = close_price - history[0]
            self.mom_values.append(mom)


    def visualise(self):
        mom_series = pd.Series(self.mom_values, index=self.dataLoader.close_data.index)
        fig = plt.figure()
        ax1 = fig.add_subplot(211, ylabel='EUR/USD')
        self.close.plot(ax=ax1, color='g', lw=2., legend=False)
        ax2 = fig.add_subplot(212, ylabel='Momentum in $')
        mom_series.plot(ax=ax2, color='b', lw=2., legend=False)
        plt.show()
```

Рисунок 3.16 – Реалізація классу Momentum

Пік значень імпульсу настає коли ціна акцій змінюється на значну величину в порівнянні з ціною 20 днів тому. У моменти, коли акції падають в ціні, спостерігаються негативні значення імпульсу.

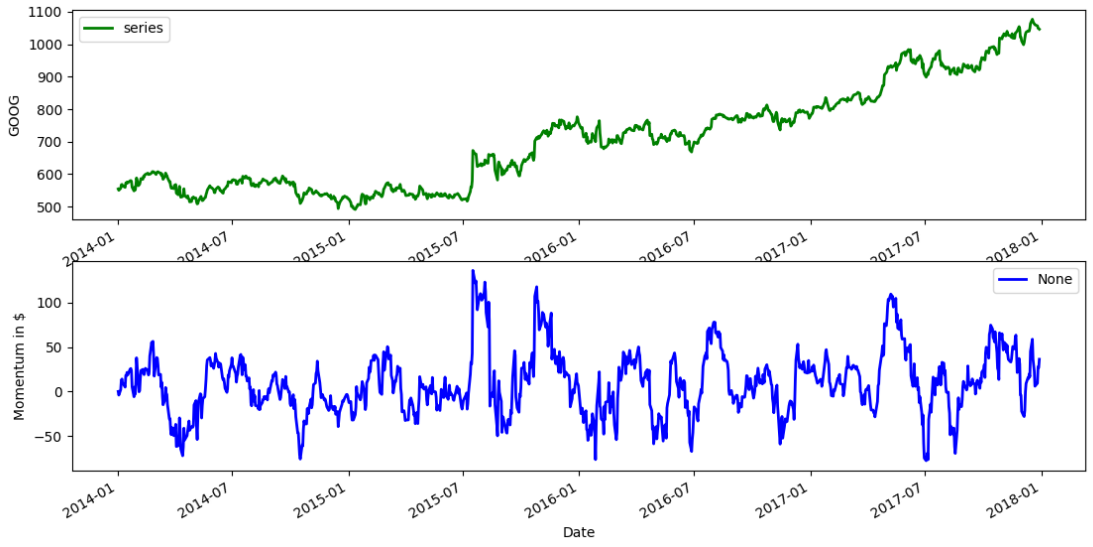

Рисунок 3.17 – Результат роботи классу Momentum

3.3 Реалізація торгової стратегії

У цьому розділі реалізовано стратегію повернення до середнього, засновану на абсолютному ціновому осцилляторі (APO). Він використовує статичну константу 10 днів для швидкої EMA і статичну константу 40 днів для повільної EMA. Стратегія здійснює операції на покупку, коли значення сигналу APO опускається нижче -10, і здійснювати операції на продаж, коли значення сигналу APO перевищує +10. Крім того, перевіряється чи відбуваються нові угоди за цінами, відмінним від ціни останньої угоди, щоб запобігти надмірній торгівлі. Позиції закриваються, коли значення сигналу APO змінює знак, тобто закриваються короткі позиції, коли APO стає негативним, і закриваються довгі позиції, коли APO стає позитивним.

Крім того, позиції також закриваються, якщо поточні відкриті позиції є прибутковими вище певної суми, незалежно від значень APO. Це використовується для алгоритмічної фіксації прибутку і відкриття більшої кількості позицій замість того, щоб покладатися тільки на значення торгового сигналу.

Історичні дані завантажуються за допомогою описаного вище класу DataLoader. Був обраний часовий проміжок з 01.01.2014 по 01.01.2020. Як торговий інструменту використовуються акції компанії Google (GOOG). Джерелом даних виступає сервіс Yahoo Finance.

На рисунку 3.18 визначаються деякі константи і змінні, необхідні для виконання розрахунків швидкої і повільної EMA і торгового сигналу APO:

```
# Variables/constants for EMA Calculation:
NUM_PERIODS_FAST = 10 # Static time period parameter for the fast EMA
K_FAST = 2 / (NUM_PERIODS_FAST + 1) # Static smoothing factor parameter for fast EMA
ema_fast = 0
ema_fast_values = [] # we will hold fast EMA values for visualization purposes

NUM_PERIODS_SLOW = 40 # Static time period parameter for slow EMA
K_SLOW = 2 / (NUM_PERIODS_SLOW + 1) # Static smoothing factor parameter for slow EMA
ema_slow = 0
ema_slow_values = [] # we will hold slow EMA values for visualization purposes

apo_values = [] # track computed absolute price oscillator value signals
```

Рисунок 3.18 – Присвоєння значень змінним для обчислення EMA

Також потрібні змінні, які визначають/контролюють поведінку торгової стратегії і управління позицією і прибутком (див. рис. 3.19):

```
# Variables for Trading Strategy trade, position & pnl management:
orders = [] # Container for tracking buy/sell order, +1 for buy order,
            #-1 for sell order, 0 for no-action
positions = []  # Container for tracking positions, positive for long positions,
                #negative for short positions, 0 for flat/no position
pnls = []   # Container for tracking total_pnls, this is the sum of closed_pnl
            #i.e. pnls already locked in and open_pnl i.e. pnls for open-position marked to market price
last_buy_price = 0  # Price at which last buy trade was made,
                    #used to prevent over-trading at/around the same price
last_sell_price = 0 # Price at which last sell trade was made,
                    #used to prevent over-trading at/around the same price
position = 0 # Current position of the trading strategy
buy_sum_price_qty = 0   # Summation of products of buy_trade_price and buy_trade_qty
                        #for every buy Trade made since last time being flat
buy_sum_qty = 0 # Summation of buy_trade_qty for every buy
                #Trade made since last time being flat
sell_sum_price_qty = 0  # Summation of products of sell_trade_price and sell_trade_qty
                        #for every sell Trade made since last time being flat
sell_sum_qty = 0 # Summation of sell_trade_qty for every sell
                  #Trade made since last time being flat
open_pnl = 0 # Open/Unrealized PnL marked to market
closed_pnl = 0 # Closed/Realized PnL so far
```

Рисунок 3.19 – Присвоєння значень змінним для роботи торгової стратегії

Нарешті, визначаються пороги входу, мінімальна зміна ціни з моменту останньої угоди, мінімальну очікувану прибуток за угоду і кількість акцій для торгівлі за угоду:

```
# Constants that define strategy behavior/thresholds
APO_VALUE_FOR_BUY_ENTRY = -10 # APO trading signal value below which
                              #to enter buy-orders/long-position
APO_VALUE_FOR_SELL_ENTRY = 10 # APO trading signal value above which
                              #to enter sell-orders/short-position
MIN_PRICE_MOVE_FROM_LAST_TRADE = 10 # Minimum price change since last trade
                                    #before considering trading again, this is
                                    #to prevent over-trading at/around same prices
MIN_PROFIT_TO_CLOSE = 10 # Minimum Open/Unrealized profit at which
                         #to close positions and lock profits
NUM_SHARES_PER_TRADE = 10 # Number of shares to buy/sell on every trade
```

Рисунок 3.20 – Присвоєння значень змінним для поведінки торгової стратегії

Основний розділ торгової стратегії, має наступні функції:

- розрахунок/оновлення швидкої і повільної EMA і торгового сигналу APO;
- реагування на торгові сигнали для відкриття довгих або коротких позицій;
- реагування на торгові сигнали, відкриті позиції, відкриті PnL і ринкові ціни для закриття довгих або коротких позицій.

