Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра математики факультету інформатики

Принцип роботи програм для прогнозування цін активів на фінансовому ринку

Курсова робота
за спеціальністю „Прикладна математика”

Керівник курсової роботи
доц. Щестюк Н.Ю.

_____
(підпис)
“19” квітня 2020 р.
Виконав студент Чорноус
Станіслава Володимирівна

“19” квітня 2020 р.

Київ 2020

Тема: <u>Принцип роботи програм для прогнозування цін активів на фінансовому ринку</u>

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	02.11.2019	
2.	Огляд технічної літератури за темою роботи.	15.11.2019	
3.	Опрацювання матеріалів	17.11.2019	
4.	Проведення аналізу	20.01.2020	
5.	Написання роботи	06.03.2020	
6.	Перевірка та погодження роботи	06.04.2020	
7.	Створення презентації та написання доповіді	10.04.2020	
8.	Остаточне оформлення роботи	18.04.2020	
9.	Захист курсової роботи	19.04.2020	

Студент <u>Чорноус Станіслава Володимирівна</u>

Керівник <u>Щестюк Наталія Юріївна</u>

“<u>19</u>” <u>квітня</u>

Вступ

Перший фондовий ринок виник в Антверпені в 1531 році. На будівлі щойно відкритої біржі було написано: “Для використання торговими людьми всіх народів і мов”. Тож 1531 рік можна вважати виникненням першої міжнародної фондової біржі.

Зараз фондовий ринок - це величезна мережа відносин між країнами, компаніями, корпораціями та окремими людьми. І цей зв’язок ніколи не буває стабільним. Тож для більшості людей, що намагаються заробляти за допомогою фінансових ринків, постає потреба в прогнозуванні. Для недосвідченого новачка зрозуміти зміну і напрямок зміни ціни того чи іншого активу - справа занадто важка. Тож на цьому фоні виникають сотні компаній шахраїв, що пропонують чудо-програми для прогнозу і заробітку на фінансових ринках. Одна кнопка, що вирішить всі ваші проблеми - шахрайство чи неймовірна можливість?

Прогнозування - одна із найнеобхідніших і найскладніших задач аналізу. Проблеми при її вирішені обумовлені багатьма причинами - недостатня якість і кількість вихідних даних, мінливість середовища, вплив суб’єктивних факторів, таких як політичні рішення та рішення окремих осіб, що не можна передбачити. Але саме якісний прогноз є ключем до вирішення подібних задач.

Метою моєї роботи є дослідження способів автоматизації прогнозування біржової інформації на основі нейромережевого моделювання засобами інтелектуального аналізу даних та доведення того, що прогнозування для вдалого заробітку на фінансових ринках - задача, котру можна вирішити.

1. Дослідження фінансових ринків

1.1. Сутність фінансового ринку

Фінансовий ринок — це сукупність фінансових відносин, які пов’язані з процесами купівлі-продажу фінансових ресурсів, які є необхідними для проведення виробничої та фінансової діяльності. Відносини купівлі-продажу полягають в передачі одним суб'єктом іншому за відповідну плату права на тимчасове чи постійне використання фінансових ресурсів. Така передача може відбуватися прямо або з допомогою фінансових посередників. У випадку безпосередніх взаємовідносин купівлі-продажу ресурсів, вони визначаються як відносинами обміну, так і відносинами перерозподілу таких ресурсів між власниками і користувачами.

Фінансовий ринок є важливою ланкою в умовах ринкової економіки. Його призначення полягає в забезпеченні необхідних умов для залучення коштів або продажу тимчасово вільних ресурсів. Функціями фінансового ринку є мобілізація тимчасово вільних ресурсів, розподіл коштів між споживачами, мінімізація фінансових ринків. На ринку існують 4 основних принципи:

1) вільний доступ до інформації та інструментів для всіх учасників;
2) прозорість та правовий захист всіх його учасників;
3) конкурентність та ефективність;
4) відповідність міжнародним стандартам.

За видами активів фінансовий ринок поділяють на:

1) валютний ринок (FOREX);

2) товарно-сировинний ринок;
3) фондовий ринок (ринок цінних паперів);
4) індекси;
5) ринок криптовалют;

Висновки до розділу

Фінансовий ринок виконує надзвичайно важливі функції. Він забезпечує своїх суб'єктів доступом до фінансової системи, належними умовами для залучення необхідних коштів і продажу тимчасово вільних ресурсів. Для того, щоб зрозуміти, що відбувається на ньому, його необхідно досліджувати.

Головна проблема в роботі з фінансовими ринками це складність в прогнозуванні руху обраних активів та власне вибір правильних активів для отримання прибутку. На цьому фоні виникає безліч програм, в тому числі і шахрайських, що компенсують некомпетентність та нестачу інформації щодо ринку.

2. Обмеження об’єкту автоматизації та форматування вимог до системи

2.1 Опис задачі, що потребує автоматизації

Прогнозування фінансових часових рядів - необхідний елемент будь-якої інвестиційної діяльності. Інвестиції - це вкладання фінансів із метою отримання прибутку в майбутньому. Для вдалих інвестицій ми потребуємо прогнозування майбутнього. Саме тому, прогнозування фінансових часових рядів лежить в основі діяльності всіх сфер, що потребують інвестицій - бірж і небіржових систем торгівлі активами.

Щоденний обіг ринку активів США перевищує 12 млрд. доларів. Депозиторій Depositary Trust Company, де зареєстровано цінних паперів на суму 11 трлн. доларів, щоденно реєструє в день операцій, приблизно на 250 млрд. доларів. Проте найактивніше йде торгівля на світовому валютному ринку FOREX. Його добовий обіг перевищує 1000 млрд. доларів. Це, приблизно 1/50 всього сукупного капіталу землі.

