НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
Інститут прикладного системного аналізу

Кафедра математичних методів системного аналізу

«До захисту допущено»

В.О.Завідувача кафедри

_____ О.Л. Тимощук

Дипломна робота

на здобуття ступеня бакалавра

з напряму підготовки 6.040303 Системний аналіз

на тему: «Моделі у формі нейронних мереж в задачах прогнозування розвитку фінансових процесів»

Виконав (-ла):
студент (-ка) IV курсу, групи КА-41
Барзій Ілля Ігорович _____

Керівник:
д.т.н., проф. Бідюк Петро Іванович _____

Консультант з економічного розділу:
доцент, к.е.н. Рощина Н. В. _____

Консультант з нормоконтролю:
доцент, к.т.н. Коваленко А.Є. _____

Рецензент:

_____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2018 року

РЕФЕРАТ

Дипломна робота: 93с.,8 табл., 29 рис., 2 дод., 16 джерел.

НЕЙРОННІ МЕРЕЖІ, МАШИННЕ НАВЧАННЯ, АНАЛІЗ ЧАСОВИХ РЯДІВ, КОРОТКОСТРОКОВЕ ПРОГНОЗУВАННЯ ФІНАНСОВИХ ПРОЦЕСІВ.

Метою виконання роботи є покращення методів прогнозування розвитку фінансових процесів різного характеру з використанням моделей нейронних мереж.

В роботі проведено огляд існуючих методів, що використовуються для розв'язання такого типу задач, проведено аналіз алгоритмів, на яких базується функціонування нейронних мереж, виконана імплементація і тестування різних архітектур моделей, зроблені висновки за результатами виконання обчислювальних експериментів.

Результатом роботи є побудовані моделі досліджуваних процесів, аналіз ефективності побудованих моделей нейромереж для прогнозування часових рядів, розроблений власний варіант нейронної мережі з використанням двох етапів навчання, виконано порівняльний аналіз запропонованої моделі з моделлю АРІКС та аналіз ефективності її використання для короткострокового прогнозування. В роботі реалізовано програмний модуль для тестування моделі з використанням бібліотеки deepnet R.

ABSTRACT

For the bachelors work of Illya Igorevych Barziy

"Neural networks models in the problems of financial processes forecasting"

The bachelors work consists of 93 p., 29 fig., 8 tabl., 2 append., 16 sources.

NEURAL NETWORKS, MACHINE LEARNING, TIME SERIES ANALYSIS, SHORT-TERM FINANCIAL PROCESS FORECASTING.

The purpose of given thesis is to improve the methods of forecasting the development of financial processes of different nature using neural network models.

In this work, research of the current methods used for financial forecasting had been carried out. The analysis of underlying processes, on which neural network models are based on, the implementation and testing of various types of them were completed, conclusions based on computational experiments were made.

The results of the study carried out is provided in the form of new models of neural networks and the analysis of the effectiveness of built neural networks models used to forecast time series; development of original neural network with two stages of learning, it's short-term forecasting results were compared with the same for ARIMA models. In this work the program module for neural network testing was build using deepnet R library.

ЗМІСТ

ВСТУП

Прогнозування розвитку фінансових процесів – важлива задача у багатьох прикладних та теоретичних галузях. Найчастіше задачею передбачення займаються спеціалісти у фінансових установах, таких як банки, інвестиційні фірми тощо. Метою таких дій є поліпшення результатів роботи організацій, прийняття кращих та більш обґрунтованих рішень, досягнення вищих прибутків. Швидкий розвиток технологій штучного інтелекту в останні кілька десятків років відкриває нові методи, що можуть бути застосовані в широкому спектрі задач, в тому числі – задачах прогнозування.

Сучасні фінансові процеси описуються складними математичними моделями, які враховують детерміновані і стохастичні тренди, змінну у часі дисперсію, сезонні ефекти і різні режими функціонування. Задача моделювання і прогнозування фінансових процесів, представлених часовими рядами, є високо актуальною, оскільки результати їх дослідження використовуються на мікро- і макрорівнях, на біржах тощо.

Ідея застосування нових технологічних винаходів у фінансовій сфері не є новою. Незважаючи на те, що нові технології нейронних мереж доступні для використання широкому кругу людей вже багато років (раніше, через недостатню потужність продуктів, націлених на широкий круг споживачів та мале розповсюдження програмних продуктів, що дозволяють легке створення моделей нейромереж, дослідження в цій сфері були доступні лише спеціалізованим інституціям), все ще не існує в науковій літературі деталізованого та обґрунтованого дослідження застосування нейромереж для передбачення фінансових процесів. Звичайно, є окремі роботи, що розглядають результати конкретних моделей мережі на окремих вхідних даних та аналізують результат.

Проте, через різноманітність як моделей нейромереж, так і фінансових даних, важко одразу зрозуміти, які моделі та підходи доцільніше

використовувати для конкретно обраної задачі. Часто спеціалістам цієї сфери доводиться застосовувати простий перебір параметрів. Окрім того, погано визначені можливості самого прогнозування.

Об’єктом дослідження є часові ряди фінансового характеру, а саме, ціни на акції великих компаній, курси валют та ціни на сировину. Так, можна буде визначити, фінансові процеси якого характеру краще піддаються прогнозуванню обраними моделями.

Предметом дослідження є сучасні методи в області аналізу та прогнозування часових рядів, їх різновиди, особливості, принципи роботи. Окрім того, будуть розглянуті програмні модулі, з використанням яких ці методи реалізуються.

Таким чином, метою роботи є забезпечення покращення ефективності прогнозування розвитку фінансових процесів з використанням моделей нейронних мереж, а також описання ситуацій, в яких використання таких моделей є доцільним.

При роботі над досягненням мети, поставимо наступні задачі:
1. Дослідити існуючі напрацювання в досліджуваній області та провести їх порівняння. Визначити бажаний рівень точності прогнозів у поставленій здачі;
2. Розробити моделі нейронних мереж, що підходять для застосування в обраній сфері досліджень, обрати інструменти для їх реалізації та тестування;
3. Знайти, обробити актуальні фінансові дані, підготувати їх для завантаження в модель описати процес побудови моделі та параметрів, що для цього використовуються;
4. Проаналізувати та представити результати отримані при використанні запропонованої моделі, порівняти їх з результатами моделі АРІКС;
5. Обґрунтувати доцільність використання обраних програмних методів та оцінити вартість застосування створеного продукту в майбутньому.

РОЗДІЛ 1 АНАЛІЗ ПОСТАВЛЕНОЇ ЗАДАЧІ ТА МЕТОДІВ ЇЇ РОЗВ'ЯЗАННЯ

1.1 Аналіз існуючих методів прогнозування часових рядів

Через актуальність даної теми, багато дослідників та торговців на біржі використали розробки в області статистичного аналізу та машинного навчання, щоб отримати більший прибуток від своїх інвестицій. Проблема пошуку актуальної літератури та досліджень частково полягає в тому, що будь-яка особа, що застосовує такі методи для отримання прибутку, не зацікавлена в розкриванні їх, бо має на меті збереження прибутку. Проте, все ж існують дослідники, які опублікували свої роботи.

1.1.1 Статистичні методи

У роботі[1] розглянуто використання регресії для вивчення зв'язку між новинами та змінами цін на акції. Дослідник мав на меті підвищити продуктивність звичайного процесу прогнозування цін на акції. Автор розв'язує задачу регресії щотижневих цін акцій з використанням новин з попереднього тижня. Результати показують, що існує істотна кореляція між змінами цін на акції та загальною оцінкою характеру новин на початку тижня. Так, новини за тиждень конвертуються в числове значення , що лежить в діапазоні (-5, 5). Збільшення вартості новин на одиницю призводить до збільшення ціни акцій на 6,33 одиниці(розглядалися акції компанії Apple). Проте автор зауважує, що значення R-squared доволі мале (0.09243). Це означає, що модель не передбачає цінові змін із задовільною точністю.

Торговці та дослідники часто використовують статистичні методи, такі як авторегресія з ковзним середнім(ARIMA) для прогнозування фінансових часових рядів. Моделі ARIMA працюють на нестаціонарних часових рядах, шляхом

застосування моделі ARMA на стаціонарному ряді, котрий був отриманий із вихідного шляхом різницевих перетворень.

1.1.2 Методи машинного навчання

Машинне навчання – це наука, що досліджує моделі на основі яких комп'ютери приймають рішення, не будучи явно запрограмованими для цього. Ці моделі показали високі результати в таких областях, як машинне розпізнавання мови, самокеровані автомобілі, розпізнавання образів тощо[2].

Сильна сторона галузі машинного навчання полягає в тому, що його можна застосувати для розв’язання широкого класу проблем. Методи машинного навчання можна розділити на: навчання з вчителем, навчання без вчителя та навчання з підкріпленням(reinforced learning).

При навчанні з вчителем, дані, що передаються алгоритму, являються розміченими. Тобто явно вказано, які данні належать до якого класу. Методи, що належать до такого типу – дерева рішень, регресія, опорні вектори та наївниі байєсові класифікатори.

При навчанні без вчителя данні не розмічені, а завдання алгоритму – їх класифікувати. До такого типу належать – методи к-середніх, гаусівські змішані методи (K-Means and Gaussian mixture).

У разі навчання з підкріпленням, алгоритму надається оцінка його дій в певній формі. Так, алгоритм вчиться приймати кращі рішення в залежності від своїх дій. Методи такого типу – деякі нейронні мережі.

Для розв’язання обраної задачі підходять методи першого та третього типу. Проте останні зазвичай потребують значних обчислювальних ресурсів та є доволі складними в налаштуванні.

1.1.3 Методи опорних векторів

У галузі машинного навчання, методи опорних векторів (SVM)[3] стають все більш популярними. Методи опорних векторів вперше були розроблені Володимиром В. Вапніком та Олексієм Я. Червоненкісом в 1963 році. Ці методи змогли класифікувати лінійні проблеми набагато ефективніше, ніж штучні нейронні мережі. Пізніше, у 1992 році разом з ще двома вченими, Вапник запропонував метод, який називається трюком ядра(kernel trick). Так, дані методи змогли розв'язувати задачі нелінійної класифікації.

В роботі[4] методи опорних векторів використані для прогнозування напрямку зміни ціни на акції та значення економічних індексів. У дослідженні було показано, що точність прогнозу суттєво залежить від параметрів, що передаються у модель SVM. Найкращі результати були досягнуті при використанні радіальних базисних функцій як нелінійних класифікаторів. Значення параметра C також вплинуло на отримані результати. Автор порівняв результати методу опорних векторів з іншими методами, такими як нейронні мережі, що використовують зворотне поширення помилки, і виявив, що найкращі результати досягнені із застосуванням опорних векторів. Проте дане дослідження проводилося до створення нейронних мереж нового покоління.

1.1.4 Дерева рішень

Дерева рішень – класифікатор, який можна представити у формі дерева. Дерево можна розділити на кореневий вузол, внутрішні вузли та вузли листя. Дерева рішень базуються на евристиці "Бритви Оккама", тобто перевага надається короткому узагальненому дереву над складними деревами, які можуть бути перенавченими(overfitted).

В роботі[5] було розроблене текстове дерево рішень для прогнозування цін на фондовому ринку. Дані про ціну перетворюються в текстову форму відповідно до ряду правил. Наприклад, кожному індикатору надається значення оцінці при тренді вгору, тренді вниз або відсутності тренду. Виводом моделі є значення, яке повідомляє, купувати, утримувати або продавати акцію. Завдяки використанню цього текстового підходу в методі дерев рішень, автори досягли точності передбачення 90,82%. Вони зазначають, що такий підхід є більш ефективним за звичайне дерево рішень. Автори також зазначають, що модель котра ґрунтується текстовій інформації, призводить до більш простого, коротшого дерева, ніж у випадку звичайного дерева рішень.

Незважаючи на те, що дерева рішень досягли багатообіцяючих результатів у галузі передбачень поведінки фондового ринку, проблема перенавчання лишилася. В роботі [6] був використаний ансамбль випадкових лісів для протидії проблемі перенавчання дерев рішень. Автор стверджує, що це відбувається тому, що дерева рішень дуже чутливі до шуму в даних, з огляду на те, що вони мають дуже низький зміст і високу дисперсію. При використанні такого підходу, жодне з дерев не використовує всі тренувальні дані, вони поділяються між різними деревами в ансамблі. За допомогою цього методу автори досягають точності в діапазоні 85% і 95% в довгострокових прогнозах, що є поліпшенням результату, отриманого у випадку простих дерев рішень.

Варто зауважити, що незважаючи на заявлені високі точності прогнозу, в задачах прогнозування складних фінансових процесів така точність прогнозу здається малоймовірною, адже найкращі за точністю прогнози фінансових процесів такого рівня зазвичай знаходяться на рівні 80%[7]. Заявлені точності передбачень були отримані при дослідженні рядів, що мають значно простішу структуру.

1.1.5 Нейронні мережі

Дослідники в роботі[8] використовують багатоагентний підхід для прогнозування, а саме використовують технічний та фундаментальний аналіз, нейронні мережі та методи ліквідності для прогнозування ринкової поведінки. Вони порівнюють продуктивність стандартних нейромереж прямого поширення(Elmand та Jordan RNNs), та архітектуру, що називається NEAT (нейроеволюція доповнюючих топологій – Neuroevolution of augmenting topologies). Найкращі результати досягаються за допомогою архітектури NEAT, яка є генетичним алгоритмом, розробленим Кеном Стенлі. Ця методика намагається знайти баланс між різноманітністю розроблених рішень та їх відповідністю(fitness).

В [9] дослідники використовують штучну нейронну мережу для прогнозу курсу цін на індекс Nikkei 225, використовуючи ряд популярних технічних індикаторів. В рамках експрименту автори вирішують розділити характеристики технічних показників на два типи. Це робиться без зазначення критеріїв, використаних для поділу показників. Але вони названі "тип 1" та "тип 2". Тим не менш, автори повідомляють про точність 60,87% для набору даних типу 1 та 81,27% для набору даних типу 2 в задачі прогнозу напрямку руху цін на наступний день, що є досить гарним результатом з огляду на волатильність та нестабільність цін на фондовому ринку.

1.2 Опис процесів, що прогнозуються

Розглянемо постановку задачі прогнозування, тобто які результати бажано отримати від побудованої моделі. Потім опишемо які саме процеси обрано до прогнозування та їх особливості.

1.2.1 Вихідні результати прогнозування

Варіантів того, що можна розуміти під передбаченням поведінки фінансових процесів є доволі багато. В даній роботі сконцентровано увагу на можливість отримання прогнозів у форматі напрямку руху часового ряду. Так, модель буде на кожному кроці передбачувати напрямок розвитку фінансового процесу.

Таку задачу можна розглядати як проблему класифікації, адже кожен запис маємо віднести до двох класів – "вгору" або "вниз". Такий підхід було обрано через одночасно спрощення задачі зі збереженням інформативності прогнозу. Більш того, при побудові нейронних мереж, такий підхід дозволить спростити їх архітектуру та підбір параметрів, що позитивно вплине на час навчання мереж.

Іншим підходом є передбачення фактичної ціни в майбутньому, тобто на наступному кроці. Така постановка задачі є проблемою регресії, враховуючи характер виводу. При розв’язанні такої задачі отримаємо більше інформації, але це значно ускладнить побудову та навчання нейронних мереж. Проте, такий підхід може бути реалізований при подальшому дослідженні.

Задача прогнозування буде розв’язуватись для різних часових горизонтів, вхідних даних та їх комбінацій. Так, в модель будуть завантажуватися данні з різними періодами вимірювання, при відмінній побудові ввідних характеристик, при різних параметрах побудови моделі. Застосовуючи такий підхід стає можливим детальніше описати можливості обраної архітектури в задачах прогнозування та зазначити кращі параметри при її побудові. Завантажуючи в модель фінансові процеси різного характеру та аналізуючи результати можна оцінювати придатність використання такої моделі в задачах різного типу.

