КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет комп'ютерних наук та кібернетики

Кафедра Моделювання Скаладних Систем

Кваліфікаційна робота

на здобуттяступеня бакалавра

за спеціальністю 113Приклана математика

на тему:

Технічний аналіз знаходження знакових рівнів та трендових ліній

Виконав студент 4 курсу
Астахов Максим Вячеславович _____
(підпис)

Науковийкерівник:
доцент
Шатирко Андрій Володимирович _____
(підпис)

Засвідчую, що в ційроботінемаєзапозичень з працьіншихавторів без відповіднихпосилань.

Студент _____
(підпис)

Роботу розглянуто й допущено до захисту на засіданні кафедри моделювання складних систем
«_____»_____202__р.
протокол №_____

Завідувач катедри

Черный Д.І. _____
(підпис)

Київ – 2021

РЕФЕРАТ

Обсяг роботи 22 сторінок, 3 ілюстрацій, 7 джерел посилань

ТЕХНІЧНИЙ АНАЛІЗ, ТРЕНДОВІ ЛІНІЇ, ОБ’ЄМ НА БІРЖІ, МЕТОД ВІДСОРТОВАНОГО СХИЛУ, МЕТОД ЛІНІЙНОГО ПЕРЕТВОРЕННЯ ХАФА,РОЗРОБКА ПРОГРАМНОГО ЗАСТОСУНКУ.

Обєктом роботи є процес прогнозування зміни ціни на біржі за допомогою технічного аналізу.Предметом роботи є розробка додатку для побудови значимих трендових прямих.

Метою роботи є розробка програмного застосунку для побудови значимих трендових ліній.

Методи розроблення: розробка програмного застосунку, аналіз існуючих алгоритмів побудови трендових ліній, знаходження значимих тренових ліній.

Інструмени розроблення. Для реалізації програми було використано мову програмування Python3.При розробці було використано інтегроване середовище PyCharm.

Результат роботи. Було проаналізовано існуючі алгоритми побудови трендових прямих та їх зв'язок з об’ємами. Було розроблено за стосунок для знаходження значимих трендових прямих

ЗМІСТ

РЕФЕРАТ 2
ВСТУП 4
РОЗДІЛ 1 6
БІРЖА ТА МЕТОДИ ДОСЛІДЖЕННЯ ЦІН 6
1.1 Біржа та її види 6
1.2 Трендові лінії 6
1.3 Об’єм 7
РОЗДІЛ 2 10
ЛІНІЇ ТРЕНДУ ТА ПРЯМІ ПІДТРИМКИ 10
2.1 Зворотні точки 10
2.1.1 Усі максимуми та мінімуми наївним методом 11
2.1.2 Чисельне диференціювання 11
2.2 Методи лінії тренду 12
2.2.1 Наївний метод 13
2.2.2 Метод відсортованого схилу 13
2.2.3 Метод лінійного перетворення Хафа 14
2.2.4 Метод імовірнісного лінійного перетворення Хафа 16
2.3 Пошук значимих ліній тренду 16
РОЗДІЛ 3 18
РОЗРОБКА ПРОГРАМНОГО КОДУ 18
3.1 Отримання даних 18
3.2 Пошук мінімумів та максимумів 18
3.3 Лінійна регресія 19
3.4 Метод лінійного перетворення Хафа 19
3.5 Метод імовірнісного лінійного перетворення Хафа 20
3.6 Пошук значимих прямих тренду 20
ВИСНОВКИ 21
ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ 22
ДОДАТОК А 23

ВСТУП

Оцінка сучасного стану об'єкта дослідження. На данний момент біржа так чи інакше стосується життя кожно з нас, оскільки державні, страхові, пенсійні фонди є учасниками бірж. Ці учасники ринку мають великий капітал і є основною причиною руху ціни. Для того щоб якісно та ефективно дослідити закономірності змін руху ціни потрібно застосовувати математичний апарат, наприклад одними з найпопулярніших методів дослідження є технічний аналіз, гіпотеза ефективного ринку та фундаментальний аналіз.

Актуальність роботи та підстави для її виконання. Біржа існує уже більше чотирьохсот років. Велика кількість учасників торгівлі у довгостроковій перспективі втрачають свої кошти торгуючи на біржі. Для того щоб отримувати прибуток потрібно робити певний аналіз цін. Моя робота полягає в проведенні графічного аналізу цін для будь-якого активу. Графічний аналіз застосовується для того, щоб прийняти рішення засноване на ймовірності для прогнозування подальшї зміни ціни з метою отримання прибутку.

Мета роботи поолягає в тому, щоб на основі історичних данних побудувати прямі: підтримки і опору та трендові лінії. Для досягнення цієї мети було поставлено такі **завдання**:

- Дослідити існуючі методи побудови прямих опору та підтримки
- Налаштувати сучасний математичний апарат для побудови цих прямих
- Розробити застосунок для створення ліній тренду та знадження серед них значимих.

Об'єкт, методи й засоби дослідження або розроблення. Обєктомроботи є процес прогнозування зміни ціни на біржі за допомогою технічного аналізу.

Предметом роботи є розробка додатку для побудови значимих трендових прямих.

Перед роботою над застосунком було проаналізовано уже існуючі методи побудови трендових ліній: наївний метод, меодом відсортованого схилу, звичайного та імовірнісного лінійного перетворення Хафа. Лінії тренду побудовані даними методами можна буде застосовувати для прогнозування цін ні активи в біржі.