Відомо, що 99% всіх укладених угод - спекулятивні, тобто вони засновані на прогнозах зміни курсу їх учасниками. При цьому передбачення учасників кожної угоди протилежні один до одного. Так що обсяг спекулятивних операцій характеризує рівень непередбачуваності фінансових ринків. Тож на перший план виходить рішення задачі вибору ефективних методів прогнозування фінансових часових рядів. Той, хто володіє інформацією - той, володіє світом. В нашому ж випадку той, хто має кращі математичні методи для прогнозування, той володіє кращим прибутком. Адже за їх допомогою можна вилучити закономірності із

зашумлених хаотичних рядів і заробляти за рахунок своїх менш оснащених та обізнаних опонентів.

В останнє десятиліття більшої популярності набуває технічний аналіз ринку. Це набір емпіричних правил, що грунтуються на різноманітних індикаторах поведінки ринку. Тобто він не враховує впливу різного роду суб'єктів та випадкових подій, таких як епідемії, вибухи вулканів, заборони на ввезення чогось в якісь країни. Технічний аналіз зосереджується на індивідуальній поведінці даного фінансового інструмента, поза його зв'язками з іншими цінними паперами.

Згідно з основним правилом технічного аналізу, передбачення повинні бути засновані на алгоритмі, тобто їх можна доручити комп'ютеру. За людиною залишається лише розробка алгоритму, що коректно буде враховувати всі аспекти змін руху активів на фінансових ринках.

Нейромережний аналіз, на відміну від технічного, не припускає ніяких обмежень на характер вхідної інформації. Це можуть бути як індикатори даного часового ряду, так і відомості про поведінку інших ринкових інструментів. Зокрема, нейронні мережі використовують великі пенсійні фонди, що працюють з великими інвестиційними портфелями, для яких особливо важливі кореляції між різними ринками.

На відміну від технічного аналізу, заснованого на загальних рекомендаціях, нейромережі здатні знаходити оптимальні індикатори і будувати за ними оптимальну стратегію передбачення. Більш того, ці стратегії адаптуються під зміни волатильного ринку.

Нейромережеве моделювання має і деякий недолік. Воно базується лише на даних, а їх може не вистачити для навчання програми, або ж розмірність потенційних входів може виявитися занадто високою.

Все це свідчить, що нейромережевий трейдинг є найбільш перспективним на коротких часових проміжках. До того ж через подібність фінансових часових рядів, прибуток за одиницю часу буде тим вище, чим менше час трейдингу. Тож автоматичні програми для трейдингу будуть корисні при торгівлі в реальному часі. Саме в такому випадку видно переваги програми над людиною-трейдером. Програма не втомлюється, не має фізичних потреб, та здатна в реальному часі реагувати на найменші зміни напрямку ціни на актив. Навчена нейронна мережа, під’єднана до електронної системи торгівлі, може приймати рішення про продаж чи придбання активу ще до того, як людина-брокер розпізнає зміни в графіку котирувань на своєму торговому терміналі.

2.2 Аналіз програм для автоматизації торгівлі

На даний момент на ринку присутній ряд готових програм, розроблених за для прогнозування зміни цін активів та автоматизації торгівлі на фінансових ринках. Серед них є певна градація, що виходить з таких параметрів, як розмір компанії розробника ьа функціональність продукту.

За результатами дослідження компанії Reuters, аналітичні пакети можна розподілити на наступні категорії:

- Пакети для новачків, що містять мінімум функціоналу, налаштування якого залишаються всередині системи і користувач не має доступу до них. Цей клас програм є скоріше ознайомчим, та його

пропонують здебільшого компанії-шахраї. Вони на є придатними для справжньої роботи.

- Легкі пакети мають обмежене число методів, але користувач має доступ до більшості налаштувань. Не менш істотним є такий параметр як візуалізація, адже вона з’являється саме в цьому класі програм.
- Середні системи відрізняються від попередніх тим, що мають більший набір методів для прогнозування.
- Важкі системи - це системи дослідницького типу, бо для їх успішного застосування потрібні специфічні знання не тільки прикладної області, а й математичних методів, що присутні в програмі. Дані пакети складніші у використанні, оскільки націлені на професіоналів в цій сфері, тобто вимагає спеціальних знань від користувача. Дані програми найчастіше використовуються в наукових дослідженнях та з метою навчання.

Наведена діаграма (табл.1) демонструє, що важкі системи практично не використовуються в реальних завданнях. Це пояснюється високими вимогами до обчислювальних потужностей. Технічно процес заключається в тому, що на важких системах розробляється той або інший підхід, який надалі набуває комерційного образу у вигляді більш легких пакетів.

Домінування легких пакетів пояснюється тим, що користувачі не є фахівцями в галузі математичного налаштування методів та ці пакети є проміжною частиною рішення, і від них вимагають максимальної продуктивності.