1.2.2 Дані та джерела їх отримання

В роботі досліджені можливості прогнозування різних часових рядів за своєю природою. Так, було обрано 4 групи часових рядів, що описують великі та значущі фінансові процеси в світі. Зокрема, це ціни на акції великих компаній, курси валют, економічні індекси країн та ціни на сировину. Прогнозування таких процесів вважається складним, оскільки на ціни впливають багато факторів.

Історичні ціни є загальнодоступними з різних джерел, проте в багатьох з них дані мають невисоку якість – присутні пропуски, стрибки цін тощо. Через це необхідно будо зібрати дані з різних ресурсів та порівняти їх між собою з метою визначення найкращого джерела для досліджень. Проблема полягає також в тому, що деякі ресурси спеціалізуються на окремих областях – наприклад, надають доступ до цін на акції, але не мають цін на сировину і навпаки.

Серед порівнюваних джерел, що розповсюджують всі необхідні дані, були: Yahoo Finance, Google Finance, Dukascopy. Порівнявши завантажені пробні дані з обраних ресурсів, було вирішено користуватися послугами Dukascopy через найменшу кількість неточних даних та простоту їх завантаження. Всі дані буди отримані у форматі CSV файлів з п’ятьма полями записів: часом, значенням відкриття, максимальним значенням, мінімальним значенням, значенням закриття.

Період, за який розглядаються фінансові процеси – минулий рік, з 20.03.2017 до 20.03.2018. Такий проміжок часу був обраний для того, щоб перевірити можливість прогнозування саме актуальних фінансових даних. Для кожного часового ряду завантажені дані на різних інтервалах – щогодинний, щоденний, щотижневий. Це дозволить зробити висновки про можливість прогнозування фінансових процесів такого характеру при зменшеній кількості інформації.

1.2.3 Ціни на акції

Ринок акцій характеризується великою кількістю учасників, високою волатильністю та малою здатністю до моделювання. Для дослідження були обрані часові ряди цін акцій великих компаній Америки, адже їх прогнозування є складною задачею через велику активність ринків та малу можливість маніпулювання цінами. Також, вони є доволі ліквідними, що означає, що результати прогнозування можна буде реально використати.

Прогнозування розвитку цін на акції малих компаній або акцій на українській біржі цінних паперів не розглядалося через можливість їх маніпуляції та меншу значущість результатів прогнозів. Проте, модель може бути побудована і на основі даних такого типу.

В даній роботі для прогнозу були обрані інформаційні та технологічні компанії: Amazon, Apple, Facebook.

1.2.4 Курси валют

Валютний ринок відрізняється від ринку акцій більшими об’ємами торгів, сильнішою волатильністю і рівнями шумів. Через це можливість прогнозувати такі фінансові процеси більш ускладнена. Тому варто проаналізувати можливості моделі мережі у фінансових процесах такого роду. Були обрані наступні пари валют, через їх значущість: EUR / USD (Євро – Долар США), GBP / USD (Британський Фунт – Долар США), BITCOIN / USD (Біткойн – Долар США), USD / JPY(Японська Єна – Долар США). Як можна відзначити, в якості базової валюти було обрано долар США, оскільки це одна з найважніших валют на світовій валютній біржі.

1.2.5 Світові показники (індекси)

Індексом називають міру змін на ринку цінних паперів країни або регіону[10]. Індекси складаються з гіпотетичного портфеля, що містить групу цінних паперів, які в цілому відображають ефективність всього ринку. Можна зазначити, що такі часові ряди мають малу волатильність через свій усереднений характер. Типові приклади таких показників включають Standard & Poor's 500 (S&P500) для Америки та STOXX Europe 50 для Європи. Для прогнозування були обрані саме ці два індекси, бо вони є двома основними в економіках Америки та Європи відповідно.

1.2.6 Ціни на сировину

Багато фінансових та економічних процесів залежать від цін на природні ресурси, вони формують ціни на продукти підприємств у виробничих та інших сферах. Наприклад, коефіцієнт кореляції цін акцій компаній авіаперевізників таких як Wizz Air та цін на нафту сягає 0.75. Так, будуть оцінені можливості прогнозування моделлю цін на основні сирові ресурси, зокрема ціни на золото, срібло та нафту.

1.3 Висновки до розділу 1

Були розглянуті актуальні методи та підходи до прогнозування часових рядів. Встановлено, що методи машинного навчання для розв’язання задач такого типу широко використовуються. З огляду на результати робіт, високою точністю передбачення руху одного з обраних фінансових процесів є 80%. Також знайдені джерела отримання даних, описані фінансові процеси та причини їх вибору.

РОЗДІЛ 2 ОГЛЯД МОДЕЛЕЙ НЕЙРОННИХ МЕРЕЖ

2.1 Застосування нейронних мереж

Штучний інтелект швидко розвивався в останні роки, машинне навчання та алгоритми класифікації стали більш доконаними та потужними. Ширшого застосування набувають підходи до машинного навчання, що називаються глибоким навчанням. Глибоке навчання полягає у вивченні ієрархічної структури даних, спочатку виокремлюючи прості ознаки низького рівня, які, у свою чергу, використовуються для побудови більш складних взаємозв’язків, що описують основні закономірності даних.

Нейромережі застосовуються для вирішення широкого кола завдань, пов'язаних з обробкою образів. Прикладами типових завдань для яких використовуються моделі нейронних мереж є:

- Регресія – пошук наближених значень функції;
- Класифікація – розділення даних по встановленим класам;
- Кластеризація – розділення даних по невідомим заздалегідь класам;
- Стиснення інформації;
- Відновлення втрачених даних;
- Асоціативна пам'ять;
- Оптимізація, оптимальне управління та ін.

Важливим елементом, що впливає на спосіб обробки інформації, є наявність або відсутність в мережі петель зворотного зв'язку. Якщо зворотні зв'язки між нейронами відсутні (тобто мережа має структуру послідовних шарів, де кожен нейрон отримує інформацію тільки з попереднього шару), обробка інформації в мережі односпрямована. Вхідний сигнал обробляється

послідовністю шарів, і відповідь буде гарантовано обрахована через число тактів, рівне числу шарів.

Наявність зворотних зв'язків може зробити динаміку нейромережі (званої в цьому випадку рекурентною) непередбачуваною, тобто мережа може зациклитися і не видати відповіді ніколи.

Оскільки нейрони в рекурентних мережах по багато разів беруть участь в обробці інформації, це дозволяє таким мережам виконувати більш різноманітну і глибоку обробку інформації. Але в випадку їх використання слід вживати спеціальних заходів для того, щоб мережа не зациклювалася (наприклад, використовувати симетричні зв'язки, як в мережі Хопфілда[11], або примусово обмежувати число ітерацій).

У моделях нейромереж навчанням називається алгоритм, за яким змінюється архітектура мережі (комбінації зв’язків між нейронами) і ваг синоптичних зв'язків (вплив сигналів по цих зв’язках на інші нейрони) з метою покращення результатів виводу моделі. Процес навчання нейромережі виконується на вибірці, що називається тренувальною. В результаті навчання, мережа повинна покращувати вихідні сигнали по заздалегідь встановленим критеріям.

2.2 Основні правила навчання нейронних мереж

Існує чотири основних правила навчання, пов'язані з архітектурами мереж: корекція за похибкою, правило Больцмана, правило Хебба і метод змагання. Наведемо опис основних ідей цих правил. В даній роботі буде використовуватися перше з огляду на обрану архітектуру мережі.

2.2.1 Корекція за похибкою

Для кожного вхідного прикладу заданий вихід (цільовий), який може не збігатися з реальним (передбаченим) значенням. Правило навчання при корекції за похибкою полягає у використанні різниці між цільовою і передбаченою змінною для зміни ваг, з метою зменшення помилки неузгодженості. Варто зазначити, що при використанні цього правила навчання моделі проводиться тільки в разі помилкового результату. Існує велика кількість модифікацій цього правила навчання.

2.2.2 Правило Больцмана

Правило Больцмана – стохастичне правило навчання, природа якого пов’язана з принципами термодинаміки. При його використанні здійснюється настройка коефіцієнтів ваг зав’язків між нейронами відповідно до необхідного розподілу ймовірностей. Навчання за правилом Больцмана може бути розглянуто в якості окремого випадку корекції за похибкою.

2.2.3 Правило Хебба

Правило Хебба є давнім алгоритмом навчання нейронних мереж, суть якого полягає в наступному: якщо нейрони з обох сторін синапсу збуджуються в один і той самий час, то синоптичні ваги збільшуються. Зауважимо, що зміна синоптичних ваг в цьому випадку залежить тільки від пов'язаних з цим синапсом нейронів і їх активністю. Запропоновано велику кількість різновидів цього правила, що розрізняються особливостями модифікації синоптичних ваг.

2.2.4 Метод змагання

В цьому методі навчання вихідні нейрони змагаються один з одним. І вихідний нейрон з максимальним значенням зваженої суми є "переможцем". Виходи ж інших вихідних нейронів встановлюються в неактивний стан. При навчанні модифікуються тільки ваги нейрона – "переможця" в бік збільшення близькості до даного вхідного прикладу.

2.3 Зворотне поширення помилки

Зараз найчастіше використовується перше правило навчання. Серед його варіацій найчастіше використовують метод зворотного поширення помилки, що є доволі простим у реалізації та дозволяє навчити модель за задовільний час. Суть його в побудові функції помилки та налаштування ваг зав’язків муж нейронами в залежності від локального градієнта цієї функції.

Після кожного проходження даних по мережі, вираховується різниця між прогнозованими і розміченими даними, які отримуються на вихідному шарі. Помилки встановлюються на кожному шарі виходячи з оцінок наступного (рис 2.1). В результаті можливо визначити внесок кожного нейрона в загальну кінцеву помилку мережі. Часто використовують метод найшвидшого спуску, що означає зміну ваг зав’язків між нейронами пропорційно їх внеску і помилку.

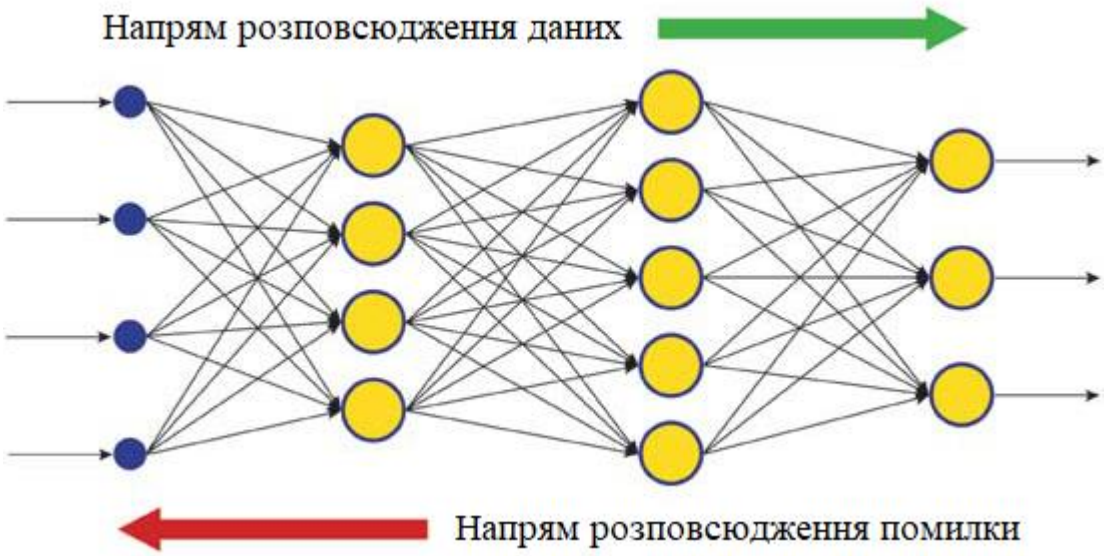


Рисунок 2.1 – Зображення методу зворотного поширення помилки.

Варто зауважити, що такий спосіб навчання не гарантує оптимального налаштування мережі через можливості потрапляння до локального мінімуму. Існує низка прийомів, використовуючи які можна уникнути зазначеної проблеми.

2.4 Глибоке навчання

В даний час теорія і практика машинного навчання переживають значні зміни, викликані успішним застосуванням методів Deep Learning (глибокого навчання). На відміну від класичних нейронних мереж 80-90-х років минулого століття, нові парадигми навчання дозволили уникнути багатьох проблем, які стримували поширення і успішне застосування традиційних нейронних мереж.

Мережі, навчені за допомогою алгоритмів глибокого навчання, не просто перевершили по точності кращі альтернативні підходи, а й у ряді завдань проявили зачатки розуміння сенсу інформації, що подається (наприклад, в задачах розпізнавання зображень, аналізу текстової інформації тощо).

Найбільш успішні сучасні методи комп'ютерного зору і розпізнавання мови побудовані на використанні глибоких мереж, а гіганти IT-індустрії, такі як Apple, Google, Facebook, наймають колективи дослідників, що займаються глибокими нейромережами для вирішення нових глобальних проблем.

2.5 Ідея глибокого навчання

Коли перед людиною поставлене певне завдання, воно часто розкладається на менші, простіші для вирішення задачі. Так, ситуація описується на різних рівнях абстракції. Наприклад, людина прагне ідентифікувати зображення. Так, особа спочатку виокремлює важливі особливості зображення, бачить спочатку силует людини на зображенні, помічає волосся на обличчі, форму тіла та одяг, і визначає, що образ є людиною. Тобто, особа ідентифікувала зображення, впізнавши у ньому певні особливості, такі як "має бороду", "має широкі плечі", "носить костюм", а потім класифікує зображення.

Самі того не усвідомлюючи, люди спочатку розглядають менші характеристики зображень, наприклад, лінії, криві та краї для визначення характеристик вищого рівня. Ці характеристики, від простих до складних, організовані в шари, складають глибоку мережу.

Глибоке навчання відноситься до моделей навчання, які використовують багато рівнів ієрархій функцій. Таким чином, глибока нейронна мережа – це багатошарова мережа, що складається з вхідних і вихідних шарів, а також численних прихованих шарів. Ці приховані шари складаються з прихованих нейронів, які можуть бути використані для опису основних закономірностей в даних. Процес навчання глибоких нейромереж спрямований на автоматичне виявлення цих абстракцій, від найнижчого до найвищого рівня. Здатність

автоматично виокремити важливі характеристики обумовлює популярність нейронних мереж з глибокою архітектурою.

2.6 Побудова нейронної мережі

Базовим елементом нейронної мережі є нейрон. На рис. 2.2 зображений перцептрон – найпростіша нейронна мережа, що складається з простого нейрона, вона приймає n вхідних значень, $x_1, x_2, \ldots, x_n$, і значення зміщення, що є константою та відповідає за зсув функції активації ліворуч або праворуч.

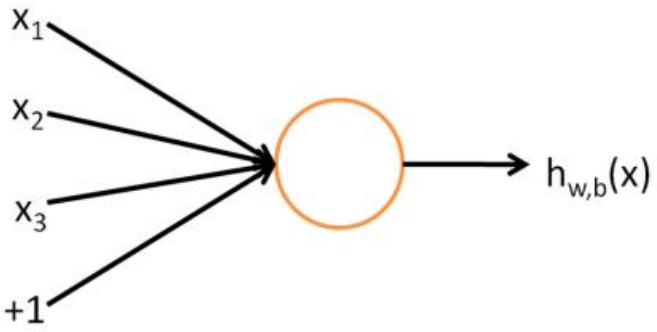


Рисунок 2.2 – Одиночний нейрон з трьома вхідними значеннями, зміщенням та вихідною гіпотезою.