Для реалізації програми було використано мову програмування Python3. Для полегшення роботи з даними було використано бібліотеки pandas inumpy, а для полегшення отримання результатів використано yfinance. При розробці було використано інтегроване середовище PyCharm.

Можливі сфери застосування. Програма може бути застосована для прогнозування цін на біржі, як на пряму так і в ролі помічника для трейдера, який буде будувати значимі трендові лінії. Ця програма може бути дуже корисною у сфері машинного навчання, у сфері прогнозування цін на біржі, вже даючи значимі трендові лінії.

РОЗДІЛ 1

БІРЖА ТА МЕТОДИ ДОСЛІДЖЕННЯ ЦІН

1.1 Біржа та її види

Біржа — організований торгівельний майданчик, на якому відбувається гуртова торгівля товарами або цінними паперва у вигляді стандартизованих біржових угод. На біржі укладаються угоди по біржових товарах, в результаті чого утворюється динаміка ціни тільки під впливом ринкового попиту та пропозиції, що дає змогу орієнтуватися учасникам ринку та прогнозувати хід торгів в майбутньому.[1]

Біржі поділяють на:

- Товарні
- Валютні
- Фондові
- Ф’ючерсні
- Криптовалютні
- Універсальні

В залежності від того, яким видом активів торгую біржа.

1.2 Трендові лінії

Тренд – це загальний напрям в якому рухається інструмент(в нашому випадку ціна на біржі). Тенденції можуть бути зростаючими, спадаючими та боковими. Не існує конкретного графіку часу для того щоб напрям вважався трендом, але, в цілому, чим більше воно просувається, тим більш явним стає тренд.

Зростаючій тренд з’являється коли ціна активу послідовно досягає більш високих максимумів та мінімумів (ціна оновлює максимум, але не оновлює мінімум), тоді як спадаючий тренд з’являється, коли ціна доходить до більш низьких максимумів(оновлює мінімум, але не оновлює максимум).

Рис. 1. Зростаючий тренд

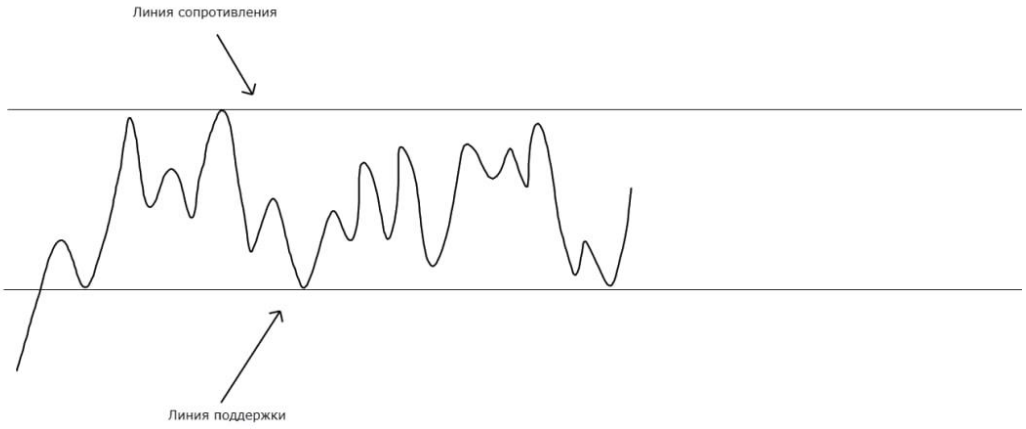

Рис. 2. Боковий тренд

1.3 Об’єм

Об’єм – це міра того, яка кількість заданого фінансового активу була проданою або купленою за певний період часу. Розглядаючи динаміку об’єму за деякий час, можна визначити силу покупа теля чи продавця. Також об’єм є необхідним фактором аналізу акумуляції та дистрибуції наприклад в теорії

Вайкоффа.. Аналіз об’єму зазвичай використовують для ідентифікації сили та слабості руху.

Об’єм може бути:

- Вертикальний. Він відображає кількість угод, здійснених протягом певного періоду часу. Вертикальний об'єм відображається, як правило, у нижній частині діаграми як стовпчаста діаграма.
- Горизонтальний. Він відображає кількість угод, здійснених за певного рівня цін. Горизонтальний об’єм або, як його ще називають, профіль обсягу може бути розташований як в лівій, так і в правій частинах діаграми (а деякі профілі можуть бути розташовані навіть посередині діаграми, як це буде показано в прикладах нижче).

Вертикальні обсяги використовувались торговцями ще на початку двадцятого століття. Представлення профілю ринку розвинулось і трансформувалося після введення електронних біржових торгів та потужного програмного забезпечення для аналізу ринку. Комп’ютери стали настільки потужними, що можуть обчислювати горизонтальні обсяги в режимі реального часу.

У зростаючому ринку має бути зростаючий об’єм. Збільшення ціни та зменшення об’єму можуть значити відсутність інтересу і це може бути попередженням про можливий розворот графіку ціни.

Хибний/істинний спад. На початку спаду з діапазону або іншого графічного паттерну збільшення об’єму свідчить про силу в русі,в той час як незначна зміна об’єму вказує на відсутність інтересу і більш високу вірогідність хибного спаду.

У випадку встановлення тренду, трендова лінія показує відносно найкращу ціну всередині руху. Розглядаючи рух ціни як взаємодію попиту та пропозиції або боротьби покупців та продавців можна зрозуміти, що з ціллю

спекуляції кожен прагне купити актив за найвигіднішою ціною та продати за найвищою. В такому випадку слідами активності капіталу або трейдерів буде об’єм. По цим слідам можливо встановити оптимальну ціну. Користуючись об’ємами можна встановити знакові рівні для великого капіталу. Ці рівні в майбутньому будуть відправною точкою для дій учасників ринку. Таким чином і будуть відрізнятися значимі прями від не значимих.