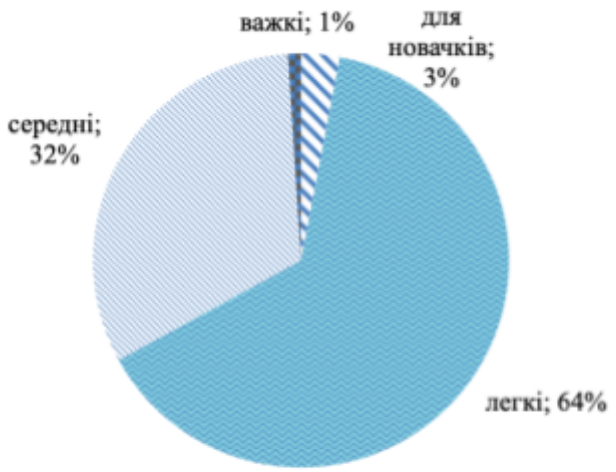


табл. 1. Порівняння обсягу користування категоріями пакетів

2.3 Вимоги до системи

Головною метою розробки аналітичної платформи є автоматизація прогнозування біржових котирувань на основі алгоритмів кластеризації і нейронних мереж.

Вибір даних для навчання мережі та їх обробка є найскладнішим етапом. Набір даних має відповідати наступним критеріям:

- репрезентативність, дані повинні ілюструвати справжні ціни активів;
- несуперечність, бо суперечні дані призведуть до поганої якості навчання мережі;
- корелювання, бо чим вище кореляція між вхідними і вихідними даними нейронної мережі, тим якіснішими будуть прогнози.

Після завантаження вхідних даних потрібно мати змогу отримати деяку статистику, що дасть первісне представлення про завантажені дані, наприклад у вигляді діаграм.

Після завершення підготовки даних, програма має самостійно створювати таблиці, що відповідають основному логічному переходу на етапі підготовки даних.

Програма повинна мати наступні переваги і властивості:

- Простота у використанні, бо саме простота роботи зі складними методами аналізу є основною перевагою над більш громіздкими програма професійного рівня;
- Швидкість отримання результатів, рішення знаходяться ітеративно і висока швидкість отримання результатів потрібна користувачу-дилетанту;
- Зрозумілість процесу, адже розуміння технологічного процесу є перевагою над звичайними програмами, що надають обмежену інформацію про процес прогнозування.

Висновки до розділу

Сукупність програм для автоматизованого трейдингу поділяється на 4 сегменти, що підлаштовані під компетентність та обізнаність користувачів. Зі збільшенням точності прогнозу та кількості налаштувань - збильшуються і вимоги до обізнаності потенційного трейдера. Ідеальна у використанні програма має бути створена на основі нейронних мереж, бо матиме переваги у швидкості і точності результатів, та матиме інтуїтивно зрозумілий принцип роботи для людини без специфічних знань в математичному моделюванні та прогнозуванні.

3. Побудова оптимальних параметрів моделей ціноутворення на фінансовому ринку.

3.1 Математична модель для прогнозування ціни активу

Математична модель формування ринкової вартості акції та портфеля акцій у загальному вигляді можуть бути записані наступним чином:

$$\widehat{r_i} = \frac{dr_i}{dt} = f(r_i, t, \alpha),\ r_i(t_0) = r_0,\ t \in [t_0, T],\ i = \overline{1, n},\quad (1)$$

і

$$\widehat{r_p} = f^p(r_p, x_i, \widehat{x_i}, r_i, \widehat{r_i}, t)\quad (2)$$

Тут r_i - очікувана ринкова вартість активу; r_p - очікувана ринкова вартість інвестиційного портфеля; x_i - частка активу у портфелі; t - час.

Важливою математичною задачею, розв'язок якої використовується для коректного формулювання задачі про побудову оптимальної структури інвестиційного портфеля, є задача ідентифікації оптимальних значень параметрів $\alpha_1, \alpha_2, \rho_{ij}$ математичних моделей формування ринкової вартості однієї акції та повного портфеля.

Задача про оптимізацію портфеля акцій з точки зору його прибутковості у статичному випадку має вигляд

$$r_p = \sum_{i=1}^{n} x_i r_i \to \max_x.$$

У ході практичного інвестування вона розглядається у такій постановці

$$r_p(T) = \sum_{i=1}^{n} x_i(T) r_i(T) \to \max_{x \in X(t)}, \quad r_p(t_0) = r_{p_0}, \ t \in [t_0, T]$$

і полягає у побудові оптимального за прибутковістю у кінцевий момент часу інвестиційного портфеля, де $X(t)$ - обмежена замкнута множина акцій. Праві частини рівнянь системи (1) є лінійними і залежать від параметрів. Для забезпечення можливості її інтегрування

$$\frac{dr_i}{dt} = (\alpha_1 SM_{ind}(t) + \alpha_2 I(t)) r_i(t) + \alpha_3 r_j(t)), \quad i, j = \overline{1,n}$$

побудуємо процедуру їх пошуку.

За різних постановок задачі за вектор параметрів можна розглядати вектори $\alpha = (\alpha_1, \alpha_2, \rho_{ij})$ або $\alpha = (\alpha_1, \alpha_2)$.

Перше формулювання вектора параметрів будемо застосовувати у випадку, коли інформація щодо ринко- вої вартісті акцій є «зашумленою». У випадку, коли ринкова вартість акцій адекватно відображає динаміку ринку, застосовують друге формулювання.

Розглянемо рівняння формування динаміки ринкової вартості однієї акції у вигляді

$$\frac{dr_i(t)}{dt} = (\alpha_1 SM_{ind}(t) + \alpha_2 I(t)) r_i(t) + \sum_{j=1}^{n} \rho_{ij} r_i(t) + \xi_i(t), \quad i = \overline{1,n}.$$

Тут α_1, α_2 - параметри моделі; ρ_{ij} - враховує кореляційні залежності між акціями; ξ_i - вектор збурень.