При заданому вхідному сигналу, x, мережа виводить гіпотезу $h(x)_{w,b}$, де w – вага, а b – зміщення, що є заданими, або можуть бути отримані в процесі навчання. Вихідна гіпотеза нейрона рахується наступним чином:

$$h(x)_{w,b} = f\left(\sum_{i=1}^{n} w_i x_i + b\right) = a \tag{2.1}$$

де w_i та x_i позначають вагу зв'язку і вхідні данні до i-го з n входів. За $f: \mathbb{R} \rightarrow \mathbb{R}$ позначено сигмоїдну функцію, що переводить вихідний сигнал у проміжок [0,1]:

$$f(z) = \frac{1}{1 + e^{-z}} \tag{2.2}$$

Шляхом об’єднання багатьох нейронів так, що вихідний сигнал одного нейрона стає вхідним сигналом іншого, утворимо більш складну нейронну мережу. Так, нейрони можуть бути об’єднані у шари, як показано на рис. 2.3.

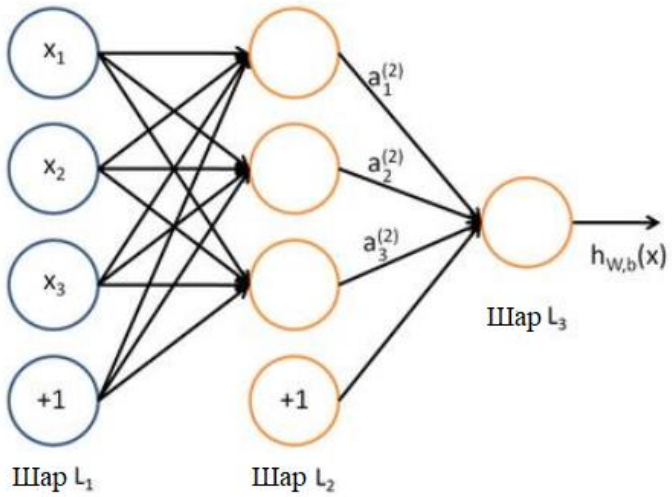


Рисунок 2.3 – Нейронна мережа з трьох шарів, де L_1– вхідний шар, L_2– прихований шар, L_3 – вихідний шар.

Лівий шар мережі називають вхідним шаром. Елемент +1 відповідає зсуву. Ця мережа має три вхідних нейрони (не рахуючи зміщення). Правий шар – це вихідний шар, який має лише 1 вихідний нейрон. Середній шар називається прихованим, оскільки його значення не відображаються на вхідних або вихідних даних. У цій мережі є 3 прихованих нейрони. З'єднання між елементами представляють ваги.

2.7 Модель автоенкодера

Автоенкодер – модель нейронної мережі з прихованими шарами, яка застосовуючи алгоритм навчання без учителя і метод зворотного поширення помилки встановлює цільове значення, рівне вхідному вектору. Автоенкодер з одним прихованим шаром показаний на рис. 2.4.

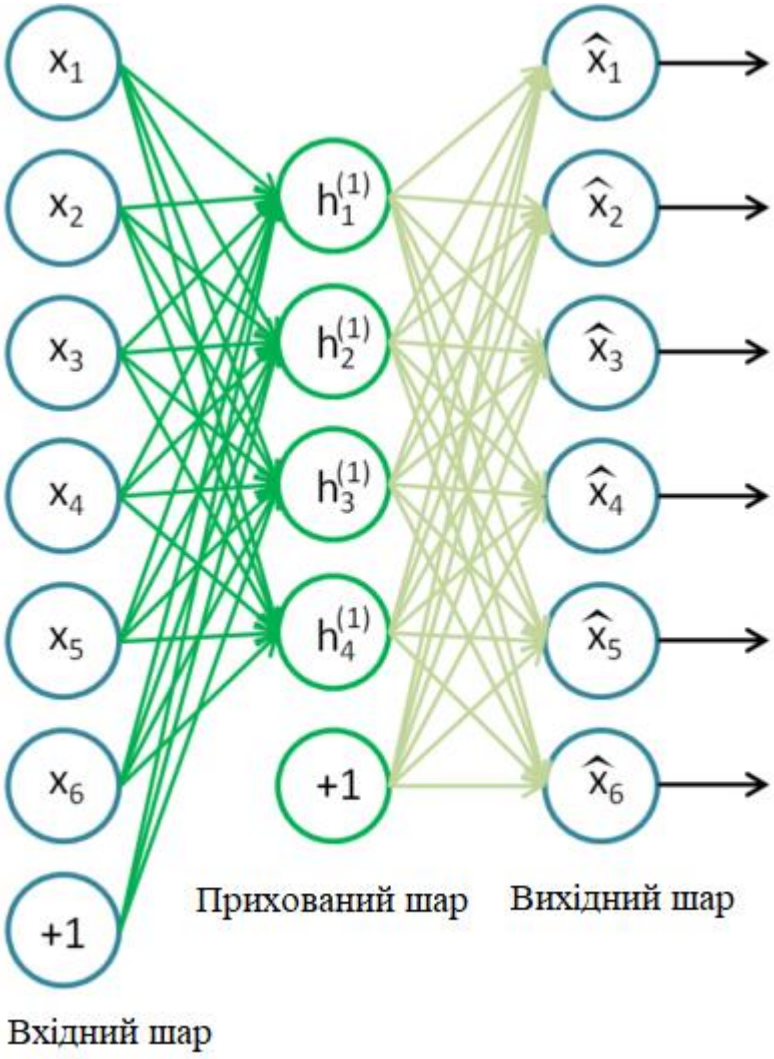


Рисунок 2.4 – Приклад архітектури автоенкодера

Автоенкодер намагається побудувати гіпотезу $h(x)_{w,b} = x$. Іншими словами, розв’язує задачу знаходження таких ваг, щоб вихідні дані нейронної

мережі приблизно дорівнювали значенню вхідних. Для того, щоб рішення цього завдання було нетривіальним, кількість нейронів прихованого шару повинно бути менше, ніж розмірність вхідних даних.

Це дозволяє отримати стиснення даних при передачі вхідного сигналу на вихід мережі. Наприклад, якщо вхідний вектор являє собою набір рівнів яскравості зображення 10x10 пікселів (всього 100 ознак), а кількість нейронів прихованого шару 50, мережу вимушено навчається компресії зображення. Адже вимога $h(x)_{w,b} = x$ означає, що виходячи з рівнів активації п'ятдесяти нейронів прихованого шару вихідний шар повинен відновити 100 пікселів вихідного зображення. Така компресія можлива, якщо в даних є приховані взаємозв'язки, кореляція ознак, і взагалі якась структура.

Пізніше з появою ідеї розрідження набув поширення розріджений автоенкодер. Розріджений автоенкодер – це автоенкодер, у якого кількість прихованих нейронів набагато більше розмірності входу, але вони мають розріджену активацію. Розріджена активація – це коли кількість неактивних нейронів в прихованому шарі значно перевищує кількість активних. Якщо описувати розрідженість неформально, то будемо вважати нейрон активним, коли значення його функції передачі близько до 1. Якщо використовується сигмоїдна функція передачі, то для неактивного нейрона її значення повинно бути близьким до 0.

2.8 Накопичуючі автоасоціативні мережі

Для вилучення з вхідного набору даних абстракцій високого рівня автоенкодери складають в мережу. На рис. 2.5 наведена структурна схема накопичуючого автоенкодера і нейромережі, які в сукупності і представляють собою глибоку нейромережу.

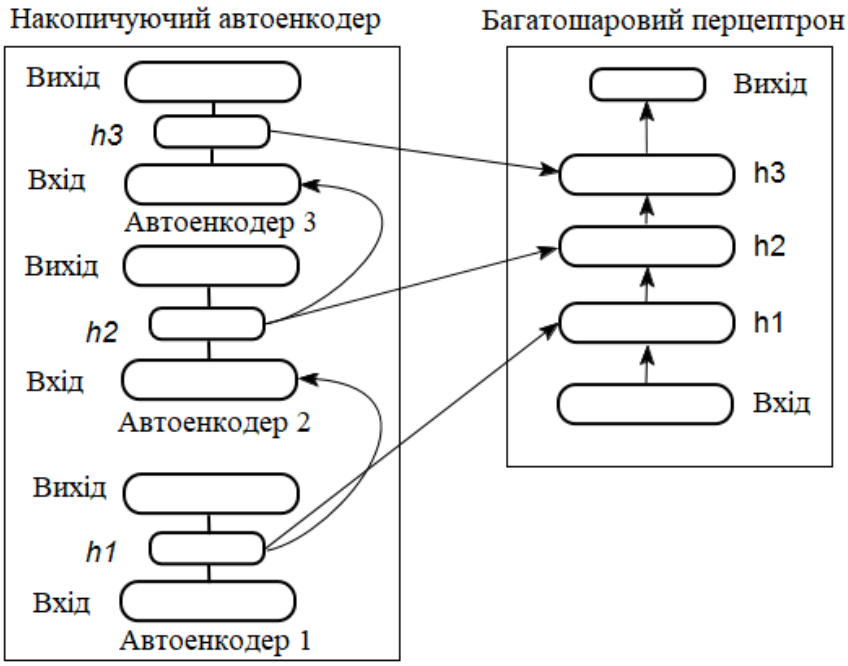


Рисунок 2.5 – Схема накопичуючого автоенкодера

2.9 Навчання автоенкодера

Навчання глибоких мереж проводять в два етапи. На першому етапі пошарово навчають без вчителя на масиву не розмічених даних автоасоціативну мережу, після чого отриманими після навчання вагами прихованих шарів автоасоціативної мережі ініціалізують нейрони прихованих шарів звичайного багатошарового перцептрона. На рис. 2.5 схематично показаний цей процес навчання і перенесення. Після навчання першого автоенкодера ваги нейронів прихованого шару стають входами другого і так далі. Тим самим з даних витягується все більше узагальнююча інформація про їх структуру. Саме через те, що кожен автоенкодер має по одному прихованому шару навчання на першому етапі виконується так швидко.

На другому етапі відбувається тонке налаштування багатошарового перцептрона (навчання з учителем) на розміченому наборі даних загальновідомими методами. Практично доведено, що така ініціалізація

встановлює ваги нейронів прихованих шарів прецептрона в область глобального мінімуму і подальше тонке налаштування відбувається за дуже короткий час.

2.10 Обрана архітектура нейронної мережі

В даній роботі в якості моделі нейронної мережі був обраний саме автоенкодер, через суть його роботи – виокремлення закономірностей з даних, а також можливість навчання у два етапи, що значно скорочує затрачуваний час. А це, в свою чергу, поліпшить задачу оптимізації параметрів вхідних даних та нейромережі.

В якості продовження роботи, можна дослідити складніші моделі мереж, наприклад, згорткових, що отримували б в якості вхідних даних перетворені в зображення часові ряди, в тій чи іншій формі. Або, розглянути можливості розв’язання задачі прогнозування за допомогою нейронних мереж з короткою пам’яттю. Щоправда, їх навчання займе значно більший час.

2.11 Недоліки використання нейронних мереж

Основні труднощі використання нейромереж – так зване "прокляття розмірності". При збільшенні розмірності входів і кількості шарів, складність мережі і відповідно час навчання зростає експоненційно, при цьому отриманий результат може не покращитися.

Інша особливість у використанні нейронних мереж полягає в складності відновлення закономірностей, які мережі знайшли в процесі навчання. Це відбувається через складність декодування взаємозв’язків та характеристик знайдених нейромережею.

2.12 Висновки до розділу 2

Були розглянуті задачі, для яких застосовуються нейронні мережі, описані принципи їх будови ота налаштування. На основі властивостей різних архітектур мереж та проблем, до яких вони застосовуються, в якості обраної архітектури, було обрано накопичуючий автоенкодер. Також було описано алгоритм зворотного поширення похибки, за яким параметри мережі будуть навчатися.

РОЗДІЛ 3 ПІДГОТОВКА ДАНИХ ТА МОДЕЛІ

3.1 Отримання даних

Варто зауважити, що побудовані моделі тестуються на даних за останній рік, через необхідність встановлення реальних рівнів ефективності прогнозування. Методи торгівлі на ринках змінюються щороку, через впровадження новіших технологічних рішень. Так, тестування на даних отриманих більше року тому може бути нерелевантним.

Дослідимо обрані часові ряди, щоб зрозуміти їх характер, особливості, природу. В подальшому завдяки цьому аналізу можна буде зробити припущення, чому деякі моделі прогнозування працюють чи ні.

Отже, будемо розглядати часові ряди на часовому проміжку в один рік (20.03.17 - 20.03.18). Графіки для аналізу будуються по наступних характеристиках: ціна відкриття проміжку, максимальна ціна за проміжок, мінімальна ціна за проміжок, ціна закриття проміжку Обрані часові ряди для дослідження:

- Валюти

Євро - Долар

Британський Фунт - Долар

Японська Йена - Долар

Долар – Біткойн

- Сировина

Нафта

Срібло

Золото

- Акції

Apple

Amazon

Facebook

- Економічні індекси

STOXX Europe 50 index

S&P 500 index

3.2 Аналіз завантажених даних

Опишемо схему підготовки даних на прикладі валютної пари Євро/Долар (EUR/USD) на годинному часовому ряді. Для інших часових рядів при різних часових інтервалах процес аналогічний. Ці дані у формі csv-файлу завантажуємо з сайту Dukascopy. Дані мають значення за 9000 періодів, з 5 значеннями – часу, ціни входу, виходу, найбільшою та найменшою.

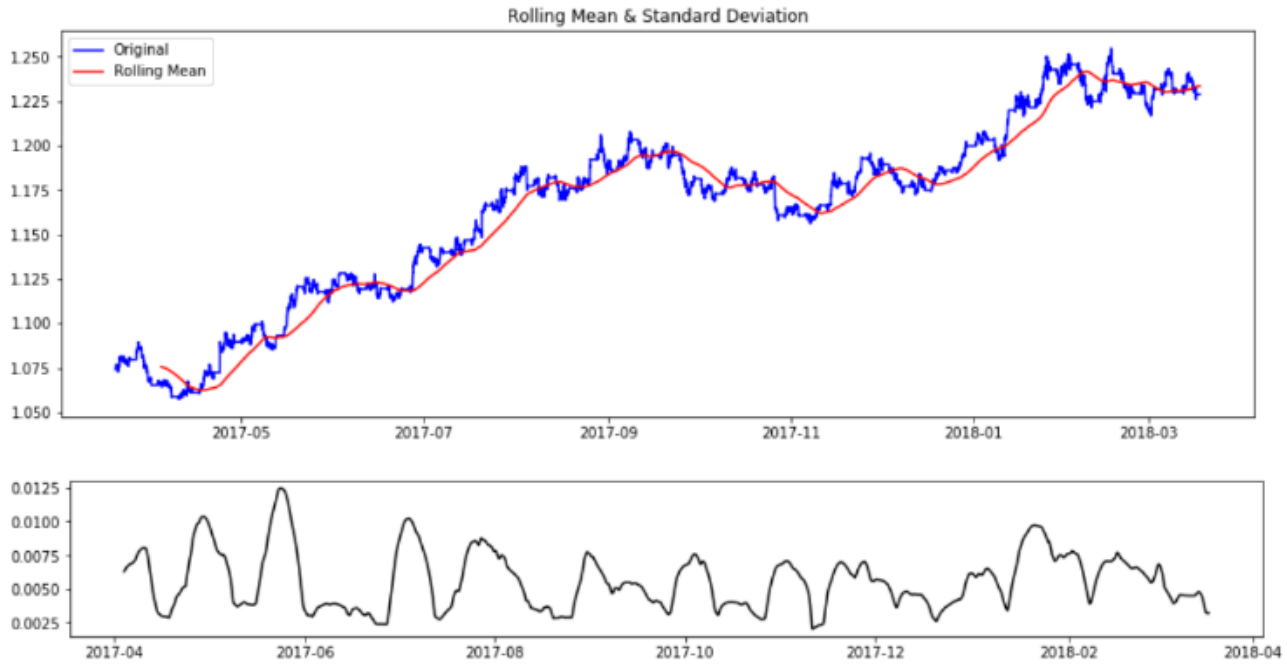


Рисунок 3.1 – Завантажені дані Євро – Долар з ковзним середнім та відносним стандартним відхиленням.