РОЗДІЛ 2.
ЛІНІЇ ТРЕНДУ ТА ПРЯМІ ПІДТРИМКИ

Аналіз лінії тренду для виявлення рівнів підтримки та опору традиційно робився економістами, проводячи вручну лінії на графіках, таких як графік ціни закриття для певного цінного папера. Комп’ютеризована автоматизація такого завдання широко не застосовується належним чином у багатьох бібліотеках.

Для початку такого завдання потрібно отримати об'єктивне визначення ліній тренду:

У фінансах лінія тренда є обмежувальною лінією для руху ціни цінних паперів. Він формується, коли діагональну лінію можна провести між мінімум трьома і більше точками розвороту ціни. Між будь-якими двома точками можна провести лінію, але вона не вважається лінією тренда, поки не буде перевірено. Звідси необхідність у третьому пункті – тесті.

Вимога до трьох і більше пунктів полягає в тому, що майже всі тривіальні спроби одразу відпадають. Тож розглянемо технічні деталі кожного цього аспекту. Це також може бути використано для ідентифікації різних схем діаграми, таких як трикутні фігури, які включають трикутники, клини, вимпели та паралельні фігури, такі як прапори, подвійні / потрійні вершини / низи, голова і плечі, прямокутники, хоча всі вони часто вимагають лише дві точки, але, безсумнівно, мали б посилену індикацію, якщо їх можна знайти з трьох, де це можливо.

2.1 Зворотні точки

Перше питання полягає у визначенні опорних точок. В подальшому, пунктами будуть ціни закриття на певний час. Пункти будуть позначатись на діаграмі як піки та западини, або як локальні максимуми та локальні мінімуми.

2.1.1 Усі максимуми та мінімуми наївним методом

Існує наївний спосіб зробити це, оскільки точка повороту вимагала б, щоб точка, що передує і досягає успіху, була нижчою або обидві вище поточної точки. У наївного методу є серйозні недоліки, однак, якби ціна залишалася незмінною протягом двох днів поспіль, не було б виявлено піку чи западини.

2.1.2 Чисельне диференціювання

Однак набагато більш науковий підхід до цього питання полягає у використанні числової похідної ціни закриття для ідентифікації пунктів. Перша похідна - це швидкість зміни, або фактично імпульс, або швидкість ціни закриття, тоді як друга похідна вказує на швидкість зміни першої похідної або її прискорення.

Звичайна диференціація тут не застосовується, оскільки дискретні часові ряди потребують дискретних інструментів чисельного аналізу. Є кілька переваг, включаючи те, що числова похідна згладить дані, враховуючи всі точки в заданому діапазоні від точки, в якій слід обчислити швидкість зміни.

При обчисленні першої та другої похідних фактично відбувся ступінь згладжування, що дало нам помітні вершини та западини. Вони є місцями, де перша похідна дорівнює нулю, оскільки жоден імпульс не вказує на зміну напрямку. Друга похідна, позитивна чи негативна, вказує на западину або пік, відповідно. Однак те, що точна перша похідна буде дорівнювати нулю дуже малоймовірна. Натомість значення на одній стороні нуль, за якою слідує інша - це реально те, що відбудеться, оскільки точка похідної нуль

відбулася між двома днями. Отже, виходячи з цього, вища або нижча ціна закриття буде обрана між двома точками, де сталась зміна напрямку функції.

2.2 Методи лінії тренду

Отже знайшовши усі точки розвороту, стає можливим побудувати n(n-1)(n-2), де n - кількість точок розвороту, ліній тренду,з деякою похибкою, яка з’являється через те, що не через будь-які три точки можна провести одну пряму. Розмір похибки дуже важливий, оскільки нам треба відфільтрувати усі неправильні набори точок і включити всі правильні.

Оскільки на графіку відбувається робота з площиною, лінії будуть задаватися рівнянням $y = kx + b$, де нас цікавить k як коефіцієнт нахилу прямої . Значення b обчислюється за однією з точок. Оскільки на графіку ціна ніколи не може дорівнювати нескінченності,то рівняння виду $x = c$, де c-константа утворитись не зможе.

Для трьох точок, якщо вписати лінію в дві точки, буде обчислюватися величина відстані від цієї лінії до третьої точки вздовж осі y. Однак це відстань як міра похибки досить умовна, оскільки воно буде різною в залежності від того, які дві точки будуть вибрані з трьох можливих варіантів, а саме: першу і останню або другу і третю. Тому для знаходження лінії яка знаходиться ближче всього о цих трьох заданих точок будемо використовувати лінійну регресію. Наводяться дві стандартні помилки: одна - помилка нахилу, інша - помилка перехоплення. Для нахилу помилка заснована на сумі квадратів залишків , що обчислюється частково шляхом ділення на кількість точок мінус дві, де кількість точок в даному випадку дорівнює трьом, що призводить до зникнення терміна, так що це просто квадратний корінь з суми квадратів різниць y над сумою квадратів різниць x. Помилка перехоплення - це помилка нахилу, скоригована на значення x.

Помилка нахилу є достатнім показником помилки, тому буде застосовуватися саме вона.