```
print(self.close.head())
for close_price in self.close:
    # This section updates fast signal and slow EMA and computes APO trading
    if (ema_fast == 0): # first observation
        ema_fast = close_price
        ema_slow = close_price
    else:
        ema_fast = (close_price - ema_fast) * self.K_FAST + ema_fast
        ema_slow = (close_price - ema_slow) * self.K_SLOW + ema_slow

    self.ema_fast_values.append(ema_fast)
    self.ema_slow_values.append(ema_slow)

    apo = ema_fast - ema_slow
    self.apo_values.append(apo)
```

Рисунок 3.21 – Обчислення EMA та APO значень

Код перевіряє торгові сигнали на відповідність торговим параметрам/порогам і позиціям. Угода на продаж по close_price надається, якщо виконуються наступні умови:

- значення торгового сигналу APO вище порога входу на продаж, і різниця між ціною останньої угоди і поточною ціною достатньо велика;

- поточна позиція довга (позитивна позиція), значення торгового сигналу APO 0 або вище, або поточна позиція досить прибуткова, щоб зафіксувати прибуток.

```
# long from negative APO and APO has gone positive or position is profitable, sell to close position
if ((apo > self.APO_VALUE_FOR_SELL_ENTRY
     and abs(close_price - last_sell_price) > MIN_PRICE_MOVE_FROM_LAST_TRADE)
     or (position > 0 and (apo >= 0 or open_pnl > MIN_PROFIT_TO_CLOSE))):

    self.orders.append(-1) # mark the sell trade
    last_sell_price = close_price
    position -= self.NUM_SHARES_PER_TRADE # reduce position by the size of this trade
    sell_sum_price_qty += (close_price*self.NUM_SHARES_PER_TRADE) #update vwap sell-price
    sell_sum_qty += self.NUM_SHARES_PER_TRADE
    print( "Sell ", self.NUM_SHARES_PER_TRADE, " @ ", close_price, "Position: ", position )
```

Рисунок 3.22 –Код, що здійснює операцію на покупку

Операція на покупку по close_price здійснюється, якщо виконуються наступні умови: значення торгового сигналу APO нижче порога Buy-Entry і різниця між ціною останньої угоди і поточною ціною досить різна. У разі короткої позиції (негативна позиція), і значенні торгового сигналу APO 0 або нижче, або поточна позиція досить прибуткова, щоб зафіксувати прибуток.

```
# short from positive APO and APO has gone negative or position is profitable, buy to close position
elif ((apo < self.APO_VALUE_FOR_BUY_ENTRY
       and abs(close_price - last_buy_price) > MIN_PRICE_MOVE_FROM_LAST_TRADE)
      or (position < 0 and (apo <= 0 or open_pnl > MIN_PROFIT_TO_CLOSE))):
    self.orders.append(+1) # mark the buy trade
    last_buy_price = close_price
    position += self.NUM_SHARES_PER_TRADE # increase position by the size of this trade
    buy_sum_price_qty += (close_price*self.NUM_SHARES_PER_TRADE) #update the vwap buy-price
    buy_sum_qty += self.NUM_SHARES_PER_TRADE
    print( "Buy ", self.NUM_SHARES_PER_TRADE, " @ ", close_price, "Position: ", position )
else:
    # No trade since none of the conditions were met to buy or sell
    self.orders.append(0)

self.positions.append(position)
```

Рисунок 3.23 – Код, що здійснює операцію на продаж

Код торгової стратегії містить логіку управління позицією/PnL. Йому необхідно оновлювати позиції і обчислювати відкриті і закриті PnL, коли ринкові ціни змінюються і/або укладаються угоди, що викликають зміну позицій (див. рис. 3.24).