Розглянемо завантажені дані. Очевидно, що такі часові ряди не є стаціонарними. Кожен часовий ряд дослідимо їх на предмет тренду, намалюємо

просте ковзне середнє за останні 365 значень (формула 3.1), та відносне стандартне відхилення (формула 3.3), що рахується як стандартне відхилення розділене на просте ковзне середнє. Приведемо формули для обрахунку ковзного середнього та відносного стандартного відхилення.

$$p_{MA} = \frac{p_L + p_{L-1} + \cdots + p_{L-(n-1)}}{n}, \tag{3.1}$$

де p_{MA} – значення ковзного середнього на кроку L, p_L– значення часового ряду на кроку L, n – кількість кроків, за якими рахується ковзне середнє.

$$D_n = \frac{\sum_{j=1}^{n}(p_j - p_{MA})^2}{n-1}, \tag{3.2}$$

$$S_{rlt} = \frac{\sqrt{D_n}}{p_{MA}}, \tag{3.3}$$

де D_n – дисперсія, обрахована за останні n значень, S_{rlt} – відносне стандартне відхилення.

Результат зображений на рис. 3.1. Можемо спостерігати, що у обраного часового ряду явно виражений зростаючий тренд та змінне відносне стандартне відхилення. Явна сезонність також відсутня.

3.3 Створення вхідних даних та їх розмітка

На вхід нейронної мережі необхідно подавати вектор даних, щоб отримати вихідний вектор. Але використовувати тільки значення ціни недостатньо, адже мережі буде досить складно визначити яку-небудь закономірність. Так, будемо використовувати набір індикаторів у якості вхідних даних. Індикатором називається певна функція від попередньої інформації про ціну. Проте

індикаторів існує велика кількість, а використовувати вектори великої розмірності для нейромережі може легко призвести до її перенавчання, що призведе до поганих результатів передбачення.

Вектор вхідних індикаторів буде створений за допомогою аналізу значущості кожного з них. Також необхідно розмітити данні, щоб зробити можливим другий етап навчання нейронної мережі, а також розуміти якість прогнозів.

3.4 Розмітка даних

Для того, щоб зробити можливим другий етап навчання нейронної мережі, а також розуміти якість прогнозів, виконаємо розмітку даних. Так, треба для кожного запису визначити, чи є він рухом вгору чи вниз. Якщо судити за різницею цін відкриття та закриття, буде доволі багато шуму, а прогноз буде малоінформативним, бо модель не зможе прогнозувати напрямок тренду.

Для нашої задачі підходить індикатор ZigZag, що фактично вказує, до якого тренду відносяться дані у минулому та на якому інтервалі. Цей індикатор фільтрує зміни у часовому ряді, що менше за заданий параметр a в процентах. Наприклад, при $a = 10$, якщо зміна ціни становила лише 7%, то це не буде зараховано як зміна тренду. В нашому випадку зручніше, щоб індикатор приймав в якості параметру число у пунктах. Позначимо його функцією ZZ.

На рис. 3.2. зображений індикатор ZigZag, побудований по середнім цінам за період, з параметрами 25 та 50, накладений на останні 500 значень часового ряду.

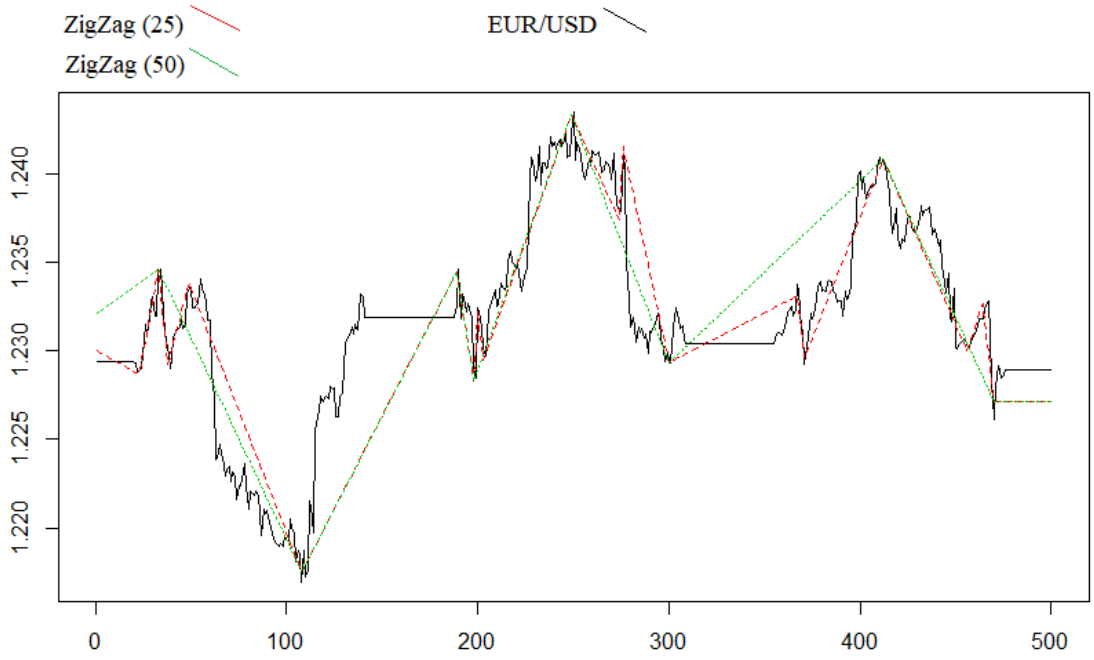


Рисунок 3.2 – Індикатор ZigZag за параметрами 25 та 50, накладений на часовий ряд.

Наші дані розділені на 2 класи – тренд «вгору» та «вниз». Тепер, маючи розмічені дані, можна балансувати класи. Це потрібно для того, щоб модель мала кращі результати навчання. Так, створимо функцію Balancing, що буде зменшувати вибірку, аж поки різниця в класах буде меншою за 15%. В розглянутому часовому ряді баланс класів становив 4545 екземплярів «вгору» та 4136 екземплярів «вниз». Так, можемо вважати класи балансованими.

3.5 Використані індикатори

Для того щоб мати на вході нейронної мережі більш інформативні дані, використовуємо наступний індикаторів, з якого оберемо нам потрібні. Зауважимо, що використовується бібліотека за реалізованими індикаторами TTR. В обраному наборі найпоширеніші індикатори:

- Welles Wilder's Directional Movement Index - ADX(HLC, n) - 4 виводи
- aroon(HL, n) – 1 вивід

- Commodity Channel Index - CCI(HLC, n) – 1 вивід
- Chaikin Volatility - chaikinVolatility (HLC, n) – 1 вивід
- Chande Momentum Oscillator - CMO(Med, n) – 1 вивід
- MACD oscillator - MACD(Med, nFast, nSlow, nSig)
- OsMA(Med,nFast, nSlow, nSig) – 1 вивід
- Relative Strength Index - RSI(Med,n) – 1 вивід
- Stochastic Oscillator - stoch(HLC, nFastK=14, nFastD=3, nSlowD=3) – 3 виводи
- Stochastic Momentum Index - SMI(HLC, n = 13, nFast = 2, nSlow = 25, nSig = 9) – 2 виводи
- Volatility (по Yang and Zhang) - volatility(OHLC, n, calc="yang.zhang", N=96) – 1 вивід

На рис. 3.3 – 3.13 представлені графіки індикаторів на останніх 400 значеннях годинного часового ряду Євро/Долар.

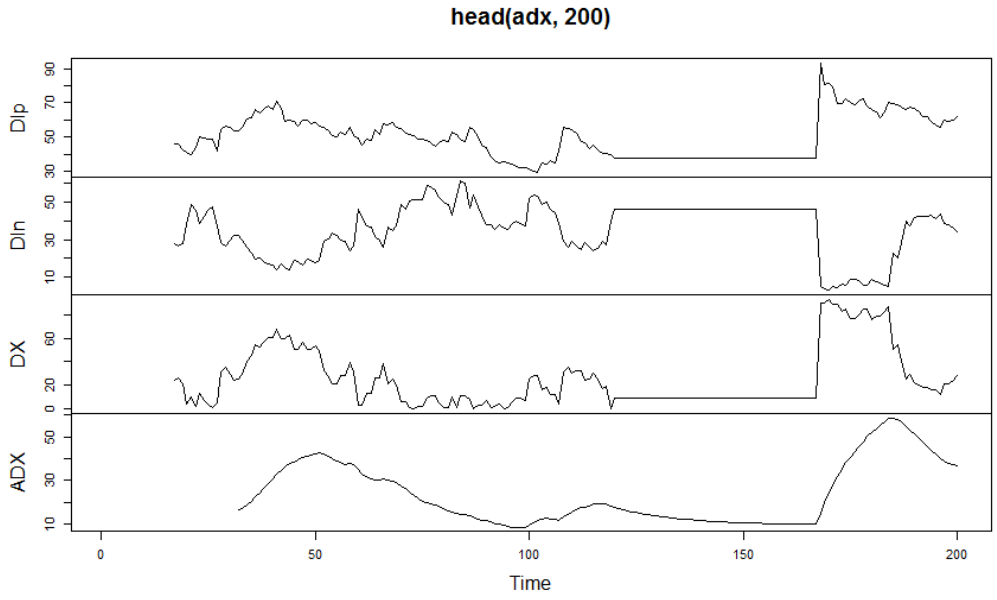


Рисунок 3.3 – Індикатор ADX за останні 200 значень.

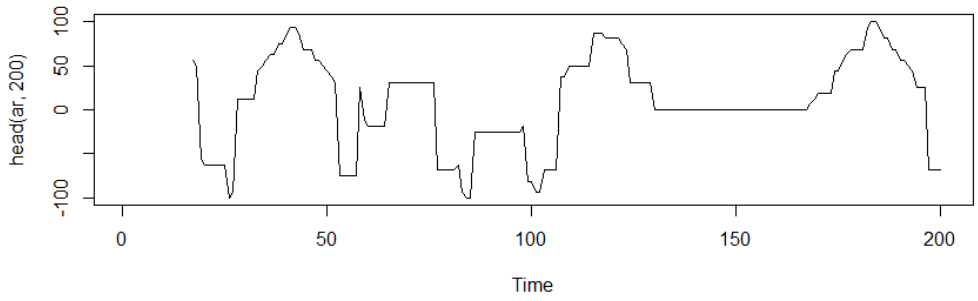


Рисунок 3.4 – Індикатор Aroon за останні 200 значень.

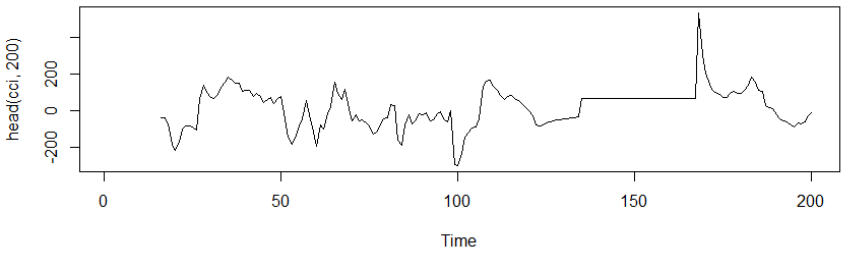


Рисунок 3.5 – Індикатор CCI за останні 200 значень.

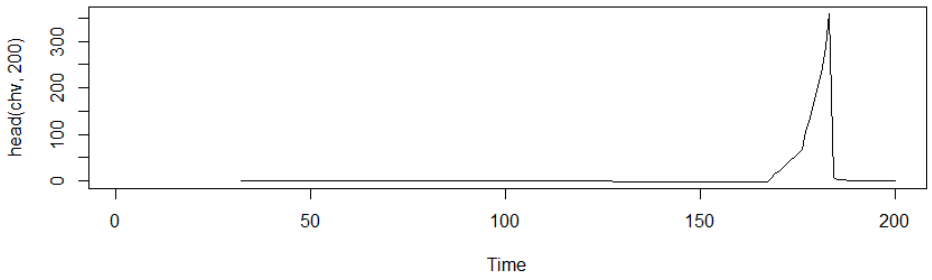


Рисунок 3.6 – Індикатор ADX за останні 200 значень.

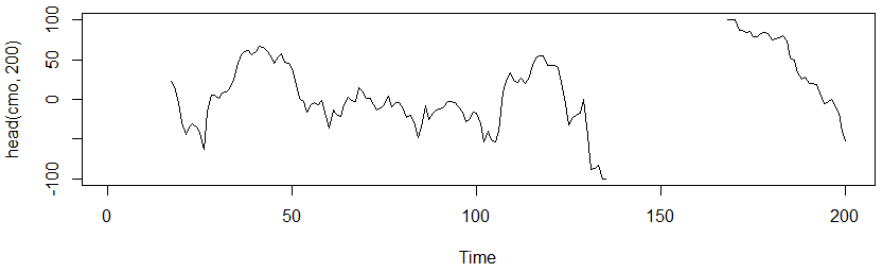


Рисунок 3.7 – Індикатор ADX за останні 200 значень.

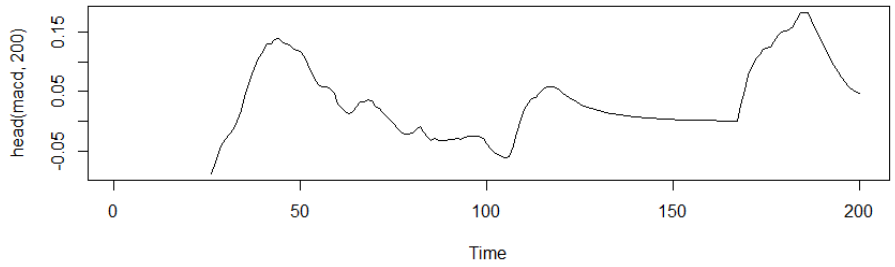


Рисунок 3.8 – Індикатор ADX за останні 200 значень.

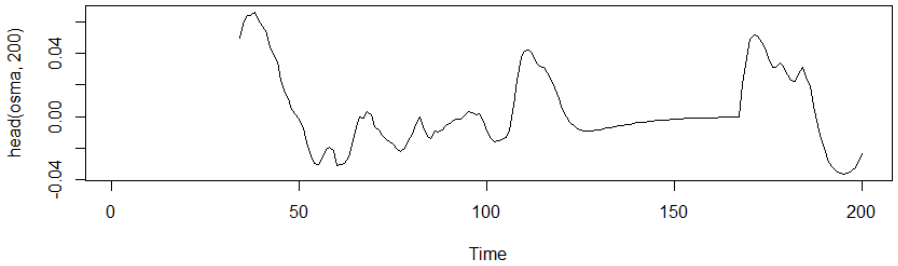


Рисунок 3.9 – Індикатор ADX за останні 200 значень.