2.2.1 Наївний метод

Цей метод просто перебирає всі комбінації з трьох точок, щоб знайти відповідну помилку і відсіяти ті, в яких значення помилки дуже велика. Звичайно, O (n^3) не є ідеальним, і якби набір даних був досить великим, це було б навіть практично нездійсненно. Але загальна кількість точок мінімуму і загального максимуму на практиці буде не настільки велике, щоб це було неможливо. Наприклад, 100 поворотних точок - це 100 * 100 * 100 = 1,000,000 або мільйон обчислень.

2.2.2 Метод відсортованого схилу

Існують певні властивості точок, які можна співвіднести, наприклад, нахил лінії (або кут від початку координат). Виконавши O (n^2) обхід всіх комбінацій двох точок і обчисливши нахил лінії, яку вони утворюють, можна отримати список нахилів для кожної точки. Тепер сортування списку має найгіршу складність в ідеалі O ($n^2 \ln n$) для ефективного алгоритму сортування типу mergesort, а нам потрібно зробити це над усіма n списками за O ($n \ln n$), що і буде складністю алгоритму. Сусідні точки в відсортованому списку схилів можуть розглядатися суміжно від трьох до скількох завгодно точок, які продовжують відповідати критерію фільтрації. Як тільки вони збігаються, група видаляється, і пошук триває. Ця частина алгоритму також є O (n^2). (Фактор найбільшої складності зазвичай є єдиним, який потрібно враховувати, щоб зберегти формулу спрощеної).

Це приблизний алгоритм і не вичерпний, оскільки сортування по ухилу не гарантує, що сусідні значення ухилу матимуть найкращі відповідності, коли

між точками можуть бути великі відстані. Проте, на практиці сценарій, при якому це сталося б, зустрічається відносно рідко.

2.2.3 Метод лінійного перетворення Хафа

Існують альтернативні способи вирішення алгоритму пошуку ліній, і один з них прийшов з обробки зображень, де пошук ліній на зображеннях є поширеною і важливою задачею в комп'ютерному зорі. Одним з таких методів є перетворення ліній Хафа, яке шукає точки, які перетинають лінії під певними кутами, і оцінює їх за кількістю знайдених точок. Цей алгоритм також має обмеження. На жаль, він використовує багато пам'яті для відстеження всіх гістограм, а його точність залежить від того, скільки кутів було випробувано. Однак чим більше кутів, тим повільніше працює програма. Пам'ять заснована на розмірі діагоналі зображення (з розмірами ширини на висоту) і кількості кутів (numangles) або $\left[\sqrt{a^2 + b^2}\right] * n$, де a – це ширина, b – довжина, n – кількість кутів. Для отримання хороших результатів необхідно масштабування зображення. Фактично, $\min(arctg\left(\frac{2}{a}\right), arctg(\frac{2}{h}))$ (де a – це ширина, h – висота), буде мінімальним кутом для пошуку всіх трьох точок, так як це найменше прирощення між вертикальною і горизонтальною лінією. Задавши алгоритму число, наприклад 360 * 5 можливих кутів між 90 і -90 градусами, стає можливим використати формулу $[\frac{2}{tg(\frac{\pi}{360*5})}]$ для знаходження максимального розміру зображення і просто масштабувати на цю величину, якщо зображення виходить за межі. Початком буде адаптації даних часового ряду в зображення. Для цього необхідно зробити значення ціни дискретним, що можна зробити шляхом множення на сто, щоб прибрати десяткову частину доларової суми. Розмір зображення повинен визначатися тільки тривалістю часу та мінімальної і максимальною ціною за цей час.

Для всіх кутів і всіх точок обчислюється відстань до прямої, перпендикулярної кутку і проходить через точку. Для кожного кута відстань до цієї перпендикулярної прямої накопичується точкою, де різні точки з однаковими відстанями повинні лежати на одній прямій.