```
# This section updates Open/Unrealized & Closed/Realized positions
open_pnl = 0
if position > 0:
    # long position and some sell trades have been made against it,
    #close that amount based on how much was sold against this long position
    if sell_sum_qty > 0:
        open_pnl = abs(sell_sum_qty) * (sell_sum_price_qty/sell_sum_qty - buy_sum_price_qty/buy_sum_qty)
    # mark the remaining position to market i.e.
    #pnl would be what it would be if we closed at current price
    open_pnl += abs(sell_sum_qty - position) * (close_price - buy_sum_price_qty / buy_sum_qty)
elif position < 0:
    # short position and some buy trades have been made against it,
    #close that amount based on how much was bought against this short position
    if buy_sum_qty > 0:
        open_pnl = abs(buy_sum_qty) * (sell_sum_price_qty/sell_sum_qty - buy_sum_price_qty/buy_sum_qty)
    # mark the remaining position to market i.e.
    #pnl would be what it would be if we closed at current price
    open_pnl += abs(buy_sum_qty - position) * (sell_sum_price_qty/sell_sum_qty - close_price)
else:
    # flat, so update closed_pnl and reset tracking variables for positions & pnls
    closed_pnl += (sell_sum_price_qty - buy_sum_price_qty)
    buy_sum_price_qty = 0
    buy_sum_qty = 0
    sell_sum_price_qty = 0
    sell_sum_qty = 0
    last_buy_price = 0
    last_sell_price = 0
self.pnls.append(closed_pnl + open_pnl)
```

Рисунок 3.24 – Код логіки управління позицією/PnL

На рисунках 3.25 та 3.26 наведено частки коду, що визуалізують результати торгової стратегії, такі як ринкові ціни, значення швидкої і повільної EMA, значення APO, угоди на купівлю і продаж, позиції і PnL, досягнуті стратегією за час її існування. Також додано стовпці даних з різними рядами, що було вирахувано в попередніх розділах, спочатку ринкову ціну, а потім значення швидкої і повільної EMA. Також окремо будується інший графік для значення торгового сигналу APO. На обох графіках наведено угоди на купівлю і продаж.

```
self.data['ClosePrice'].plot(color='blue', lw=3., legend=True)
self.data['Fast10DayEMA'].plot(color='y', lw=1., legend=True)
self.data['Slow40DayEMA'].plot(color='m', lw=1., legend=True)
plt.plot(self.data.loc[ self.data.Trades == 1 ].index,
         self.data.ClosePrice[self.data.Trades == 1 ],
         color='r', lw=0, marker='^', markersize=7, label='buy')
plt.plot(self.data.loc[ self.data.Trades == -1 ].index,
         self.data.ClosePrice[self.data.Trades == -1 ],
         color='g', lw=0, marker='v', markersize=7, label='sell')
plt.legend()
plt.show()

self.data['APO'].plot(color='k', lw=3., legend=True)
plt.plot(self.data.loc[ self.data.Trades == 1 ].index,
         self.data.APO[self.data.Trades == 1 ],
         color='r', lw=0, marker='^', markersize=7, label='buy')
plt.plot(self.data.loc[ self.data.Trades == -1 ].index,
         self.data.APO[self.data.Trades == -1 ],
         color='g', lw=0, marker='v', markersize=7, label='sell')
plt.axhline(y=0, lw=0.5, color='k')
for i in range( self.APO_VALUE_FOR_BUY_ENTRY,
                self.APO_VALUE_FOR_BUY_ENTRY*5,
                self.APO_VALUE_FOR_BUY_ENTRY ):
    plt.axhline(y=i, lw=0.5, color='r')
for i in range( self.APO_VALUE_FOR_SELL_ENTRY,
                self.APO_VALUE_FOR_SELL_ENTRY*5,
                self.APO_VALUE_FOR_SELL_ENTRY ):
    plt.axhline(y=i, lw=0.5, color='g')
plt.legend()
plt.show()
```

Рисунок 3.25 – Код відображення покупок та продажей та графіку APO