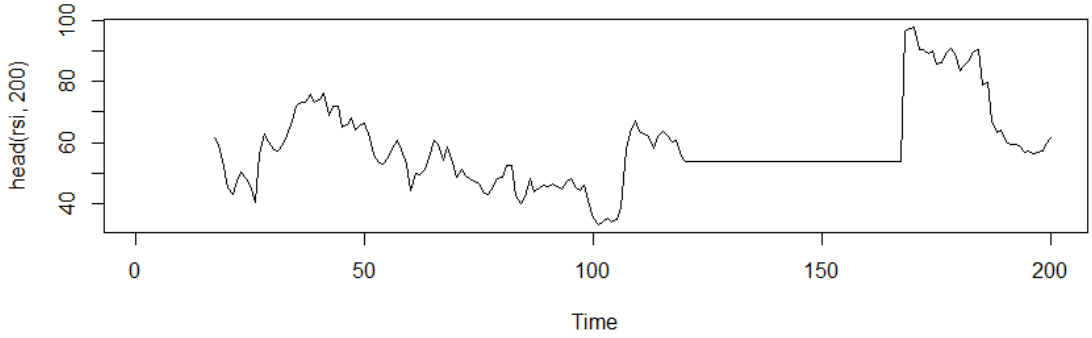


Рисунок 3.10 – Індикатор ADX за останні 200 значень.

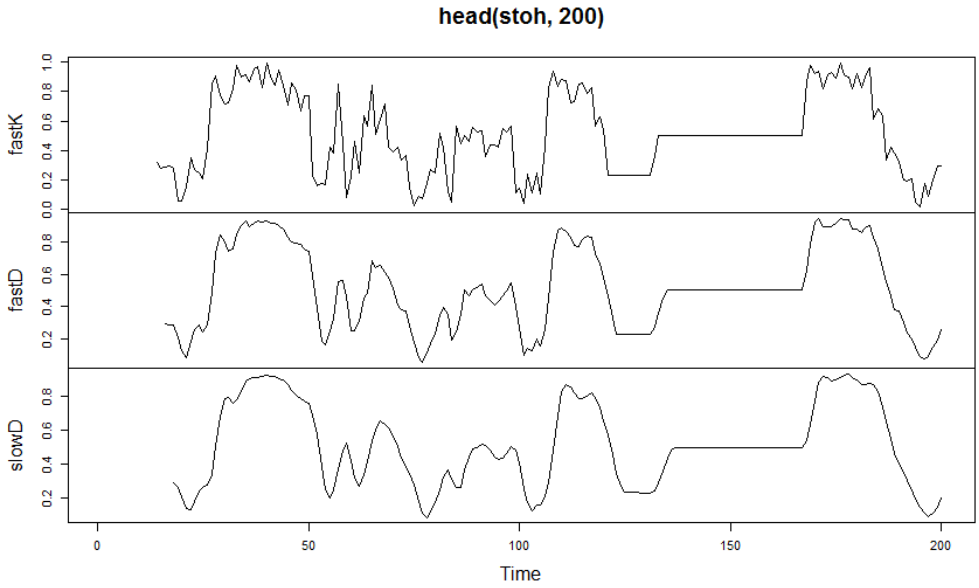


Рисунок 3.11 – Індикатор ADX за останні 200 значень.

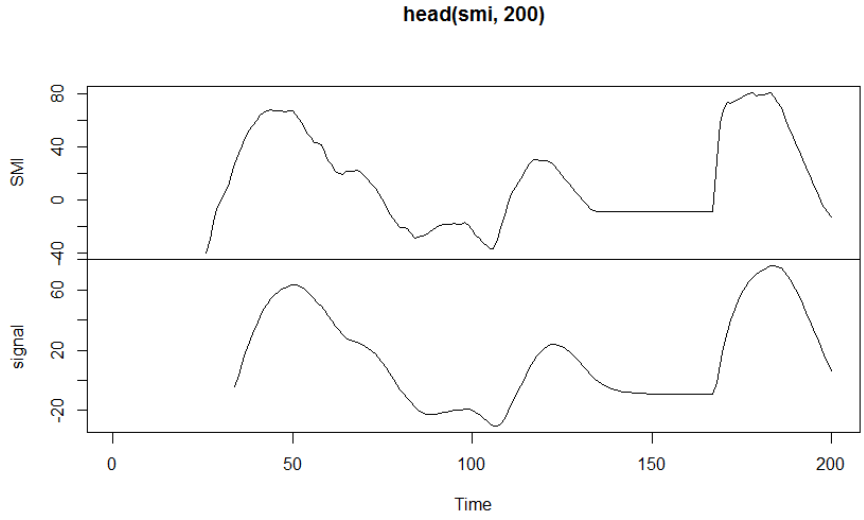


Рисунок 3.12 – Індикатор ADX за останні 200 значень.

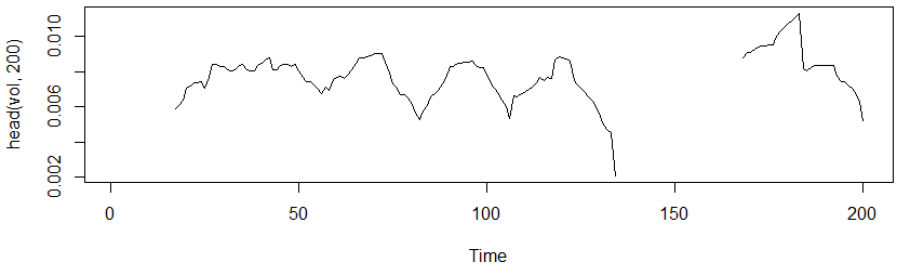


Рисунок 3.13 – Індикатор ADX за останні 200 значень.

3.6 Перетворення даних

Частину даних буде використано для тренування нейронної мережі, а частину для оцінки якості прогнозу моделі. Існує кілька способів розбиття вихідних даних на навчальну і тестову вибірки. Ми застосуємо звичайний поділ вихідних даних на тренувальну і тестову вибірки в співвідношенні 80% до 20%.

Також, варто привести вхідні данні до діапазону [-1, 1]. Так нейронна мережа зможе сама визначити пріоритети в значущості характеристик, незважаючи на початкові розходження в порядках вхідних даних.

3.7 Побудова моделей та їх випробування

Побудуємо і навчимо модель тришарового автоенкодера. Модель приймає на вхід наступні параметри:

- *x* – матриця вхідних даних;
- *y* – вектор цільових змінних;
- *hidden* – вектор з числом нейронів в кожному прихованому шарі, беремо рівним 10;
- *activationfun* – функція активації прихованих нейронів. Може бути "sigm", "linear", "tanh". За замовчуванням "sigm";
- *learningrate* – рівень навчання для градієнтного спуску. За замовчуванням = 0.8;
- *momentum* – момент для градієнтного спуску. За замовчуванням = 0.5;
- *learningrate_scale* – рівень навчання може бути помножений на цю величину після кожної ітерації. За замовчуванням = 1.0;
- *numepochs* – кількість ітерацій для навчання. За замовчуванням = 3;
- *batchsize* – розмір вибірок на яких проводиться навчання. За замовчуванням = 100;

- *output* – функція активації для вихідних нейронів, може бути "sigm", "linear", "softmax". За замовчуванням "sigm";
- *sae_output* – функція активації вихідних нейронів SAE, може бути "sigm", "linear", "softmax". За замовчуванням "linear";
- *hidden_dropout* – частина прихованих шарів, що видаляється. За замовчуванням = 0;
- *visible_dropout* – частина видимих шарів, що видаляється. За замовчуванням = 0.

Побудуємо модель розміром (17, 100, 100, 100, 1), навчимо її у два етапи, оцінимо час навчання, і точність передбачень. Спочатку пошарово тренуються автоенкодер, після чого тренується кінцева нейромережа. Параметрами встановлено невелику кількість епох навчання і велику кількість прихованих нейронів в трьох шарах. Весь процес навчання зайняв трохи більше 10 секунд.

На годинному часовому ряді Євро/Долар точність передбачення становила 74%. Це доволі непоганий результат. Але можна ввести іншу метрику продуктивності прогнозування – обрахувати можливий прибуток, якщо здійснювати операції по купівлі і продажу залежно від отриманих сигналів. Для цього використаємо функцію кумулятивної суми. Подивимось на результати використання наших передбачень на останніх 500 значеннях і отримаємо криву балансу, що зображена на рис. 3.14. Так, за тиждень використання такої стратегії баланс збільшився би на 0,20%.

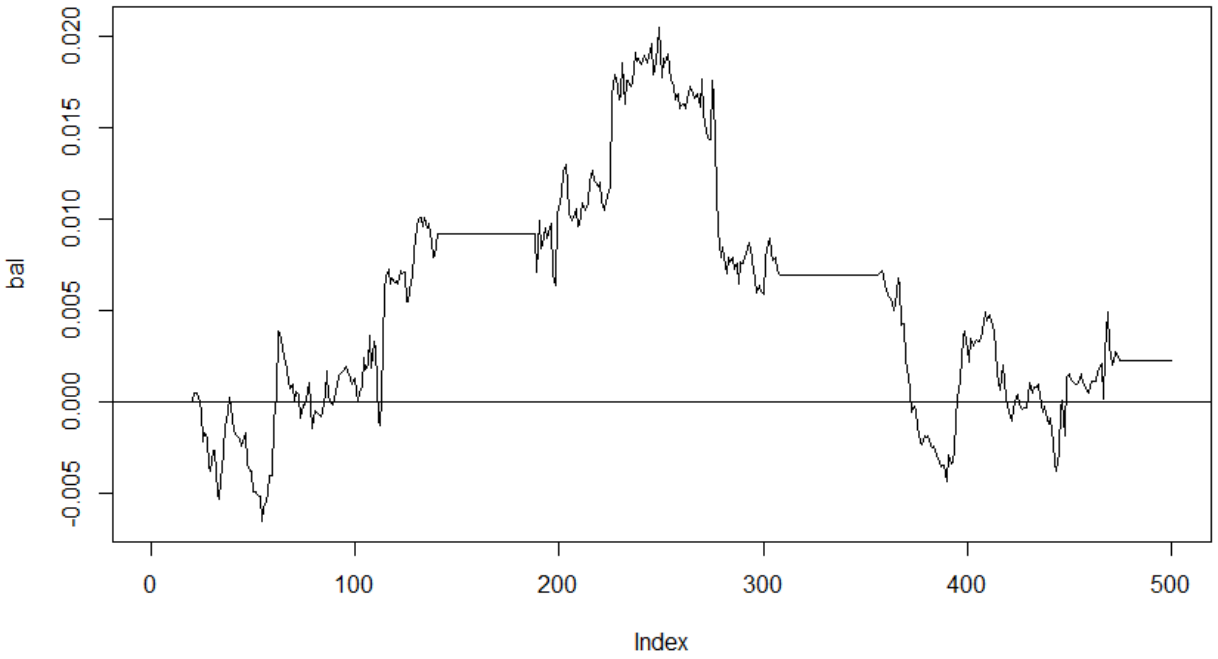


Рисунок 3.14 – Крива балансу тестування отриманої моделі нейромережі для валюти Євро/Долар.

Також отриманий результат балансу можна порівняти з ідеальною в такому разі стратегією – наперед відомими значеннями індикатора ZigZag. На рис. 3.15 зображений графік з одночасно накладеними графіками балансу результатів нейромережі в порівнянні з оптимальною стратегією. При її використанні баланс збільшився би на 0,85%. Зазначимо, що застосування такої стратегії потребує знання майбутніх даних, що використовуються при обрахунку значень індикатора.

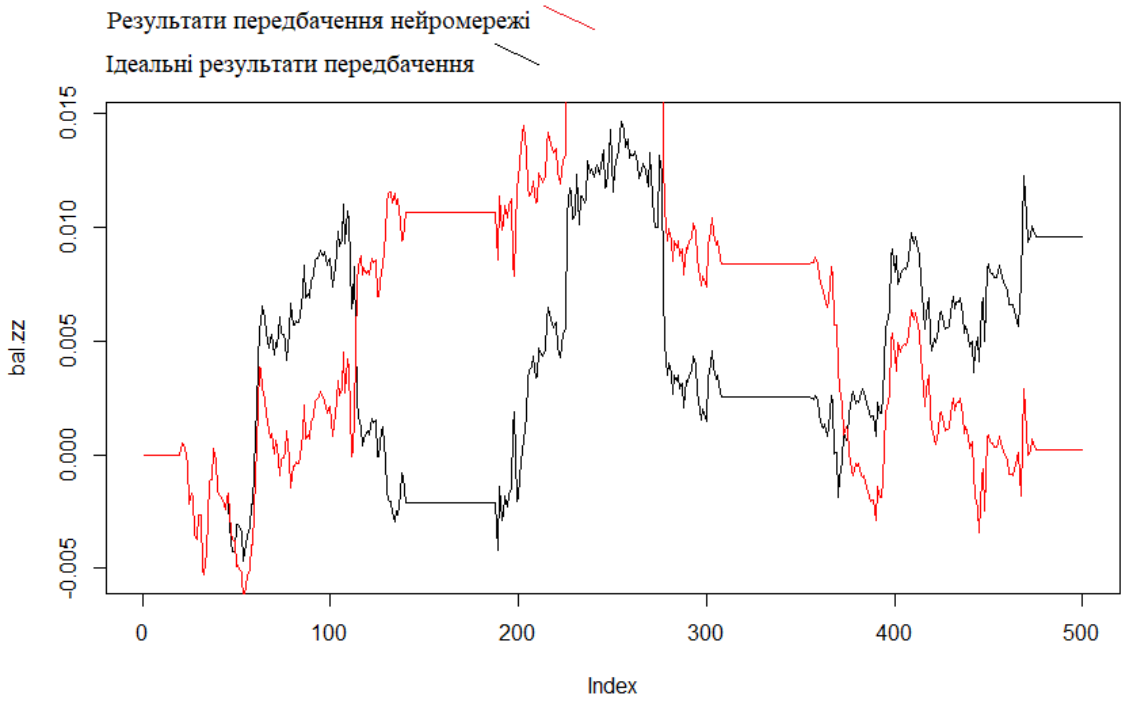


Рисунок 3.15 – Порівняння стратегій нейомережі та ідеальної можливої на часовому ряді Євро/Долар.

3.8 Оптимізація параметрів моделі

Результати, з попередього пункту, отримані нейромережею, що була побудована за довільно визначеною архітектурою. Задачу оптимізації параметрів мережі можна вирішувати за допомогою генетичних алгоритмів, адже маємо простір великої розмірності, а деякі за параметрів приймають тільки цілі значення.

3.9 Висновки до розділу 3

Завантажені фінансові дані за допомогою індикаторів були підготовлені для надання моделі нейронної мережі. Була побудована сама модель та отримані прогнози для всіх обраних фінансових процесів.

РОЗДІЛ 4 АНАЛІЗ РОБОТИ ПРОГРАМНОГО ПРОДУКТУ

4.1 Результат роботи програми

В попередньому розділі розглядалися результати роботи програми тільки на одному часовому ряду – годинному Євро/Долар. В таблиці 4.1 наведено результати прогнозування для інших годинних часових рядів, а також їх порівняння з ідеальними результатами.

Таблиця 4.1 Результати прогнозування напрямку руху моделі нейронної мережі.

Часовий ряд	Точність прогнозування	Результат отриманої стратегії	Результат ідеальної стратегії
Євро - Долар	76%	0,20%	0,85%
Британський Фунт - Долар	42%	-2,07%	4,02%
Японська Йена - Долар	48%	0,63%	3,22%
Долар - Біткойн	42%	-3,38%	1,77%
Нафта	61%	1,05%	-2,38%
Срібло	64%	-0,42%	-0,46%
Золото	64%	-2,05%	-0,9%
Apple	71%	0,06%	1,32%
Amazon	77%	0,51%	0,99%
Facebook	54%	-0,07%	1,31%
STOXX Europe 50	67%	1,6%	0,8%
S&P 500	45%	-0,3%	0,07%

Розглянувши результати, можна зробити висновок, що така модель нейронної мережі з заданими даними має найкращу точність прогнозування акцій та природних ресурсів. Можливо, через характер прихованих закономірностей.