У випадку, якщо α_1, α_2 =const для обраного і можемо отримати

$$(\alpha_1 SM_{ind}(t) + \alpha_2 I(t) + \sum_{j=1}^{n} \rho_{ij}) r(t) = \xi(t) - \frac{dr}{dt}$$

Якщо вектори $\widehat{r}$, r, ξ є доступними для визначення, то це рівняння можна розглядати як прямий алгоритм пошуку параметрів математичної моделі. Похибка ідентифікації лінійно залежить від похибок визначень $\widehat{r}$, ξ . Аналіз вказує на те, що вплив похибок визначень r на оцінювання параметрів збільшується, якщо рівняння системи погано обумовлені, тобто коли матриця R близька до особливої, а визначник близький до 0. Для того, щоб рівняння $\widehat{R} - \alpha R = \xi$,

де $R = (r_1, r_2, .., r_n)$, $\widehat{R} = (\widehat{r_1}, \widehat{r_2}, .., \widehat{r_n})$, було добре обумовлено, то необхідно, щоб $r_i(t)$ були незалежними. Якщо ж процес є стохастичним, то ця умова зводиться до некорельованості значень $r_i(t)$, $i = \overline{1,k}$.

Далі процедура ідентифікації параметрів моделі зводиться до отримання розв'язку останнього рівняння для різних моментів часу. Розв'язки рівняння для k+2 моментів часу дають можливість отримати систему із k+2 алгебраїчних рівнянь відносно невідомих параметрів моделі. Розглянувши $\sum_{j=1}^{n} \rho_{ij}$ як параметр моделі, маємо можливість уточнити кореляційні залежності між акціями на волатильному фінансовому ринку.

Модифікацією цього підходу може слугувати алгоритм розрахунку середніх значень параметрів на відрізку $[t_0, t_0 + T_k]$ за достатньо значної їх зміни в межах вказаного інтервалу

$$(\alpha_1 SM_{ind}(t) + \alpha_2 I(t) + \sum_{j=1}^{n} \rho_{ij}) r(t) = \xi(t) - \frac{dr}{dt}$$

Висновки до розділу

У розділі подано оптимальну модель для оцінки ціни на активи в інвестиційному портфелі, що дасть максимальний прибуток для трейдера. Тобто подана математична модель може гарантувати максимальний прибуток за правильного вибору активів.

4. Прогнозування за допомогою теорії хвиль Елліота

4.1. Теорія хвиль Елліота

Хвильва теорія Елліота – одна з найпвідоміших теорій аналізу та прогнозування фінансових ринків. Незважаючи на складність її інтерпретації, вона є одним з найбільш популярних методів технічного аналізу.

Елліот почав спостерігати за фондовим ринком та більш детально вивчати рух показників починаючи з 1930-х років. Вивчаючи рух індексів протягом багатьох років, Елліот знайшов певну структуру всередині цих рухів, які були подібними. Шляхи та способи досягнення цінами нових максимумів та мінімумів повторювалися з певною періодичністю. Проводячи спостереження з великим ентузіазмом, Елліот виявив хвильову структуру руху цін акцій. Почавши з простої системи, він з'ясував багато подробиць щодо структури і розміру хвиль, методів їх визначення та усунення спотворень.

Система надзвичайно важка для застосування, оскільки хвилі відрізняються по довжині, інтенсивності і часу. Їх розпізнавання є однією з головних проблем технічного аналізу. Оскільки система з хвиль Елліота містить хвилі всередині хвиль, хвилі всередині хвиль і так далі, то навіть малі помилки при розрахунках можуть призвести до критичних помилок у висновках щодо поточного стану ринку.

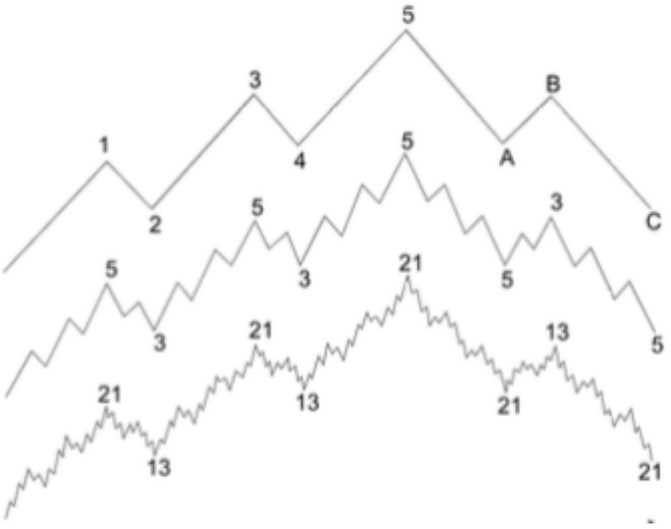
За теорією Елліота, ринок рухається циклами, які є самоподібними, тобто великі хвилі складаються з менших хвиль, які містять менші хвилі, і так далі. Назви хвиль розділяються згідно з їх розміром, проміжком у часі, величині періоду.

Основні правила, визначені Елліотом:

- Хвилі, напрям яких збігається з головним трендом, складаються з п'яти менших хвиль.
- Коригувальні хвилі, хвилі, спрямовані проти основного тренда складаються з трьох менших хвиль.

Ця основна концепція буде повторюватися на будь-якому рівні хвиль знову, вона є однаковою для хвиль будь-якого розміру. Система схожа на всіх рівнях тривалості, задача спеціаліста з технічного аналізу одна – правильний відлік хвилі.