```
self.data['Position'].plot(color='k', lw=1., legend=True)
plt.plot(self.data.loc[ self.data.Position == 0 ].index,
         self.data.Position[self.data.Position == 0 ],
         color='k', lw=0, marker='.', label='flat')
plt.plot(self.data.loc[ self.data.Position > 0 ].index,
         self.data.Position[self.data.Position > 0 ],
         color='r', lw=0, marker='+', label='long')
plt.plot(self.data.loc[ self.data.Position < 0 ].index,
         self.data.Position[self.data.Position < 0 ],
         color='g', lw=0, marker='_', label='short')
plt.axhline(y=0, lw=0.5, color='k')
for i in range( self.NUM_SHARES_PER_TRADE,
                self.NUM_SHARES_PER_TRADE*25,
                self.NUM_SHARES_PER_TRADE*5 ):
    plt.axhline(y=i, lw=0.5, color='r')
for i in range( -self.NUM_SHARES_PER_TRADE,
                -self.NUM_SHARES_PER_TRADE*25,
                -self.NUM_SHARES_PER_TRADE*5 ):
    plt.axhline(y=i, lw=0.5, color='g')
plt.legend()
plt.show()

self.data['Pnl'].plot(color='k', lw=1., legend=True)
plt.plot(self.data.loc[ self.data.Pnl > 0 ].index,
         self.data.Pnl[ self.data.Pnl > 0 ],
color='g', lw=0, marker='.')
plt.plot(self.data.loc[ self.data.Pnl < 0 ].index,
         self.data.Pnl[ self.data.Pnl < 0 ],
color='r', lw=0, marker='.')
plt.legend()
plt.show()
```

Рисунок 3.26 – Код відображення позицій і графіків PnL

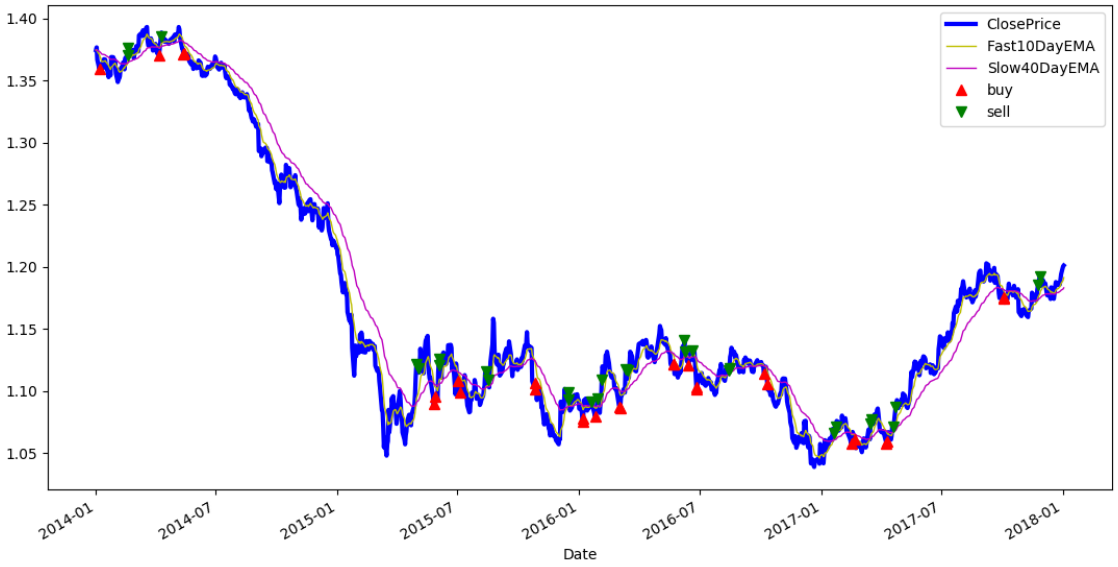

Рисунок 3.27 – Графік ціни з угодами на продаж або купівлю