Розглянувши результати, можна зробити висновок, що така модель нейронної мережі з заданими даними має найкращу точність прогнозування акцій та природних ресурсів. Можливо, через характер прихованих закономірностей.

4.2 Порівняння отриманих результатів

Для розв'язання задачі прогнозування розвитку часових рядів загальноприйнято використовувати авторегресійну інтегровану модель з ковзним середнім (APIMA). Модель побудовано та тестовано на мові програмування Python з використанням бібліотеки statsmodels.

Пороілюструємо процес побудови моделі на прикладі часового ряду Євро/Долар. З рис. 3.1 видно, що в даного часового ряду є загальний зростаючий тренд. Через це візьмемо різницю часового ряду порядку 1. Далі, розглянемо автокореляцію лагів обраного ряду, що зображена на рис. 4.1.

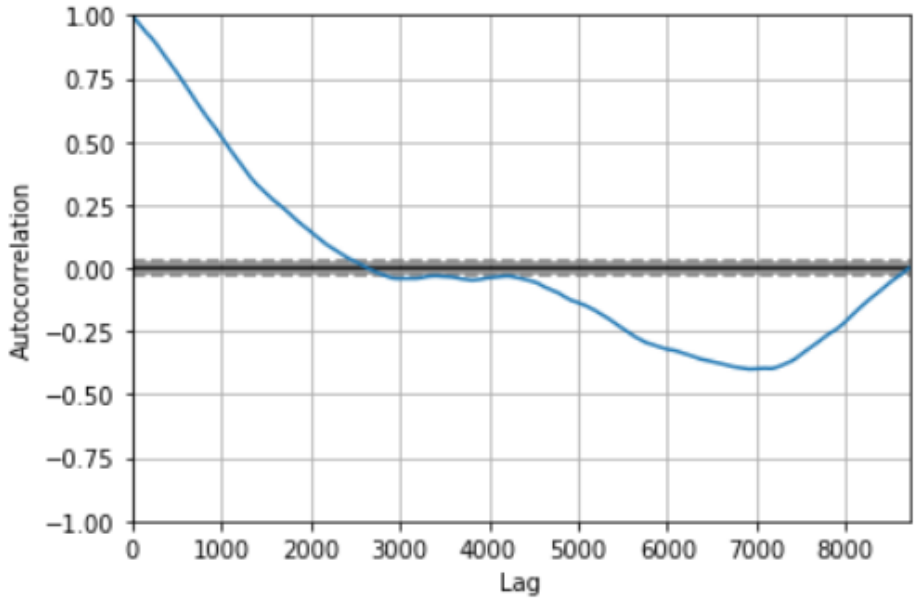


Рисунок 4.1 – Автокореляція лагів у часовому ряді Євро/Долар.

Наявна кореляція для перших 1000 лагів. Побудуємо моделі з різними параметрами *p* до 15 та *d = 1*. З огляду на результати, найменша помилка в моделі APIKC(1,1,0).

Далі, розділивши дані таким же чином на тренувальні і тестові, що і в моделі на основі нейронної мережі, будемо прогнозувати значення часового ряду на один крок вперед. Для порівняння оцінок методів, перетворимо виводи моделі АРІКС до двох класів – «зростаюча ціна» та «спадаюча ціна». Так, зможемо сказати, з якою точністю АРІКС модель здатна кластеризувати дані за трендом, розміченим індикатором ZigZag. Результати експериментів нваедено в таблиці 4.2.

Таблиця 4.2 Результати прогнозування напрямку руху моделі АРІКС.

Часовий ряд	Точність прогнозування	Результат отриманої стратегії	Результат ідеальної стратегії
Євро - Долар	52%	-	-
Британський Фунт - Долар	54%	-	-
Японська Йена - Долар	53%	-	-
Долар - Біткойн	45%	-	-
Нафта	52%	-	-
Срібло	60%	-	-
Золото	57%	-	-
Apple	60%	-	-
Amazon	61%	-	-
Facebook	62%	-	-
STOXX Europe 50	59%	-	-
S&P 500	45%	-	-

З огляду на результати можна зробити висновок, що запропонована модель показує кращі результати в задачі класифікації тренду. Проте, її порівняння з регресійною моделлю є дещо неточним з огляду на різний характер прогнозів.

Обрана модель показує найкращі результати в задачах прогнозу цін на акції (середня точність 70% у розглянутій групі), дещо гірше ціни на природні ресурси (середня точність 63% у розглянутій групі). Прогнози фінансових індексів (середня точність 55% у розглянутій групі) та валют (середня точність 57% у розглянутій групі) не мають бажаного рівня точності.

4.3 Огляд розробленого програмного продукту

В ході написання роботи був розроблений крос платформний модуль на мові програмування R. Основний лістинг модуля наведений у додатку Б. Приведемо структуру модуля та функції, що були створені.

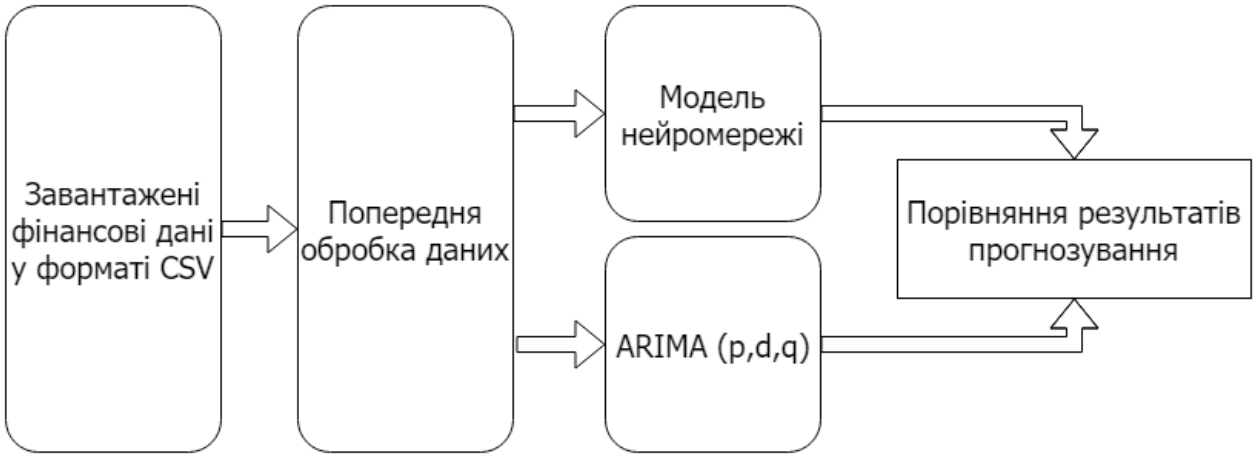


Рисунок 4.2 – Процес аналізу даних та результатів прогнозів моделей.

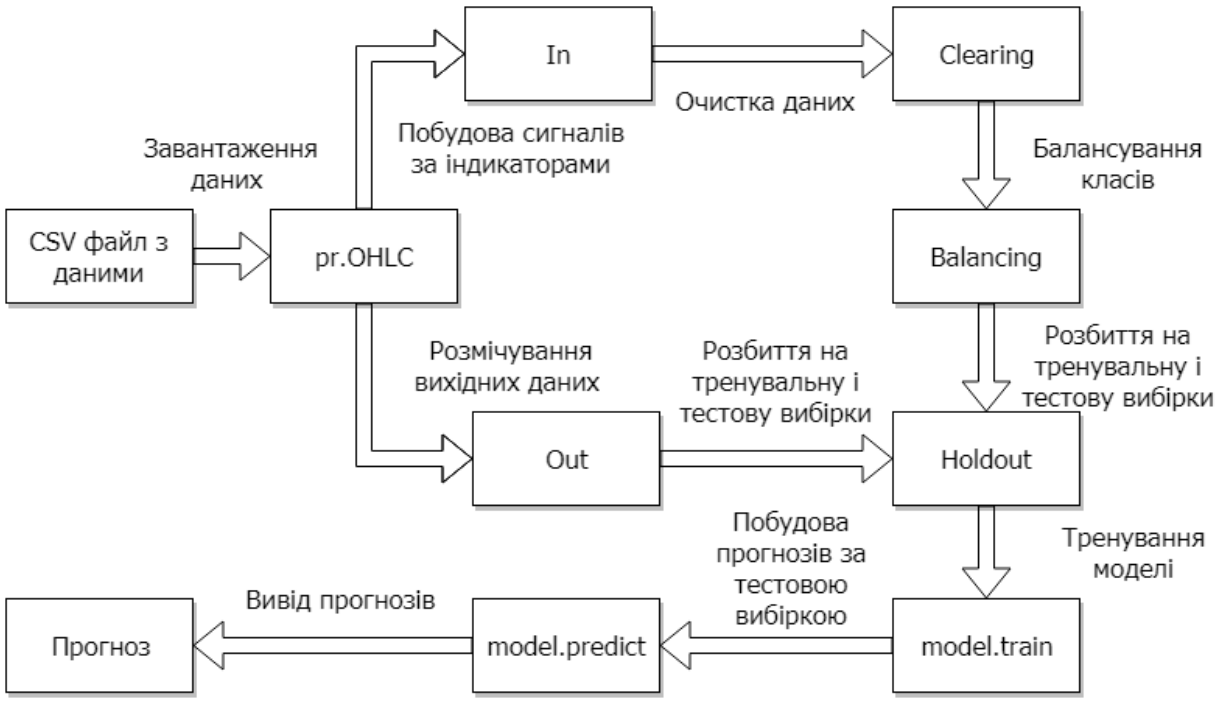


Рисунок 4.3 – Структура програмного модулю.

В побудованому програмному модулі є 8 основних функцій. Приведемо їх більш детальний опис:

- pr.OHLC – відповідає за початкове завантаження даних у модель.
- In – виконує обрахунок значень всіх індикаторів над завантаженими даними, записує результати до масиву.
- Out – розмічає завантажені дані, тобто застосовує до них індикатор ZigZag та присвоює записам значення 1 чи 0 в залежності від встановленого напрямку руху.
- Clearing – очищує дані від нульових значень, якщо такі були в завантажених даних.
- Balancing – балансує дані так, щоб розбіжність між класами отриманими з функції Out складала менше 15%.
- Holdout – поділяє остаточні дані на дві вибірки – тестову і тренувальну.
- model.train – виконує побудову нейромережі та її навчання на тренувальних даних.

- model.predict – будує прогнозування даних з тестової вибірки, оцінює точність отриманих результатів.

Створений програмний модуль може бути включений в склад програми більшого масштабу як допоміжний, або обгорнутий в графічну оболонку для простоти використання. Вигляд інтерфейсу для користування модулем через програму R Studio наведений на рис. 4.4.

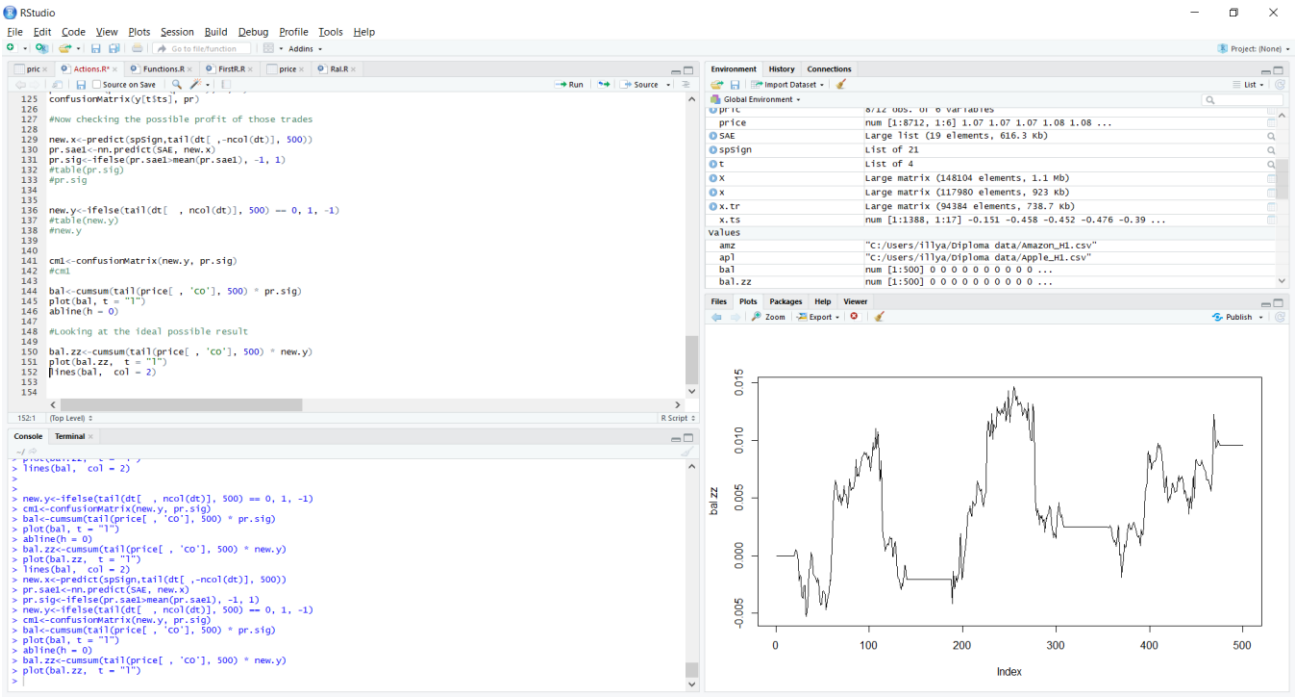


Рисунок 4.4 – Інтерфейс роботи з модулем у програмі R Studio.

4.4 Висновки до розділу 4

Отримані результати прогнозування за допомогою побудованої моделі були проаналізовані та порівняні з результатами стохастичної моделі АРІКС. Виявилося, що модель показує найкращу точність прогнозу на фінансових процесах типу акцій, а найгіршу на валютних парах. Побудована модель показала кращу точність за звичайну АРІКС. Точність прогнозів для деяких процесів сягає 75%, що, з огляду на їх характер, можна вважати гарною.

РОЗДІЛ 5 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

5.1 Вступна частина

У даному розділі проводиться оцінка основних характеристик програмного продукту, що містить реалізацію моделі штучної мережі для прогнозування розвитку фінансових процесів у формі часових рядів, а також тестові програми для знаходження і обробки даних для завантаження в модель нейромережі. Далі, в цій частині проводиться аналіз варіантів реалізації моделі, з метою вибору найбільш оптимальної з економічної точки зору, а також з огляду на продуктивність програми, сумісність з апаратним забезпеченням та простоту використання. Для досягнення цієї мети був використаний апарат функціонально-вартісного аналізу (ФВА).

Програмна реалізація моделі була розроблена на мові програмування R з використанням бібліотеки deepnet. Кінцевий продукт призначений для використання на персональних комп’ютерах під управлінням таких операційних систем як Windows, Linux, MacOS , після встановлення необхідних бібліотек.

Функціонально-вартісний аналіз — технологія, створена для оцінки реальної вартості розробленого продукту або послуги без огляду на структуру компанії, для якої був розроблений. Її суть полягає в тому, щоб розподілити трати – прямі та додаткові – в залежності від об’ємів ресурсів, потрібних на кожному етапі виробництва. Функціями в цьому методі називаються дії, виконані на цих етапах.

ФВА має на меті забезпечення оптимального розподілу ресурсів, що виділяються на створення продукції та послуг між прямими і непрямими витратами. У разі розробки програмного продукту, це виявлення усіх витрат на його реалізацію.

Опишемо алгоритм ФВА:

- Формується послідовність функцій, що потрібні для виробництва продукту. Всі вони розподіляються по двох групах – одні впливають на вартість продукту, інші – ні.
- Кожній функції співставляютья повні річні витрати ті кількість робочих годин.
- Кожній функції визначається джерело витрат.
- Проведення кінцевого розрахунку для визначення загальної вартості виробництва продукту.