На наступному рисунку зображено типову хвилю Елліота:



Малюнок 1. Типові хвилі Елліота

Оскільки ринки не рухаються ідеально, то корекційні хвилі можуть провалюватися нижче попередніх коригувань. Окремі непарні частини можуть закриватися нижче ціни закриття попереднього імпульсу та інше. Ці спотворення визначають виявлення та ідентифікацію хвиль вкрай

важким завданням, що багато в чтому залежить від самаго спеціаліста з технічного аналізу. Найбільш цікава завжди п'ята хвиля, оскільки вона показує на розворот тенденції. Бувають випадки, коли п'ята хвиля розтягується, поділяється на п'ять хвиль меншого розміру. Це поділ зазвичай характеризує ринок з сильною тенденцією. Для спотворень ідеальної хвильової структури Елліот сформував досить завершену несуперечливу теорію з трикутників, діагоналей, корекцій і розширень. Оскільки спотворені хвилі можуть йти одина за одною, то їх виявлення є найважчим завданням хвильового аналізу. Система може вірно працювати на довгостроковому періоді прогнозування, але давати збої під час аналізу коротких інтервалів. Хвильова теорія Елліота дозволяє дізнатися причину помилки лише в тому випадку, коли до неї включають всі елементи, які були виявлені та вивчені. Її завжди можна розпізнати. В цьому криється як потужність цієї теорій, так і складність її застосування.

4.2 Вейвлет-аналіз для теорії хвиль Елліотта

Вейвлет аналіз є одним із видів спектрального аналізу. Роль базових фунцій відіграють вейвлети. Базисом вейвлету є функція наступного типу:

$$\psi_{ab}(\mathrm{t}) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right)$$

де b – зсув, a – масштаб.

Також функція повинна мати нульову площу та бажано нульові моменти. Вейвлети використовуються для представлення сигналів і функцій у вигляді суперпозиції вейвлетів.

Спектр вейвлета C(a,b) на відміну від спектра Фур'є є функцією двох змынних: масштабу *a* та зсуву *b*. Параметри можуть приймати будь-які значення в межах областей їх визначення.

Неперервне вейвлет перетворення визначається наступним чином:

$$T(\mathrm{a},\mathrm{b}) = \frac{1}{\sqrt{|a|}} \int \mathrm{x(t)}\psi^{*}\left(\frac{t-b}{a}\right)$$

де ψ* – комплексне спряження.

Найбільш поширені базисні функціє утворені з похідних вейвлета. Вибір конкретного вейвлета повністю залежить від поставленої задачі.

Результатом вейвлет-перетворення одновимірного сигналу є двовимірний масив значень *C(a,b)*. Розподіл цих значень у просторі *(a,b)* вейвлет-перетворенням або вейвлет-спектром. Спектр *C(a,b)* одновимірного сигналу можна уявити як поверхню в тривимірному просторі. Є різні способи візуалізації результату, серед яких найпоширеніший – проекція на площину з ізолініями (ізорівнями), що дозволяє побачити картину локальних екстремумів. При широкому діапазоні значень часто застосовують логарифмічні координати.

Висновки до розділу

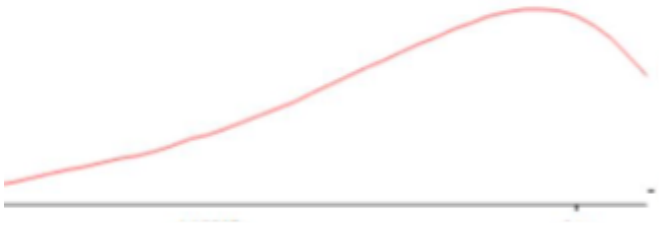
Тут розглянуто теорію хвиль Елліота та причини чому саме ця теорія вдало підходить для прогнозування та оцінки руху цін на активи на фінансових ринках. Також розглянуто вейлет-аналіз для цієї теорії, що допомагає аналізувати отримані дані за допомогою цієї теорії.

5. Аналіз однієї з програм для автоматичної торгівлі з відкритим кодом

Дані були взяті з сайту https://finance.yahoo.com/sector/technology. Для тесту та прогнозування я обрала часовий ряд котирувань акцій компанії Microsoft.

Малюнок 2. Графік обраного для тестування програми активу - акцій компанії Microsoft

Було проведено близько 10 тестів з програмою на основі історичних даних обраного активу. Далі наведено найближчий по часу результат роботи. Обрана програма для прогнозування надає наступні результати на інтервалі від 15 березня 2020 року до 15 квітня 2020 року:



Малюнок 3. Графік результат роботи програми для прогнозування

Тобто вказує нам на зниження ціни активу, проте не вказує точної ціни на обраний момент - 15 квітня. Тобто програма вказує нам лише на тенденцію падіння ціни активу, що згодом справджується. Станом на 15-17 квітня ціна на акції Microsoft дійсно падає.

В обраній програмі з відкритим програмним кодом застосований підхід прогнозування за допомогою нейронних мереж, адже є момент навчальної вибірки для подальшого тестування, та застосовується теорія хвиль Елліота з вейлет-аналізом. В додатку А міститься програмний код даної програми, котру було використано.

Щодо роботи даної програми можна сказати, що вона відповідає всім вимогам, які були поставлені мною до програми - швидкість роботи, інтуїтивно зрозумілий інтерфейс та налаштування, проте сама програма вказує лише на тенденцію падіння чи зростання ціни активу, та задопомогою неї неможливо оцінити тенденції на отримання прибутку від інвестиційного пакету.

Висновки до розділу

На основі вивченого матеріалу з прогнозування було вивчено та протестовано програму для автоматизації торгівлі. Результати роботи дійсно правдиві, проте програма лише вказує на напрямок зміни ціни активу - вона вказує на падіння та зростання ціни на актив.