На графіку можна бачити, де були здійснені угоди на купівлю і продаж при зміні ціни акцій Google за останні роки. Відповідно до правил торгової стратегії очікується угоди на продаж, коли значення APO є позитивними, і очікується угоди на покупку, коли значення APO негативні (див. рис. 3.27):

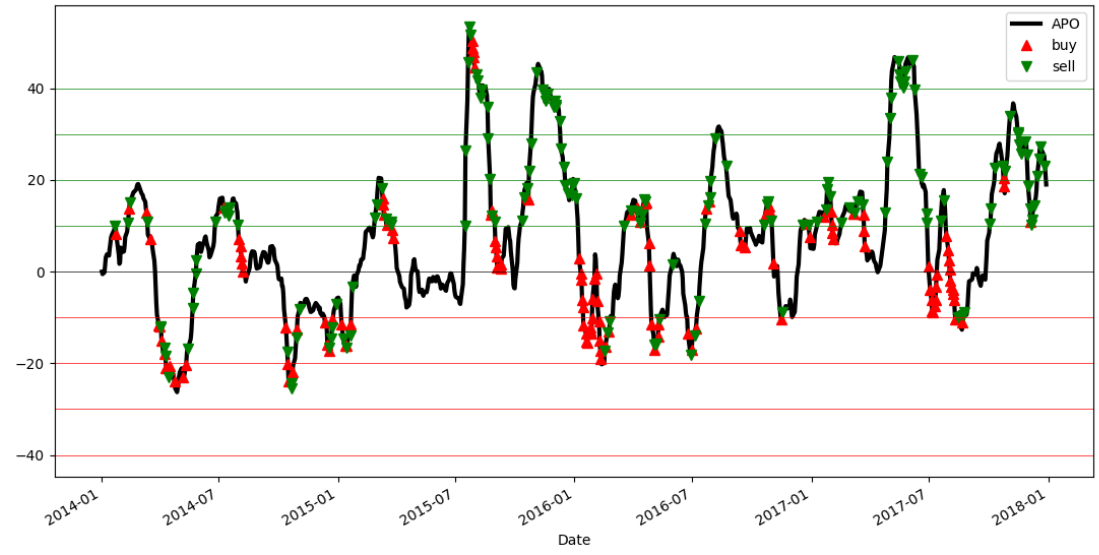

Рисунок 3.28 – Графік APO

На графіку можна бачити, що безліч угод на продаж виконується, коли значення торгового сигналу APO позитивні, і багато угод на покупку виконується, коли значення торгового сигналу APO негативні. Також спостерігається, що деякі угоди на купівлю виконуються, коли значення торгового сигналу APO позитивні, а деякі угоди на продаж виконуються, коли значення торгового сигналу APO негативні.

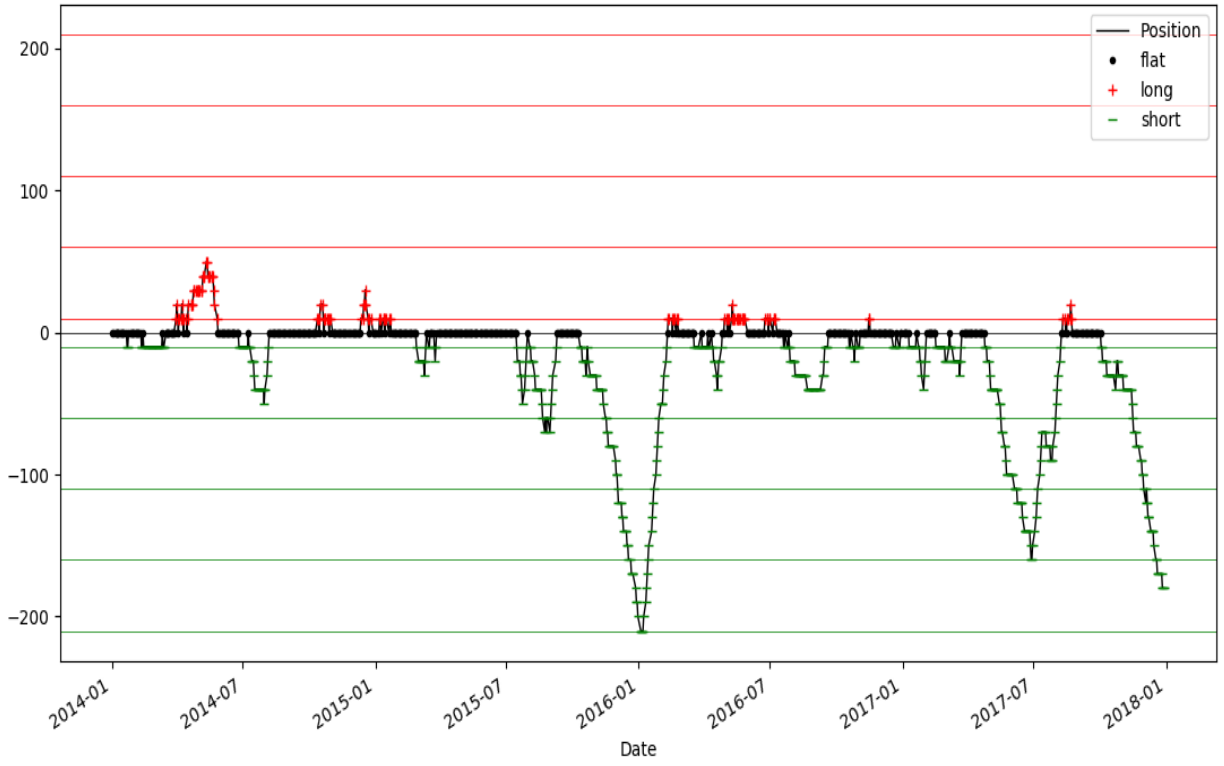

Рисунок 3.29 – Графік позицій

На графіку позицій можна побачити кілька великих коротких позицій в період 2016-01, потім знову в 2017-07 і, нарешті, знову в 2018-01.

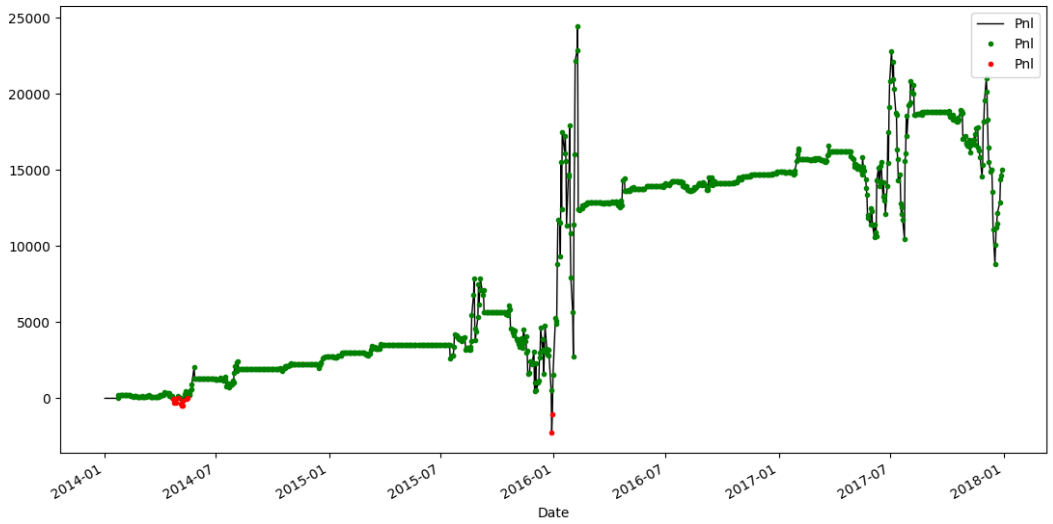

Рисунок 3.30 – Графік PnL

Базова стратегія повернення до середнього значення робить гроші досить стабільно з плином часу, з деякою волатильністю прибутковості протягом 2016-01 і 2017-07 років, коли стратегія має великі позиції, але в кінцевому підсумку закінчується на рівні близько 15 тисяч доларів, що близько до досягнутого максимуму PnL.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи проведено огляд теоретичних аспектів створення торгової стратегії та сформульовано мету та актуальність роботи.

Виконано проектування бібліотеки прогнозування фінансових часових рядів основною особливістю якої є використання технічних індикаторів і основних принципів алгоритмічної торгівлі при прогнозуванні руху ціни.