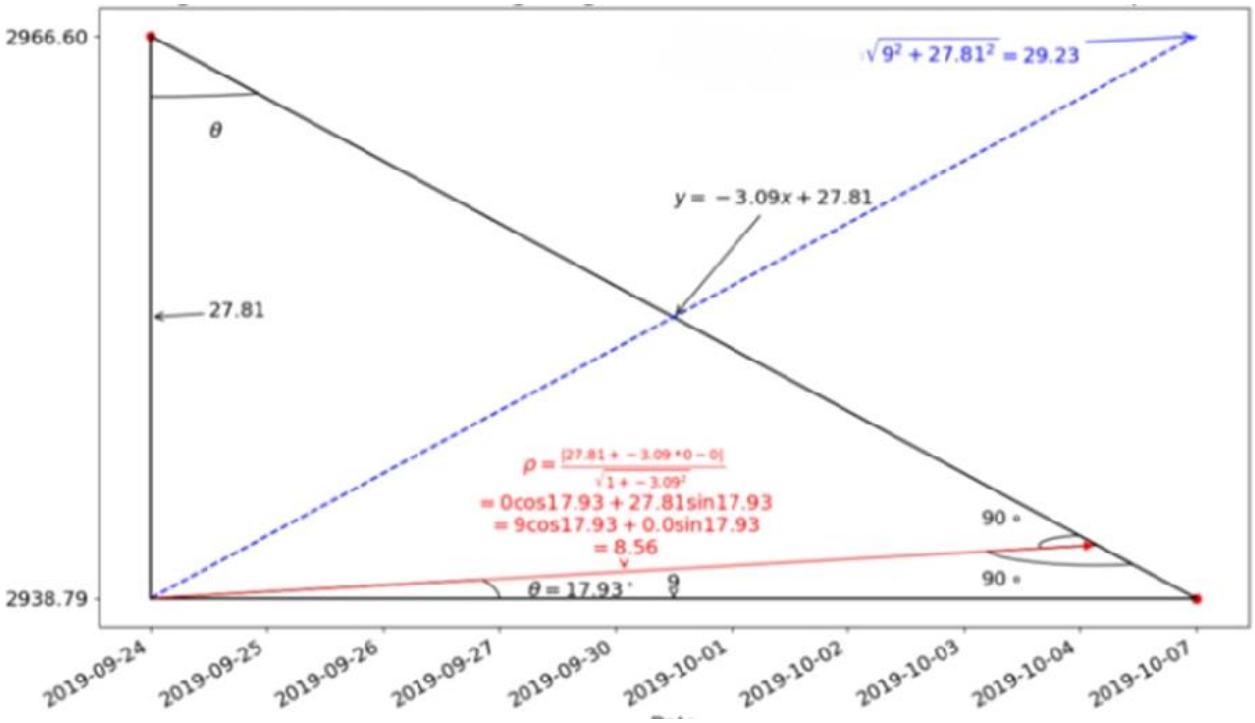

Рис.3 Побудова ліній Хафа

Нахил перпендикулярних ліній: $m_p = -\frac{1}{m}, mm_p = -1$

Перпендикулярна лінія що проходить через точку: $y = \frac{x_0 - x}{m} + y_0$

Точка (x,y) перетину ліній: $mx + b = \frac{x_0 - x}{m} + y_0 = \frac{x_0 + my_0 - mb}{m^2 + 1}$

Відстань від точки до лінії після спрощення :$d = \frac{|b + mx_0 - y_0|}{\sqrt{1+m^2}}, p = xcos\Theta + ysin\Theta$

Це призводить до алгоритму O (n * m), де m - число кутів, а для використання пам'яті потрібно в m разів більше довжини діагоналі двовимірного простору точок.

Перетворення Хафа для економії пам'яті не повертає ніякої інформації по конкретній точці, що для цілей візуалізації може бути дуже корисно. Тому розраховується відстань для всіх точок до прямої, впорядковується і береться якомога більше точок, які укладаються в

допустиму похибку. Відстань точки до лінії використовується для визначення правильного акумулятора для інкремента. Нахил перпендикуляра є негативним зворотною величиною нахилу, тому їх множення дорівнює мінус один. Будується перпендикулярна лінія до точки щодо нової точки, і обчислюється точка перетину.Інша частина виводиться за допомогою відстані між точкою і точкою перетину. На малюнку, що показує перетворення Хафа, також приведена явна формула.

2.2.4 Метод імовірнісного лінійного перетворення Хафа

Точні використовувані кути і високі вимоги до обробки і пам'яті, необхідні для звичайної ідентифікації лінії Хафа, роблять її трохи непрактичною для цього завдання з невеликою кількістю точок. Чим більше число точок, тим краще він може працювати. Однак цей попередній метод був швидше через те що втрачає в точності. На практиці використовується розподіл усіх перетворення лінії Хафа, при якому пошук здійснюється випадковим чином, а для фільтрації результатів використовуються параметри, включаючи поріг для кількості точок.

Тут як і раніше повертаються тільки дві точки лінії. На жаль, результати будуть відрізнятися при кожному запуску, оскільки імовірнісний алгоритм має випадковий характер. Це в цілому робить даний метод не особливо підходящим для пошуку всіх ліній тренда. Оскільки його можна запускати кілька разів, це можна робити до тих пір, поки не відбудеться певна кількість запусків, чи не буде знайдено певну кількість ліній, щоб значно збільшити ймовірність знаходження всіх ліній.

2.3 Пошук значимих ліній тренду

Як було написано раніше, деякі лінії тренду можуть давати корисну інформацію про рух ціни у майбутньому, в той час як деякі, побудовані по трьом точкам зміни напряму ціни, більше не знадобиться, тому метою роботи є знаходження саме значимих ліній тренду.

Для цього треба проаналізувати зв'язок між конкретною лінією тренду та об’ємом. Значимою лінією тренду є та, яка, при дотику графіку ціни до неї, супроводжується приростом об’єму.

На практиці це буде перевірятись так:

Якщо об’єм в момент часу, при якому графік ціни відбився від лінії тренду, збільшується в nразів,де $n > 1$, за середнє значення об’єму за останні t одиниць часу, то лінія тренду є значимою.

РОЗДІЛ 3

РОЗРОБКА ПРОГРАМНОГО КОДУ

3.1 Отримання даних

Для цього завдання було використано Python 3 оскільки він має потрібні бібліотеки, розроблені для роботи з наборами даних. Спочатку потрібно отримати дані про запаси. Для цієї серії прикладів буде використовуватися переважаючий індекс S&P500 (Ticker: ^GSPC)щоб мати практичну актуальність.

Бібліотека Python yfinance може дуже легко отримати усі необхідні дані. Наприклад, для зручності брати трохи менше 4 років даних або останні 1000 значень. Він поверне дані у вигляді pandas DataFrame для зручного індексування та інших операцій(Додаток А, рис. 1).

Для відтворюваних результатів набір даних було взято на момент закриття 7 жовтня 2019 року та показано фільтрування коментарів з цією метою.

3.2 Пошук мінімумів та максимумів

Локальні екстремуми в програмі знаходились за допомогою методу чисельного диференціювання(Див. Додаток А, рис. 2).