Висновок

Все частіше тенденція щодо збільшення інтересу до нейронних мереж у математичному моделюванні, адже їх застосування відкриває нові можливості в прогнозуванні у сферах, що стосуються економіки та фінансів. До однієї з таких задач відноситься і прогнозування тенденцій зміни часових рядів. Враховуючи складність розрахунків, актуальність автоматизації стає очевидною.

Проаналізувавши стан вже існуючих інформаційних систем нейромережевого моделювання, слід відзначити, що для практичного використання в основному застосовують легкі та середні автоматизовані системи, які мають не високий рівень наочності. Оцінювана мною програма поєднує в собі підходи важких систем, але в той же час є досить легким у використанні.

Оцінюваний прототип є більш ефективним за людину-брокера, адже не схильний до емоцій та азарту, не потребує сну і відпочинку, швидкий в аналізі та реакції на зміни. За допомогою цієї програми можна приймати рішення ще до того, як людина-брокер помітить зміни в котируванні цін на активи.

Мета даної роботи досягнута, було розглянуто програму-прототип та розібрано найкращі шляхи прогнозування для оцінки ціни активу на фінансових ринках. На основі програмного коду було проаналізовано теоретичний матеріал, що стосувався всіх методів застосованого в прогнозуванні. Проте, як автор роботи, застерігаю від торгівлі на фінансових ринках із сумнівною репутацією. Та довіряйте програмам для

автоматизованої торгівлі не більше, ніж людині, що обіцяє вам мільйон через тиждень без жодних зусиль.

Список літератури

1. A. J. Frost Elliott Wave Principle: Key to Market Behavior. / A. J. Frost, R. R. Prechter — Delhi: Elliott Wave International, 2005. — 190 p.
2. Джозеф Т. Упрощенный Анализ Волны Эллиота. / Джозеф Т. — СПб.: Литера, 2012. — 80 с.
3. S. B. Achelis Technical Analysis from A to Z. / S. B. Achelis — Probus: Probus Pub, 1995. — 80 p.
4. Короновский А.А. Непрерывный вейвлетный анализ и его приложения. / Короновский А. А., Храмов А.Е. — М.: Физматлит, 2003. — 158 с.
5. H. J. Nussbaumer Fast Fourier Transform and Convolution Algorithms. / H. J. Nussbaumer — London: Springer, 1982. — 240 p.
6. Roads C. The computer music tutorial / Roads C. — Boston: The MIT Press, 1996. — 1256 p.
7. P. Bloomfield Fourier Analysis of Time Series: An Introduction. / P. Bloomfield — Boston: The MIT Press, 2014. — 288 p.
8. B. Mandelbrot Fractals in Petroleum Geology and Earth Processes. / B. Mandelbrot — New York: P.R. La Pointe, 1969. — 969 p.
9. А.Н. Ширяев Вероятность. / А.Н. Ширяев — М.: Физматлит, 2012. — 520 с
10. Д. Возный Код Эллиотта: волновой анализ рынка FOREX / Д. Возный – М.: Омега-Л, 2006. – 240 с.
11. Вітлінський В.В. Економічний ризик: ігрові моделі [Текст]: навч. посіб. / В.В. Вітлінський, П. І. Верченко, А.В. Сігал. – К. : КНЕУ, 2002. – 446 с. ISBN 966–574–318–X;

Додаток А Програмний код, застосований для перевірки

```
import numpy as np
import matplotlib.pyplot as plt
import os
import pywt
import datetime

elliot_waves = []
wavelets = pywt.families()
def showPlot(date, data, file_name):
fig, ax = plt.subplots()
ax.plot(date, data)
fig.savefig(file_name)
plt.close(fig)
def generate_elliot_waves(folder):
    common_folder = 'static/results/'
    folder_name = 'elliot/'
def trend(data):
    return 0 * np.arange(len(data))
def print_wave(data, file_name):
    date = np.arange(len(data))
    return showPlot(date, data, file_name)
def print_fft(data, file_name):
    A = np.fft.fft(data)
    frrAbs = np.abs(A)
    return print_wave(frrAbs, file_name)
def get_filter(arr):
    return list([x for x in arr])
def print_wavelet(data, wavelet_name, filter, file_name):
    result = pywt.dwt(data, wavelet_name)
    if filter == 'high':
        return print_wave(result[0], file_name)
    elif filter == 'low':
        return print_wave(result[1],file_name)
def build_elliot_waves(path):
```

```
    x = [1, 3, 2, 4]
    z = [1, 3, 2, 4, 3, 5]
    if not os.path.exists(os.path.abspath(path)):
        os.makedirs(os.path.abspath(path))
    minus_x = np.subtract(np.max(x), x)
    minus_z = np.subtract(np.max(z), z)
    elliot_waves.append('x')
    elliot_waves.append('z')
    elliot_waves.append('minus_x')
    elliot_waves.append('minus_z')
def generate_elliot_waves(scale=4):
    import random


    w1 = np.zeros(6)
    w2 = np.zeros(4)
    w1[1] = int(random.randint(3,scale))
    w1[2] = int(random.randint(1, w1[1]- 1))
    valid = True
 while valid:
    temp = int(random.randint(1,3)) * int(random.randint(1, scale))
    if w1[1] >= w1[2] + temp:
        continue
    w1[3] = int(random.randint(w1[1], w1[2] + temp))
    if (w1[3] - w1[2]) > w1[1]:
        valid = False
    w1[4] = int(random.randint(w1[1], w1[3]))
    w1[5] = w1[4] + w1[1]
    w2[0] = w1[5]
    valid = True
    result = list(w1) + list(w2)[1:]
    proliferated_result = [] #
    multiply number of points by
    step_number
    step_number = 20
 for i in range(1, len(result)):
    for j in range(0, step_number):
        proliferated_result.append(result[I - 1] + (float(j) * (result[i] -
        result[i - 1])) / step_number)
        result = proliferated_result
        result_m =np.subtract(np.max(result), result)
 return result, result_m
 def generate_elliot_waves_wrapper(scale=4):
    print("generate_elliot_waves_wrapper")
    value, _ = generate_elliot_waves(scale)
    base = datetime.datetime(2001, 1, 1)
```

```
date = [base + datetime.timedelta(days=x) for x in range(0, len(value))]
date = list(date)
print(date)
value = np.array(value)
return date, value
```