У відповідності до розробленого проекту реалізовано та протестовано високорівневу Python бібліотеку, яка може застосовуватись як модуль у автоматизованій системі торгів на ринку валют та цінних паперів. Проведено ряд обчислювальних експериментів та продемонстровано високу прибутковість створеної торгової стратегії на прикладі акцій компанії Google.

Перевагою розробленої бібліотеки є уніфікована можливість використання різних технічних індикаторів і наявність функціоналу, що дозволяє тестувати і застосовувати на історичних даних створену торгову стратегію.

Розроблена бібліотека дозволяє налаштовувати велику кількість параметрів, які впливають на поведінку технічних індикаторів, що, в свою чергу, дозволяє показувати й аналізувати дані на різних часових проміжках.

Перспективи подальших досліджень пов'язані з розширенням функціоналу бібліотеки за допомогою більшої кількості технічних індикаторів і допоміжних інструментів, що значно дозволить полегшити завдання аналізу і прогнозування фінансових часових рядів а також спростить завдання контролю ризиків і прибутку. Також необхідним розвитком даної бібліотеки буде додавання модуля, що дозволяє працювати з біржею безпосередньо і в реальному часі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Sebastien Donadio, Sourav Ghosh. Learn Algorithmic Trading. Birmingham - Mumbai: Packt Publishing, 2019. 378 p.
2. Stefan Jansen. Hands-On Machine Learning for Algorithmic Trading. Birmingham - Mumbai: Packt Publishing, 2018. 503 p.