5.2 Задача техніко-економічного аналізу

Через застосування методу ФВА для техніко-економічного аналізу розробки, введемо систему критеріїв якості програмного продукту. Поставимо наступні вимоги до продукту:

- Програмний продукт має працювати на широкому класі персональних комп’ютерів.

- Забезпечити простоту у використанні як у вигляді окремої програми, так і програмного модуля.

- Передбачити малі витрати на встановлення програмного продукту.

5.3 Обґрунтування функцій програмного продукту

5.3.1 Формування варіантів функцій

Головна функція F0 – розробка програмного продукту, який містить в собі реалізації основних структур і компонентів, якими оперують моделі нейронних мереж, створення обраних архітектур моделей для аналізу та прогнозування часових рядів, розробка тестового програмного забезпечення, відповідального

за запуск алгоритмів, їх порівняння та якісний аналіз показників роботи. Виходячи з конкретної мети, можна виділити наступні основні функції програмного продукту:

F1 – вибір цільової платформи програмного продукту;

F2 – вибір мови програмування;

F3 – вибір бібліотеки для створення нейронних мереж.

Кожна з основних функцій може мати декілька варіантів реалізації.

Функція F1:

а) крос-платформний додаток;

б) додаток Linux;

в) додаток Windows;

г) додаток Android;

д) скрипт Matlab;

Функція F2:

а) мова програмування Python;

б) мова програмування C++;

в) мова програмування R;

г) мова програмування Java;

д) мова програмування Matlab;

Функція F3:

а) Theano;

б) Torch;

в) Caffe;

г) TebsorFlow;

д) Keras;

е) deepnet;

з) Matlab;

5.3.2 Варіанти реалізації основних функцій

Варіанти реалізації основних функцій наведено у морфологічній карті системи (рис. 5.1). Далі, на основі цієї карти побудовано позитивно-негатвину матрицю варіантів основних функцій (таблиця 5.1). Морфологічна карта відображає всі можливі комбінації варіантів реалізації функцій, які складають повну множину варіантів програмного продукту. Враховуючи характеристики зазначені в позитивно-негативній таблиці, можемо обрати оптимальний варіант функції розроблюваного продукту. При чому таке рішення буде обґрунтованим.

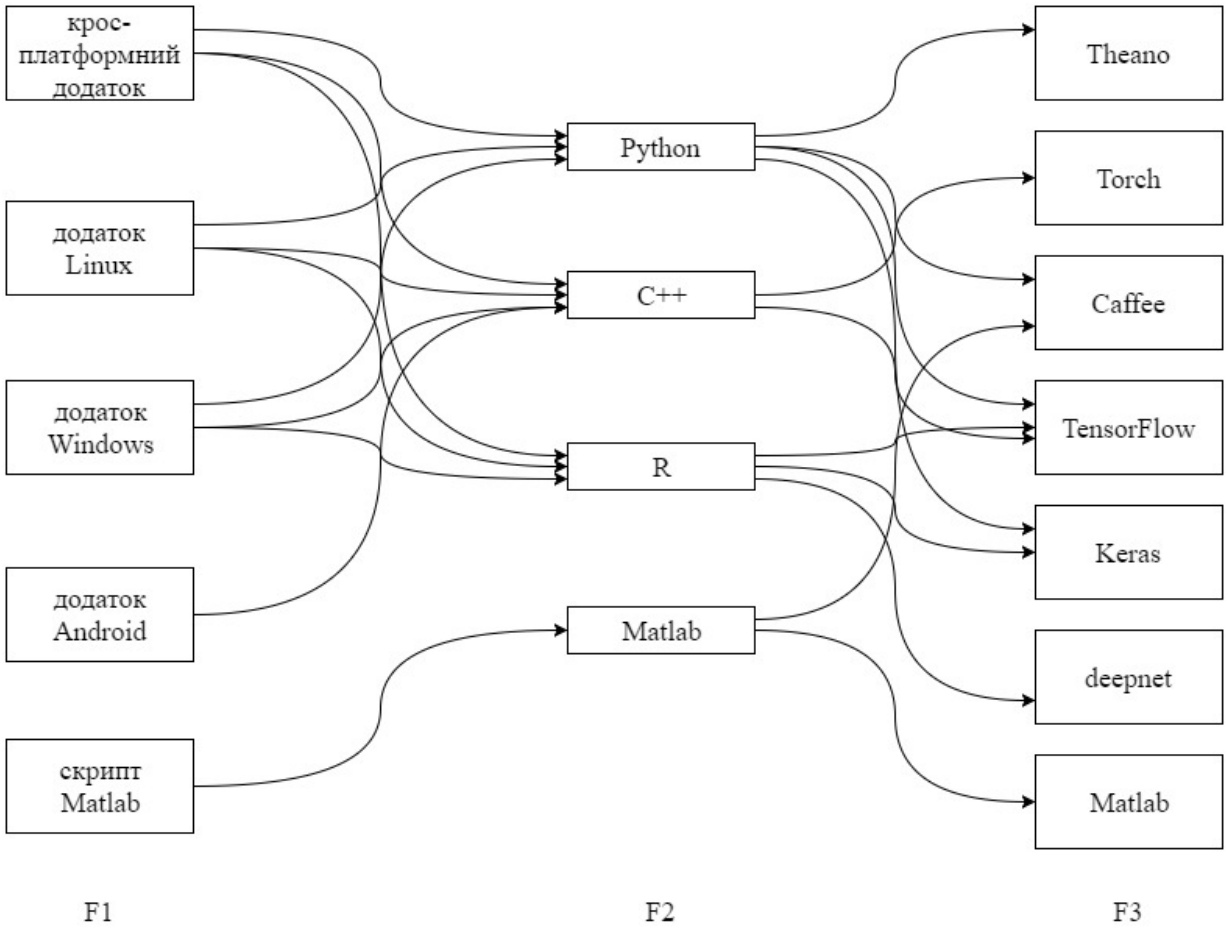


Рисунок 5.1 – Морфологічна карта

Таблиця 5.1- Позитивно-негативна матриця

Основні функції	Варіанти реалізації	Переваги	Недоліки
F1	*А*	Кросплатформність	Обмеженість інструментів розробки
	Б	Простота створення	Обмежена сфера застосування
	В	Висока поширеність платформи	Обмежена сфера застосування
	Г	Подальше використання на широко поширених пристроях	Необхідність поглиблення вузькоспеціалізованих знань, складність обслуговування
	Д	Простота створення	Малий функціонал платформи, дороговизна Matlab
F2	*А*	Попередній досвід у розробці, простота побудови програмного продукту, широкий вибір бібліотек	Порівняно нижча швидкість роботи
	Б	Кросплатформний код, швидкодія програмного продукту, легкість роботи	Вимагає розробки на найнижчому рівні, довгий час на створення продукту
	В	Кросплатформний код, широке поширення, хороша інформаційна база, легкість роботи	Середня швидкодія, малий досвід в розробці
	Г	Широка інформаційна база, простота написання	Дороговизна програмного продукту, низька швидкодія, відсутність досвіду в розробці

Продовження таблиці 5.1

Основні функції	Варіанти реалізації	Переваги	Недоліки
ГЗ	*А*	Кросплатформність, продукт з відкритим кодом, є підтримка CUDA	Інтерфейс для використання тільки з Python
	Б	Кросплатформність, продукт з відкритим кодом, є підтримка CUDA	Інтерфейс орієнтований для використання з C/C++
	В	Продукт з відкритим кодом	Не підтримує Android та iOS,інтерфейс для використання з Python, Matlab
	Г	Має інтерфейс для всіх обраних варіантів мов, має відкритий код	Не підтримує Android та iOS
	Д	Продукт з відкритим кодом, є підтримка CUDA	Не підтримує Android та iOS
	Е	Простота процесу розробки	Має інтерфейс тільки для R
	Ж	Підтримка продукту розробником	Продукт з закритим кодом, має інтерфейс тільки для Matlab

Оберемо варіанти реалізації ПП:

Функція *F1:*

Через необхідність розробки універсального продукту, що міг би бути використаним на широкому обсязі пристроїв, був обраний варіант а).

Функція *F2:*

Досвід розробника у використанні мови вплине як на якість кінцевого продукту, так і на час його розробки. Також, для розв'язання поставленої задачі потрібна мова досить високого рівня, створення прототипів на якій не буде займати забагато часу. Так, були обрані варіанти а) та в).

Функція *F3:*

На вибір бібліотек впливає мова, що була обрана на попередньому кроці. Будемо орієнтуватися на простоту розробки та доступність необхідного функціоналу. Буди обрані функції д) та е).

Таким чином, будемо розглядати такі варіанти реалізації програмного продукту:

F1а – F2а – F3д

F1а – F2в – F3е

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.4 Обґрунтування системи параметрів програмного продукту

4.4.1 Опис параметрів

Маємо функції та вимоги до програмного продукту, що повинні бути впроваджені в ньому. Стоїть необхідність визначити параметри вибору, що будуть використані для обрахування коефіцієнта технічного рівня.

Щоб охарактеризувати програмний продукт, введемо наступні параметри:

- X1 – швидкодія мови програмування;

- X2 – об’єм пам’яті для збереження даних;

- X3 – час обробки даних, залежно від методу апроксимації;

- X4 – потенційний об’єм програмного коду.

X1: Характеризує швидкість виконання операцій в залежності від обраної мови для розробки продукту.

X2: Відображає об’єм пам’яті в оперативній пам’яті персонального комп’ютера, необхідний для збереження та обробки даних під час виконання програми.

X3: Відображає час, який витрачається на обробку даних.

X4: Описує розмір програмного коду який необхідно написати розробнику.

5.3.3 Кількісна оцінка параметрів

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію ПП як показано у табл. 5.2.

Таблиця 5.2 – Основні параметри ПП

Назва параметра	Умовні значення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	Оп/мс	19000	11000	2000
Об’єм пам’яті для збереження даних	X2	Мб	32	16	8
Час обробки даних алгоритмом	X3	мс	800	420	60
Потенційний об’єм програмного коду	X4	строки коду	2000	1500	1000

За даними таблиці 5.2 будуються графічні характеристики параметрів – рисунку 5.2 – рисунку 5.4.

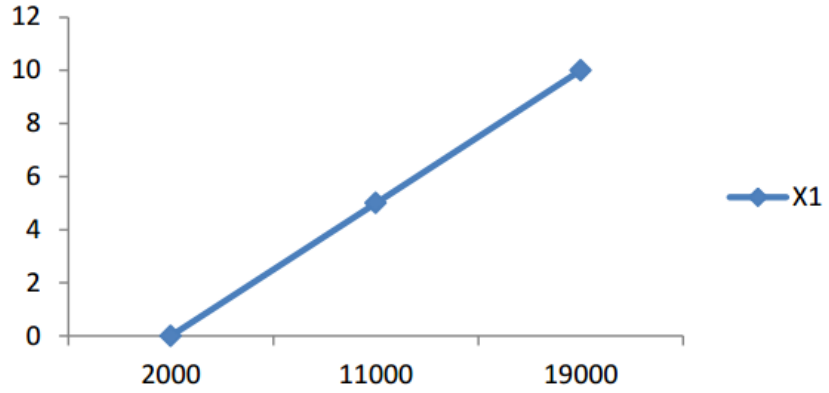


Рисунок 5.2 – X1, швидкодія мови програмування

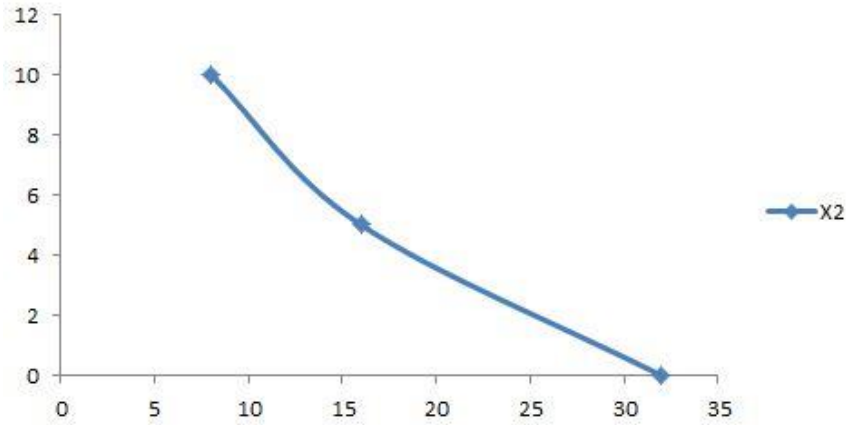


Рисунок 5.3 – X2, об’єм пам’яті для збереження даних

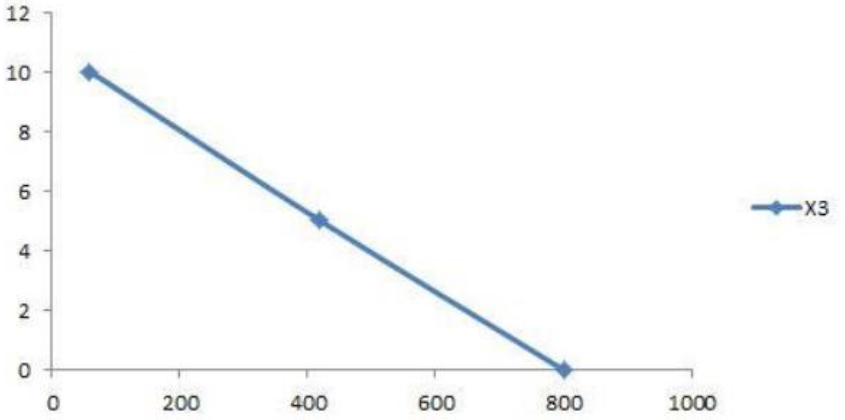


Рисунок 5.4 – X3, час виконання запитів користувача

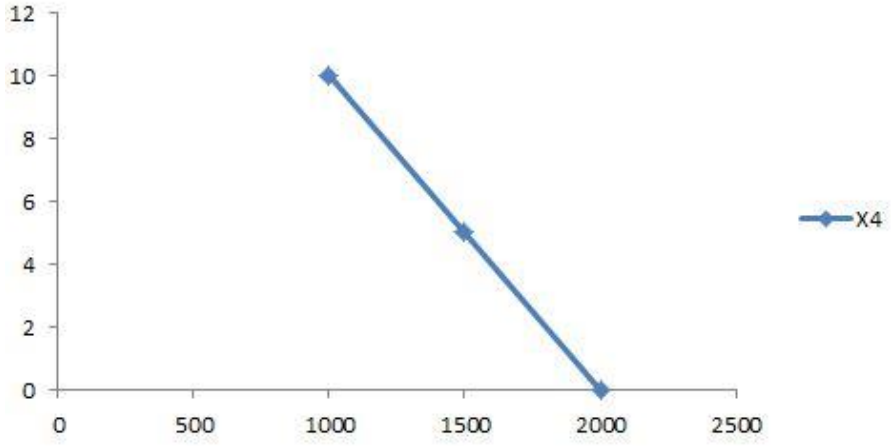


Рисунок 5.5 – X4, потенційний об’єм програмного коду

5.3.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

– визначення рівня значимості параметра шляхом присвоєння різних рангів;

– перевірку придатності експертних оцінок для подальшого використання;

– визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 5.3.