transform.py

```
from __future__ import division
import numpy as np
import scipy
import scipy.signal
import scipy.optimize
import scipy.special
from .wavelets import Morlet
__all__ = ['cwt', 'WaveletAnalysis','WaveletTransform']
def cwt(data, wavelet=None,widths=None, dt=1, frequency=False,axis=-1):
    if widths is None:
        raise UserWarning('Have␣to␣specify␣some␣widths␣(scales)')
    if not wavelet:
        raise UserWarning('Have␣to␣specify␣a␣wavelet␣function')
    if frequency:
```

```
            slices = [slice(None) for _ inout.shape]
            slices[axis] = slice(None, N)
        if data.ndim == 1:
            return out[slices].squeeze()
        else:
            return out[slices]
    class WaveletTransform(object):
        def __init__(self, data=None, time=None, dt=1, dj=0.125, wavelet=Morlet(),
unbias=False, mask_coi=False, frequency=False, axis=-1):
            self.data = data
            if time is None:
                time = np.indices((data.shape[axis],)).squeeze() * dt
            self.time = time
            self.anomaly_data    =    self.data    -    self.data.mean(axis=axis,
keepdims=True)
            self.N = data.shape[axis]
            self.data_variance = self.data.var(axis=axis, keepdims=True)
            self.dt = dt
            self.dj = dj
            self.wavelet = wavelet
            self.cwt = cwt
            self.frequency = frequency
            self.unbias = unbias
            self.mask_coi = mask_coi
            self.axis = axis
        @property
        def fourier_period(self):
            return getattr(self.wavelet,'fourier_period')
        @property
        def fourier_periods(self):
            return self.fourier_period(self.scales)
        @property
```

```
def s0(self):
    if not hasattr(self, '_s0'):
        return self.find_s0()
    else:
        return self._s0
@s0.setter
def s0(self, value):
    setattr(self, '_s0', value)
    def find_s0(self):
    dt = self.dt
    def f(s):
    return
    self.fourier_period(s) - 2 * dt
    return scipy.optimize.fsolve(f,1)[0]
@property
def scales(self):
    if not hasattr(self, '_scales'):
        return self.compute_optimal_scales()
    else:
        return self._scales
@scales.setter
def scales(self, value):
    setattr(self, '_scales', value)
```

```
def compute_optimal_scales(self):
    dt = self.dt
    dj = self.dj
    s0 = self.s0
    J = int((1 / dj) *
    np.log2(self.N * dt / s0))
    sj = s0 * 2 ** (dj *
    np.arange(0, J + 1))
    return sj
    dt = self.dt
    N = self.N
    a = 2 * np.pi / (N * dt)
    if k is None:
        k = np.arange(N)
        w_k = np.arange(N) * a
        w_k[np.where(k > N // 2)]*= -1
    elif type(k) is np.ndarray:
        w_k = a * k
        w_k[np.where(k > N // 2)]*= -1
    else:
        w_k = a * k
    if k <= N // 2:
        pass
    elif k > N // 2:
        w_k *= -1
    return w_k
@property
def wavelet_transform(self):
    if self.frequency:
        wavelet = self.wavelet.frequency
    else:
        wavelet = self.wavelet.time
    return   self.cwt(self.anomaly_data,   wavelet=wavelet,   widths=widths,
```

```
dt=self.dt, frequency=self.frequency, axis=self.axis)
            @property
            def wavelet_power(self):
                if self.unbias:
                    return (np.abs(self.wavelet_transform).T** 2 / self.scales).T
                elif not self.unbias:
                    return np.abs(self.wavelet_transform) ** 2
            def reconstruction(self, scales=None):
                dj = self.dj
                dt = self.dt
                C_d = self.C_d
                Y_00 = self.wavelet.time(0)
            if scales is not None:
                old_scales = self.scales
                self.scales = scales
                s = self.scales
                W_n = self.wavelet_transform
            if scales is not None:
                self.scales = old_scales
                real_sum = np.sum(W_n.real.T /s ** .5, axis=-1).T
                x_n = real_sum * (dj * dt ** .5 / (C_d * Y_00))
                x_n += self.data.mean(axis=self.axis, keepdims=True)
```

```
    return x_n
@property
def global_wavelet_spectrum(self):
    if not self.mask_coi:
        mean_power = np.mean(self.wavelet_power, axis=1)
    elif self.mask_coi:
        mean_power = self.coi_mean(self.wavelet_power, axis=1)
    var = self.data_variance
    return mean_power / var
def coi_mean(self, arr, axis=1):
    s = self.scales
    t = self.time
    T, S = np.meshgrid(t, s)
    inside_coi = (coi(S) < T) & (T < (T.max() - coi(S)))
    mask_power = np.ma.masked_where(~inside_coi, self.wavelet_power)
    mask_mean = np.mean(mask_power, axis=axis)
    return mask_mean
@property
def C_d(self):
    if hasattr(self.wavelet, 'C_d'):
        return self.wavelet.C_d
    else:
        return self.compute_Cdelta()
def compute_Cdelta(self):
    dj = self.dj
    dt = self.dt
    s = self.scales
    W_d = self.wavelet_transform_delta
    Y_00 = self.wavelet.time(0)
    real_sum = np.sum(W_d.real / s ** .5)
    C_d = real_sum * (dj * dt ** .5 / Y_00)
    return C_d
```

```
            _
        @property
        def wavelet_variance(self):
            dj = self.dj
            dt = self.dt
            C_d = self.C_d
            N = self.N
            s = np.expand_dims(self.scales, 1)
            A = dj * dt / (C_d * N)
            var = A * np.sum(np.abs(self.wavelet_transform) ** 2 / s)
            return var
        @property
        def coi(self):
            Tmin = self.time.min()
            Tmax = self.time.max()
            Tmid = Tmin + (Tmax - Tmin) / 2
            s               =              np.logspace(np.log10(self.scales.min()),
np.log10(self.scales.max()), 100)
            c1 = Tmin + self.wavelet.coi(s)
            c2 = Tmax - self.wavelet.coi(s)
            C = np.hstack((c1[np.where(c1 < Tmid)], c2[np.where(c2 > Tmid)]))
            S = np.hstack((s[np.where(c1 < Tmid)], s[np.where(c2 > Tmid)]))
            iC = C.argsort()
            sC = C[iC]
```

```
        sS = S[iC]
        return sC, sS
def plot_power(self, ax=None, coi=True):
    import matplotlib.pyplot as plt
    if not ax:
        fig, ax = plt.subplots()
    Time, Scale = np.meshgrid(self.time, self.scales)
    ax.contourf(Time, Scale, self.wavelet_power, 100)
    ax.set_yscale('log')
    ax.grid(True)
    if coi:
        coi_time, coi_scale = self.coi
    ax.fill_between(x=coi_time,
    y1=coi_scale,
    y2=self.scales.max(),
    color='gray',
    alpha=0.3)
    ax.set_xlim(self.time.min(),
    self.time.max())
    return ax
WaveletAnalysis = WaveletTransform
```