3. Вендров А.М. Проектирование программного обеспечения экономических информационных систем. Москва, 2006. 545 с.
4. Yan H., Ouyang H. Financial Time Series Prediction Based on Deep Learning. // *Wireless Pers Commun*. 2018, Vol. 102. Issue 2, P. 683–700. DOI : https://doi.org/10.1007/s11277-017-5086-2.
5. Catania L., Grassi S. Modelling Crypto–Currencies Financial Time–Series. 2017. DOI : 10.2139/ssrn.3028486.
6. Dingli A., Fournier K.S. Financial Time Series Forecasting – A Machine Learning Approach. // *Machine Learning and Applications : An International Journal (MLAIJ)*. 2017, Vol.4, No.1/2/3.
7. Kavitha S., Raja Vadhana P., Nivi A.N. Big Data Analytics In Financial Market. // *International Journal of Research in Engineering and Technology*. 2015, Vol. 04, Issue 02. P. 422–427.
8. Kapila Tharanga Rathnayaka R.M., Seneiratna D.M.K.N, Arumawadu H.I. A New Financial Time Series Approach for Volatility Forecasting. // *Symposium on Statistical & Computational Modelling with Applications*. 2016, P. 5–8.
9. Barbulescu A. Bautu E. A Hybrid Approach for Modeling Financial Time Series. // *The International Arab Journal of Information Technology*. 2012, Vol. 9, N. 4, P. 327–335.
10. Leng J. Modelling and Analysis on Noisy Financial Time Series. // *Journal of Computer and Communications*. 2014, Vol. 2, P. 64–69.

11. Chakraborti A., Patriarca M., Santhanam M.S. Financial time-series analysis : A brief overview. 2007. URL: https://arxiv.org/abs/0704.1738 (дата звернення: 20.09.2020).
12. Brockwell P.J., Davis R.A. Introduction to Time Series and Forecasting. New York : Springer, 2016. 425 p.