За потреби його можна замінити на метод звичайного перебору(Див. додаток А, рис. 3)

3.3 Лінійна регресія

За рахунок швидкості, ми будемо реалізовувати це для будь-якої кількості точок, оскільки кращої лінією тренда буде та, яка може мати навіть більше трьох точок, враховуючи, що три - це просто мінімум(Див. Додаток А, рис. 4)

Будемо використовувати функції бібліотеки numpy, а саме polyfit таpoly1dдля розрахунку середньої лінії підтримки і середньої лінії опору на основі всіх локальних точок мінімуму і максимуму відповідно. Нахил і перехоплення повертають polyfit і poly1d, хоча poly1d забезпечує більш легке управління і використання, і тому демонструється, де повертається помилка є просто сумою квадратів залишкової помилки, але для цілей порівняння є послідовною.Хоча така загальна функція підходить для одиничного використання, наприклад, для загальної середньої, вона може бути не такою швидкою, як оптимізовані функції, написані вище, тому необхідно пройти через їх написання. Звичайно, для підгонки цієї лінії необхідно було визначити як мінімум 2 точки.

3.4 Метод лінійного перетворення Хафа

Можемо використовувати функцію перетворення лінії Хафа бібліотеки scikit-image під назвою hough_line. Корисними деталями є те, що він змінює осі місцями і має пороговий параметр, який більше, але не дорівнює.Існують невеликі детальні відмінності у внутрішньому функціонуванні, особливо щодо вибору локальних максимумів , тому результати не будуть повністю ідентичними. Це призводить до двох різних методів виконання розрахунку (один оптимізований для точок, інший перетворює точки в зображення для бібліотеки,Див. Додаток А, рис. 9)

3.5 Метод імовірнісного лінійного перетворення Хафа

. Для цього методу, на відміну від звичайного методу Хафа, працює інша бібліотечна функція probabilistic_hough_line.. Параметр hough_prob_iter задає кількість ітерацій для запуску, наприклад, 10 ітерацій для збільшення ймовірності отримання достатньої кількості ліній. (Див. Додаток А. рис. 10).

3.6 Пошук значимих прямих тренду

У даній програмі значення nбуло взято за 2 та t за останні 3 хвилини(за умови що розглядається ділянка в 1 годину (Див. Додаток А рис. 11).

ВИСНОВКИ

Було проаналізовано і продемонстровано, як використовувати різні методи, включаючи чисельне диференціювання, для знаходження поворотних точок.

Крім того, було показано, як використовувати різні методи ідентифікації лінії тренду, включаючи використання відсортованого нахилу, перетворення лінії Хафа і імовірнісного перетворення лінії Хафа. Також проаналізовано метод пошуку значимих ліній тренду за допомогою спостережень за об’ємом.

Було розроблено застосунок для побудови ліній тренду та знаходження серед них значимих. Розроблений застосунок можна використати для написання торгового бота для гри на біржі, щоб робити аналіз стану ціни цінних паперів застосовуючи більш складні методи технічного аналізу. Також додаток буде корисний для машинного навчання нейронних мереж, даючи їм як один з видів даних уже готові значимі лінії тренду.

Навіть з урахуванням роботи з історичними данними ці показання неймовірно корисні, оскільки знаходиться значимі рівні, до яких ціна в майбутньому скоріш за все повернеться.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Біржа [Електронний ресурс].- Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Біржа.
2. Об’єм [Електронний ресурс].-Режим доступу до ресурсу: https://atas.net/volume-analysis/basics-of-volume-analysis/how-to-use-a-horizontal-volume/
3. Лінії тренду[Електронний ресурс].- Режим доступу о ресурсу: https://profitov.net/bulish-and-bearish-trend/
4. Kaabar S. The handbook of exotic trading strategies: Uncommon techniques to diversify your prediction methods / Sofien Kaabar., 2019.
5. Kaabar S. New Technical Indicators in Python / Sofien Kaabar., 2021.
6. Coulling A. A Complete Guide To Volume Price Analysis / Anna Coulling., 2013.
7. Dormeier B. Investing with Volume Analysis: Identify, Follow, and Profit from Trends (paperback) 1st Edition / Buff Dormeier., 2013.

ДОДАТОК А

```
import numpy as np
import pandas as pd
import yfinance as yf #pip install yfinance
tick = yf.Ticker('^GSPC')
hist = tick.history(period="max", rounding=True)
#hist = hist[:'2019-10-07']
hist = hist[-1000:]
h = hist.Close.tolist()
```

Рис. 1. Програмний код отримання данних

```
from findiff import FinDiff #pip install findiff
dx = 1 #1 day interval
d_dx = FinDiff(0, dx, 1)
d2_dx2 = FinDiff(0, dx, 2)
clarr = np.asarray(hist.Close)
mom = d_dx(clarr)
momacc = d2_dx2(clarr)
```

```
def get_extrema(isMin):
  return [x for x in range(len(mom))
    if (momacc[x] > 0 if isMin else momacc[x] < 0) and
      (mom[x] == 0 or #slope is 0
        (x != len(mom) - 1 and #check next day
          (mom[x] > 0 and mom[x+1] < 0 and
           h[x] >= h[x+1] or
           mom[x] < 0 and mom[x+1] > 0 and
           h[x] <= h[x+1]) or
         x != 0 and #check prior day
          (mom[x-1] > 0 and mom[x] < 0 and
           h[x-1] < h[x] or
           mom[x-1] < 0 and mom[x] > 0 and
           h[x-1] > h[x])))]
minimaIdxs, maximaIdxs = get_extrema(True), get_extrema(False)
```