wawelets.py

```
from __future__ import division
import numpy as np
import scipy
import scipy.signal
import scipy.optimize
import scipy.special
from scipy.misc import factorial
__all__ = ['DOG', 'Morlet', 'Paul', 'Ricker', 'Ricker']
class Morlet(object):
```

```
class Morlet(object):
    def __init__(self, w0=6):
        self.w0 = w0
        if w0 == 6:
            self.C_d = 0.776
    def __call__(self, *args, **kwargs):
        return self.time(*args, **kwargs)
    def time(self, t, s=1.0, complete=True):
        w = self.w0
        x = t / s output = np.exp(1j * w * x)
        if complete:
            output -= np.exp(-0.5 * (w ** 2))
            output *= np.exp(-0.5 * (x ** 2)) * np.pi ** (-0.25)
        return output
    def fourier_period(self, s):
        x = w * s
        Hw = np.array(w)
        Hw[w <= 0] = 0
        Hw[w > 0] = 1
        return np.pi ** -.25 * Hw * np.exp((-(x - self.w0) ** 2) / 2)
    def coi(self, s):
        return 2 ** .5 * s


        class Paul(object):
            def __init__(self, m=4):
                self.m = m def __call__(self, *args, **kwargs):
                return self.time(*args, **kwargs)
            def time(self, t, s=1.0):
                m = self.m
                x = t / s
                const = (2 ** m * 1j ** m * factorial(m)) / (np.pi * factorial(2 * m))
  ** .5
                functional_form = (1 - 1j * x) ** -(m + 1)
                output = const * functional_form
                return output
            def fourier_period(self, s):
                return 4 * np.pi * s / (2 * self.m + 1)
            def frequency(self, w, s=1.0):
                m = self.m
                x = w * s
                Hw = 0.5 * (np.sign(x) + 1)
                const = 2 ** m / (m * factorial(2 * m - 1)) ** .5
                functional_form = Hw * (x) ** m * np.exp(-x)
                output = const * functional_form
                return output
            def coi(self, s):
                return s / 2 ** .5
        class DOG(object):
            def __init__(self, m=1):
                if m == 2:
                    self.C_d = 3.541
                elif m == 6:
                    self.C_d = 1.966
                else:
                    pass
                self.m = m
```

```
    He_n = scipy.special.hermitenorm(m)
    gamma = scipy.special.gamma
    const = (-1) ** (m + 1) /
    gamma(m + 0.5) ** .5
    function = He_n(x) * np.exp(-x ** 2 / 2)
    return const * function
def fourier_period(self, s):
    return 2 * np.pi * s / (self.m + 0.5) ** .5
def frequency(self, w, s=1.0):
    m = self.m
    x = s * w
    gamma = scipy.special.gamma
    const = -1j ** m / gamma(m + 0.5) ** .5
    function = x ** m * np.exp(-x ** 2 / 2)
    return const * function
def coi(self, s):
    return 2 ** .5 * s
```

```
def frequency(self, w, s=1.0):
        m = self.m
        x = w * s
        Hw = 0.5 * (np.sign(x) + 1)
        const = 2 ** m / (m * factorial(2 * m - 1)) ** .5
        functional_form = Hw * (x) ** m * np.exp(-x)
        output = const * functional_form
        return output
```