Рис. 2. Пошук екстремумів метоом чисельного диференціювання

```
minimaIdxs = np.flatnonzero(
 hist.Close.rolling(window=3, min_periods=1,
center=True).aggregate(
   lambda x: len(x) == 3 and x[0] > x[1] and x[2] >
x[1])).tolist()
maximaIdxs = np.flatnonzero(
 hist.Close.rolling(window=3, min_periods=1,
center=True).aggregate(
   lambda x: len(x) == 3 and x[0] < x[1] and x[2] <
x[1])).tolist()
```

```
hs = hist.Close.loc[hist.Close.shift(-1) != hist.Close]
x = hs.rolling(window=3, center=True)
     .aggregate(lambda x: x[0] > x[1] and x[2] > x[1])
minimaIdxs = [hist.index.get_loc(y) for y in x[x == 1].index]
x = hs.rolling(window=3, center=True)
     .aggregate(lambda x: x[0] < x[1] and x[2] < x[1])
maximaIdxs = [hist.index.get_loc(y) for y in x[x == 1].index]
```

Рис. 3. Програмний код пошуку екстремумів

```
def get_bestfit3(x0, y0, x1, y1, x2, y2):
  xbar, ybar = (x0 + x1 + x2) / 3, (y0 + y1 + y2) / 3
  xb0, yb0, xb1, yb1, xb2, yb2 =
    x0-xbar, y0-ybar, x1-xbar, y1-ybar, x2-xbar, y2-ybar
  xs = xb0*xb0+xb1*xb1+xb2*xb2
  m = (xb0*yb0+xb1*yb1+xb2*yb2) / xs
  b = ybar - m * xbar
  ys0, ys1, ys2 =
    (y0 - (m * x0 + b)),(y1 - (m * x1 + b)),(y2 - (m * x2 + b))
  ys = ys0*ys0+ys1*ys1+ys2*ys2
  ser = np.sqrt(ys / xs)
  return m, b, ys, ser, ser * np.sqrt((x0*x0+x1*x1+x2*x2)/3)
```

```
def get_bestfit(pts):
  xbar, ybar = [sum(x) / len (x) for x in zip(*pts)]
  def subcalc(x, y):
    tx, ty = x - xbar, y - ybar
    return tx * ty, tx * tx, x * x
  (xy, xs, xx) =
    [sum(q) for q in zip(*[subcalc(x, y) for x, y in pts])]
  m = xy / xs
  b = ybar - m * xbar
  ys = sum([np.square(y - (m * x + b)) for x, y in pts])
  ser = np.sqrt(ys / ((len(pts) - 2) * xs))
  return m, b, ys, ser, ser * np.sqrt(xx / len(pts))
```

Рис. 4. Лінійна регреся

```
ymin, ymax = [h[x] for x in minimaIdxs], [h[x] for x in maximaIdxs]


zmin, zmne, _, _, _ = np.polyfit(minimaIdxs, ymin, 1, full=True)
#y=zmin[0]*x+zmin[1]
pmin = np.poly1d(zmin).c
zmax, zmxe, _, _, _ = np.polyfit(maximaIdxs, ymax, 1, full=True)
#y=zmax[0]*x+zmax[1]
pmax = np.poly1d(zmax).c
print((pmin, pmax, zmne, zmxe))
```

Рис. 5.Лінійна регресія, продовження

```
def get_trend(Idxs):
  trend = []
  for x in range(len(Idxs)):
    for y in range(x+1, len(Idxs)):
      for z in range(y+1, len(Idxs)):
        trend.append(([Idxs[x], Idxs[y], Idxs[z]],
          get_bestfit3(Idxs[x], h[Idxs[x]],
                       Idxs[y], h[Idxs[y]],
                       Idxs[z], h[Idxs[z]])))
  return list(filter(lambda val: val[1][3] <= fltpct, trend))
mintrend, maxtrend = get_trend(minimaIdxs), get_trend(maximaIdxs)
```

Рис. 6. Нативний метод

```
def get_trend_opt(Idxs):
  slopes, trend = [], []
  for x in range(len(Idxs)): #O(n^2*log n) algorithm
    slopes.append([])
    for y in range(x+1, len(Idxs)):
      slope = (h[Idxs[x]] - h[Idxs[y]]) / (Idxs[x] - Idxs[y])
      slopes[x].append((slope, y))
  for x in range(len(Idxs)):
    slopes[x].sort(key=lambda val: val[0])
    CurIdxs = [Idxs[x]]
    for y in range(0, len(slopes[x])):
      CurIdxs.append(Idxs[slopes[x][y][1]])
      if len(CurIdxs) < 3: continue
      res = get_bestfit([(p, h[p]) for p in CurIdxs])
      if res[3] <= fltpct:
        CurIdxs.sort()
        if len(CurIdxs) == 3:
          trend.append((CurIdxs, res))
          CurIdxs = list(CurIdxs)
        else: CurIdxs, trend[-1] = list(CurIdxs), (CurIdxs, res)
      else: CurIdxs = [CurIdxs[0], CurIdxs[-1]] #restart search
  return trend
mintrend, maxtrend =
  get_trend_opt(minimaIdxs), get_trend_opt(maximaIdxs)
```

Рис. 7. Метод відсортованого схилу

```
def make_image(Idxs):
  max_size = int(np.ceil(2/np.tan(np.pi / (360 * 5)))) #~1146
  m, tested_angles =
    hist.Close.min(), np.linspace(-np.pi / 2, np.pi / 2, 360*5)
  height = int((hist.Close.max() - m + 0.01) * 100)
  mx = min(max_size, height)
  scl = 100.0 * mx / height
  image = np.zeros((mx, len(hist))) #in rows, columns or y, x
  for x in Idxs:
    image[int((h[x] - m) * scl), x] = 255
  return image, tested_angles, scl, m
```

```
def hough_points(pts, width, height, thetas):
  diag_len = int(np.ceil(np.sqrt(width * width + height * height)))
  rhos = np.linspace(-diag_len, diag_len, diag_len * 2.0)
  # Cache some resuable values
  cos_t = np.cos(thetas)
  sin_t = np.sin(thetas)
  num_thetas = len(thetas)
  # Hough accumulator array of theta vs rho
  accumulator =np.zeros((2 * diag_len, num_thetas), dtype=np.uint64)
  # Vote in the hough accumulator
  for i in range(len(pts)):
    x, y = pts[i]
    for t_idx in range(num_thetas):
      # Calculate rho. diag_len is added for a positive index
      rho=int(round(x * cos_t[t_idx] + y * sin_t[t_idx])) + diag_len
      accumulator[rho, t_idx] += 1
  return accumulator, thetas, rhos
```

```
def find_line_pts(Idxs, x0, y0, x1, y1):
  s = (y0 - y1) / (x0 - x1)
  i, dnm = y0 - s * x0, np.sqrt(1 + s*s)
  dist = [(np.abs(i+s*x-h[x])/dnm, x) for x in Idxs]
  dist.sort(key=lambda val: val[0])
  pts, res = [], None
  for x in range(len(dist)):
    pts.append((dist[x][1], h[dist[x][1]]))
    if len(pts) < 3: continue
    r = get_bestfit(pts)
    if r[3] > fltpct:
      pts = pts[:-1]
      break
    res = r
  pts = [x for x, _ in pts]
  pts.sort()
  return pts, res
```

Рис. 8. Метод Лінійного перетворення Хафа

```
def houghpt(Idxs):
  max_size = int(np.ceil(2/np.tan(np.pi / (360 * 5)))) #~1146
  m, tested_angles =
    hist.Close.min(), np.linspace(-np.pi / 2, np.pi / 2, 360*5)
  height = int((hist.Close.max() - m + 1) * 100)
  mx = min(max_size, height)
  scl = 100.0 * mx / height
  acc, theta, d = hough_points(
    [(x, int((h[x] - m) * scl)) for x in Idxs], mx, len(hist),
    np.linspace(-np.pi / 2, np.pi / 2, 360*5))
  origin, lines = np.array((0, len(hist))), []
  for x, y in np.argwhere(acc >= 3):
    dist, angle = d[x], theta[y]
    y0, y1 = (dist - origin * np.cos(angle)) / np.sin(angle)
    y0, y1 = y0 / scl + m, y1 / scl + m
    pts, res = find_line_pts(Idxs, 0, y0, len(hist), y1)
    if len(pts) >= 3: lines.append((pts, res))
  return lines
mintrend, maxtrend = houghpt(minimaIdxs), houghpt(maximaIdxs)


def hough(Idxs): #pip install scikit-image
  image, tested_angles, scl, m = make_image(Idxs)
  from skimage.transform import hough_line, hough_line_peaks
  h, theta, d = hough_line(image, theta=tested_angles)
  origin, lines = np.array((0, image.shape[1])), []
  for pts, angle, dist in
                  zip(*hough_line_peaks(h, theta, d, threshold=2)):
    y0, y1 = (dist - origin * np.cos(angle)) / np.sin(angle)
    y0, y1 = y0 / scl + m, y1 / scl + m
    pts, res = find_line_pts(Idxs, 0, y0, image.shape[1], y1)
    if len(pts) >= 3: lines.append((pts, res))
  return lines
mintrend, maxtrend = hough(minimaIdxs), hough(maximaIdxs)
```

Рис. 9.Два методи для методу Хафа

```
def prob_hough(Idxs): #pip install scikit-image
  image, tested_angles, scl, m = make_image(Idxs)
  from skimage.transform import probabilistic_hough_line
  lines = []
  for x in range(hough_prob_iter):
    lines.append(probabilistic_hough_line(image, threshold=2,
                theta=tested_angles, line_length=0,
      line_gap=int(np.ceil(np.sqrt(
        np.square(image.shape[0]) + np.square(image.shape[1]))))))
  l = []
  for (x0, y0), (x1, y1) in lines:
    if x0 == x1: continue
    if x1 < x0: (x0, y0), (x1, y1) = (x1, y1), (x0, y0)
    y0, y1 = y0 / scl + m, y1 / scl + m
    pts, res = find_line_pts(Idxs, x0, y0, x1, y1)
    if len(pts) >= 3: l.append((pts, res))
  return l
mintrend, maxtrend = prob_hough(minimaIdxs), prob_hough(maximaIdxs)
```

Рис. 10. Імовірнісний метод Хафа

```
def find_good_lines(l, v, x): #l-lines, v-volume, x-stock diagram
  good_lines = list()
  good_coords = list()
  for line in l:
    lines_coords = np.argwhere(np.diff(np.sign(line -
cost))).flatten()
    for i in range(len(v)):
      if (v[i-3]+v[i-2]+v[i-1])/3 <= 2*v[i]:
        good_coords.append(x[i])
    if set(lines_coords) & set(good_coords) != set():
      good_lines.append(line)
  return good_lines
```

Рис. 11.Пошук значимих ліній