Таблиця 5.3 – Результати ранжування параметрів

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	4	3	4	4	4	4	4	27	0,75	0,56
X2	Об’єм пам’яті для збереження даних	Мб	4	4	4	3	4	3	3	25	-1,25	1,56
X3	Час обробки запитів користувача	Мс	2	2	1	2	1	2	2	12	-14,25	203,06
X4	Потенційний об’єм програмного коду	кількість строк коду	5	6	6	6	6	6	6	41	14,75	217,56
	Разом		15	15	15	15	15	15	15	105	0	420,75

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^{N} r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 105, \quad (5.1)$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 26{,}25. \quad (5.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T \tag{5.3}$$

Сума відхилень по всім параметрам повинна дорівнювати 0;

г)загальна сума квадратів відхилення:

$$S = \sum_{i=1}^{N} \Delta_i^2 = 420{,}75. \tag{5.4}$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3-n)} = \frac{12 \cdot 420{,}75}{7^2(5^3-5)} = 1{,}03 > W_k = 0{,}67 \tag{5.5}$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 5.4.

Таблиця 5.4 – Попарне порівняння параметрів

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	=	>	=	<	=	<	<	<	0,5
X1 і X3	<	<	<	<	<	<	<	<	0,5
X1 і X4	>	>	>	>	>	>	>	>	1,5
X2 і X3	<	<	<	<	<	<	<	<	0,5
X2 і X4	>	>	>	>	>	>	>	>	1,5
X3 і X4	>	>	>	>	>	>	>	>	1,5

Числове значення, що визначає ступінь переваги *i*–го параметра над *j*–тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1{,}5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases} \tag{5.6}$$

З отриманих числових оцінок переваги складемо матрицю А=$\| a_{ij} \|$.

Для кожного параметра зробимо розрахунок вагомості $K_{вi}$ за наступними формулами:

$$K_{\text{вi}} = \frac{b_i}{\sum_{i=1}^{n} b_i}, \text{де } b_i = \sum_{i=1}^{N} a_{ij} \ . \quad (5.7)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%).На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{\text{вi}} = \frac{b_i'}{\sum_{i=1}^{n} b_i'}, \text{де } b_i' = \sum_{i=1}^{N} a_{ij} b_j. \quad (5.8)$$

Як видно з таблиці 5.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 5.5 – Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	$K_{\text{вi}}$	b_i^1	$K_{\text{вi}}^1$	b_i^2	$K_{\text{вi}}^2$
X1	1,0	0,5	0,5	1,5	3,5	0,219	22,25	0,216	100	0,215
X2	1,5	1,0	0,5	1,5	4,5	0,281	27,25	0,282	124,25	0,283
X3	1,5	1,5	1,0	1,5	5,5	0,344	34,25	0,347	156	0,348
X4	0,5	0,5	0,5	1,0	2,5	0,156	14,25	0,155	64,75	0,154
Всього:					16	1	98	1	445	1

5.3.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів *X2*(об’єм пам’яті для збереження даних) та *X1* (швидкодія мови програмування) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра *X2* (об’єм пам’яті для збереження даних) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту а) 16 Мб або варіанту в) 8 Мб.

Абсолютне значення параметра *X3 (*час обробки даних) обрано не найгіршим (не максимальним), тобто це значення відповідає або варіанту д) 800 мс або варіанту е) 80мс.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 5.6):

$$K_K(j) = \sum_{i=1}^{n} K_{вi,j} B_{i,j}\,, \qquad (5.9)$$

де n – кількість параметрів;$K_{вi}$– коефіцієнт вагомості i–го параметра; B_i – оцінка i–го параметра в балах.

Таблиця 5.6 – Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1(X1)	А	11000	3,6	0,215	0,774
F2(X2)	А	16	3,4	0,283	0,962
	В	8	1,2	0,313	1,054
F3(X3,X4)	Д	800	2,4	0,154	0,248
	Е	80	1	0,348	0,835

За даними з таблиці 5.6 за формулою

$$K_K = K_{\text{ТУ}}[F_{1k}] + K_{\text{ТУ}}[F_{2k}] + \ldots + K_{\text{ТУ}}[F_{zk}], \quad (5.10)$$

визначаємо рівень якості кожного з варіантів:

K_{K1} = 0,774 + 0,962 + 0,248 = 1,98

K_{K2} = 0,774 + 1,054 + 0,835 = 2,66

Як видно з розрахунків, кращим є перший варіант, для якого коефіцієнт технічного рівня має найбільше значення.

5.3.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;

2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань. Загальна трудомісткість обчислюється як

$$T_O = T_P \cdot K_П \cdot K_{СК} \cdot K_М \cdot K_{СТ} \cdot K_{СТ.М}, \quad (5.11)$$

де $Т_Р$ – трудомісткість розробки ПП;$К_П$ – поправочний коефіцієнт; $К_{СК}$ – коефіцієнт на складність вхідної інформації; $К_М$ – коефіцієнт рівня мови програмування; $К_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм; $К_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру степеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $Т_Р$ =90 людино-днів. Поправочний коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $К_П$ = 1.5. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $К_{СК}$ = 1. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $К_{СТ}$ = 0.8. Тоді, за формулою 5.1, загальна трудомісткість програмування першого завдання дорівнює:

$Т_1 = 90 \cdot 1.5 \cdot 0.8 = 108$ людино-днів.

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $Т_Р$ =29 людино-днів, $К_П$ =0.8,$К_{СК}$ = 1,$К_{СТ}$ =0.8:

$Т_2 = 29 \cdot 0.8 \cdot 0.8 = 18.56$ людино-днів.

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$Т_I = (108 + 18.56 + 4.8 + 18.56) \cdot 8 = 1199,36$ людино-годин;

$Т_{II} = (108 + 18.56 + 6.91 + 18.56) \cdot 8 = 1216.24$ людино-годин;

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 8000 грн., один фінансовий аналітик з окладом 10000грн. Визначимо зарплату за годину за формулою:

$$С_ч = \frac{М}{T_m \cdot t} грн., \quad (5.12)$$

де М – місячний оклад працівників; T_m – кількість робочих днів тиждень; t – кількість робочих годин в день.

$$С_ч = \frac{8000+8000+10000}{3 \cdot 21 \cdot 8} = 51{,}58 \text{ грн.}$$

Тоді, розрахуємо заробітну плату за формулою

$$С_{ЗП} = С_ч \cdot Т_i \cdot К_Д, \quad (5.13)$$

де $С_ч$– величина погодинної оплати праці програміста; $Т_i$ – трудомісткість відповідного завдання; $К_Д$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

I. $С_{ЗП}$ = 51,58· 1199.36 · 1.2 = 74235,58грн.

II. $С_{ЗП}$ = 51,58· 1216.24 · 1.2 = 75280,39 грн.

Відрахування на соціальний становить 22,0%:

I. $С_{ВІД} = С_{ЗП} \cdot 0.22$ = 74235,58*0.22 = 16331,82 грн.

II. $С_{ВІД} = С_{ЗП} \cdot 0.22$ = 75280,39*0.22 = 16561,68 грн.

Тепер визначимо витрати на оплату однієї машино-години. ($С_М$)

Так як одна ЕОМ обслуговує одного програміста з окладом 8000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$С_Г = 12 \cdot М \cdot К_З$ = 12 · 8000· 0,2 = 19200 грн.

З урахуванням додаткової заробітної плати:

$С_{ЗП} = С_Г \cdot (1 + К_З) = 19200 \cdot (1 + 0.2) = 23040$ грн.

Відрахування на соціальний внесок:

$С_{ВІД} = С_{ЗП} \cdot 0.3677 = 23040 * 0,22 = 5068,80$ грн.

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 10000 грн.

$С_А = К_{ТМ} \cdot К_А \cdot Ц_{ПР} = 1.15 \cdot 0.25 \cdot 10000 = 2875$ грн.,

де $К_{ТМ}$– коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача; $К_А$– річна норма амортизації; $Ц_{ПР}$– договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$С_Р = К_{ТМ} \cdot Ц_{ПР} \cdot К_Р = 1.15 \cdot 10000 \cdot 0.05 = 575$ грн.,

де $К_Р$– відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$Т_{ЕФ} = (Д_К – Д_В – Д_С – Д_Р) \cdot t_З \cdot К_В = (365 – 104 – 8 – 16) \cdot 8 \cdot 0.9 = 1706.4$ годин,

де $Д_К$ – календарна кількість днів у році; $Д_В$, $Д_С$ – відповідно кількість вихідних та святкових днів; $Д_Р$ – кількість днів планових ремонтів устаткування; t –кількість робочих годин в день; $К_В$– коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$С_{ЕЛ} = Т_{ЕФ} \cdot N_С \cdot К_З \cdot Ц_{ЕН} = 1706,4 \cdot 0,22 \cdot 0,78 \cdot 2,28974 = 670,48$ грн.,

де $N_С$ – середньо-споживча потужність приладу; $К_З$– коефіцієнтом зайнятості приладу; $Ц_{ЕН}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$С_Н = Ц_{ПР} \cdot 0.67 = 10000 \cdot 0,67 = 6700$ грн.

Тоді, річні експлуатаційні витрати будуть:

$$С_{ЕКС} = С_{ЗП} + С_{ВІД} + С_А + С_Р + С_{ЕЛ} + С_Н$$

$С_{ЕКС}$ = 23040 + 5068,80 + 2875 + 575 + 670,48 + 6700 = 38929,28 грн.

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$С_{М-Г} = С_{ЕКС} / Т_{ЕФ}$ = 38929,28 /1706,4 = 22,81 грн/час.

Оскільки в даному випадку всі роботи, які пов‘язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$С_М = С_{М-Г} \cdot Т$$

I. $С_М$ = 22,81 * 1199,36 = 27357,40 грн.;

II. $С_М$ = 22,81 * 1216.24 = 27742,43 грн.;

Накладні витрати складають 67% від заробітної плати:

$$С_Н = С_{ЗП} \cdot 0,67$$

I. $С_Н$ = 27357,40 * 0,67 = 18329,45 грн.;

II. $С_Н$ = 27742,43 * 0,67 = 18587,43 грн.;

Отже, вартість розробки ПП за варіантами становить:

$$С_{ПП} = С_{ЗП} + С_{ВІД} + С_М + С_Н$$

I. $С_{ПП}$ = 23040 + 5068,80 + 27357,40 + 18329,45 = 73795.65 грн.;

II. $С_{ПП}$ = 23040 + 5068,80 + 27742,43 + 18587,43 = 74438.66 грн.;

4.4.6 Вибір кращого варіанта ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$К_{ТЕРj} = К_{кj} / С_{фj}, \quad (5.14)$$

$К_{ТЕР1}$ = 1,98 / 73795.65 = $2,683 \cdot 10^{-5}$;

$К_{ТЕР2}$ = 2,66 / 74438.66 = $3,573 \cdot 10^{-5}$;

Як бачимо, найбільш ефективним є другий варіант реалізації програми з коефіцієнтом техніко-економічного рівня $К_{ТЕР1}= 3,573 \cdot 10^{-5}$.

4.5 Висновки до розділу 4

В даному розділі проведено повний функціонально-вартісний аналіз ПП, який було розроблено в рамках дипломного проекту. Процес аналізу можна умовно розділити на дві частини.

В першій з них проведено дослідження ПП з технічної точки зору: було визначено основні функції ПП та сформовано множину варіантів їх реалізації; на основі обчислених значень параметрів, а також експертних оцінок їх важливості було обчислено коефіцієнт технічного рівня, який і дав змогу визначити оптимальну з технічної точки зору альтернативу реалізації функцій ПП.

Другу частину ФВА присвячено вибору із альтернативних варіантів реалізації найбільш економічно обґрунтованого. Порівняння запропонованих варіантів реалізації в рамках даної частини виконувалось за коефіцієнтом ефективності, для обчислення якого були обчислені такі допоміжні параметри, як трудомісткість, витрати на заробітну плату, накладні витрати.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості $К_{ТЕР} = 3,573 \cdot 10^{-5}$.

Цей варіант реалізації програмного продукту має такі параметри:

– цільова платформа ПП – кросплатформний;

– мова програмування R;

– бібліотека моделювання нейронних мереж – deepnet.

Даний варіант виконання програмного комплексу є найоптимальнішим з точки зору глибини розробки та корисності продукту при найменших затратах часу.

ВИСНОВКИ

В даній роботі проведений огляд ефективних методів прогнозування розвитку фінансових процесів у формі часових рядів, була порівняна якість прогнозування за множиною статистичних критеріїв якості. Далі були описані архітектури нейронних мереж, що можуть використовуватися для розв’язання задач з цього класу, було обрано метод розв’язання поставленої задачі.

Створено програмний модуль на мові програмування R, що будує модель обраної архітектури, обробляє отримані дані для завантаження в нейромережу, виконує процес її навчання та виводить результати прогнозування. Важливою особливістю модулю є кросплатформність, можливість його підключення до інших програм та малу кількість часу необхідного на налаштування моделі (приблизно 15 секунд на даних розміром 9000 записів).

Програмні методи для вирішення поставленої задачі були визначені за допомогою функціонально-вартісного аналізу, що довів їх обґрунтованість та ефективність за критеріями економічними, зручності тощо.

Побудована модель через малий час навчання та ефективність передбачення, можуть бути використані в реальних задачах, більше того, застосовані для прогнозування напряму розвитку фінансових процесів в реальному часі.

В роботі також був описаний процес збору та перетворення даних для експериментів, обґрунтовано, чому саме такі дані обрані до використання.

Побудована модель показала високі результати в задачах короткострокового прогнозування напрямку руху фінансових процесів складного характеру, зокрема цін на акції та цін на природну сировину. Точність прогнозів моделі перевищила точність моделі АРІКС.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Vapnik V. N. An Overview of Statistical Learning Theory / *IEEE TRANSACTIONS ON NEURAL NETWORKS*, 1999, pp. 12–21.
2. Harrington P. Machine Learning in Action. Manning, 2012. p. 127.
3. Kim K. Financial time series forecasting using support vector machines / *Department of Information Systems*, Dongguk University, 2003, pp. 34–37.
4. Panigrahi S. S., Mantri J. K. A text based Decision Tree model for stock market forecasting / *Green Computing and Internet of Things (ICGCIoT)*, International Conference, Noida, 2015, pp. 40–42.
5. Siripurapu A. Convolutional Networks for Stock Trading / *Stanford University*, Department of Computer Science, 2014, pp. 55–62.
6. G. Iuhasz, M. Tirea, V. Negru Neural Network Predictions of Stock Price Fluctuations / *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, 2012 14th International Symposium on, Timisoara, 2012, pp. 32–37.
7. Poon S.-H. A Practical Guide to Forecasting Financial Market Volatility. John Wiley & Sons, Ltd, 2005. pp. 178–220.
8. Qiu M., Song Y. Predicting the Direction of Stock Market Index Movement Using an Optimized Artificial Neural Network / *Department of Systems Management*, Fukuoka Institute of Technology, Fukuoka, Japan, 2016, p. 56.
9. Clemen T. R. Combining forecasts: A review and annotated bibliography / *International Journal of Forecasting*, North-Holland,1989, pp. 22–34.
10. Shilling H. The International Guide to Securities Market Indices / *Routledge Library Editions: Financial Markets*, 1996, p. 53.
11. Gurney K. An Introduction to Neural Networks *Routledge Library Editions: Financial Markets*, 2002, pp. 78–82.

12. Stuber G. Implications of Uncertainty about Long-Run Inflation and the Price Level / *Research Department*, Bank of Canada, 2001, pp. 12–18.
13. Vitek F. An Empirical Analysis of Dynamic Interrelationships Among Inflation, Inflation Uncertainty, Relative Price Dispersion, and Output Growth / *Research Department*, Bank of Canada, 2002, pp. 89 -101.
14. Xekalaki E., Degiannakis S. ARCH Models for Financial Applications / *Department of Statistics*, Athens University of Economics and Business, Greece, 2010, p. 155.
15. Бідюк П.І. Адаптивне прогнозування фінансово-економічних процесів на основі принципів системного аналізу // Економіко-математичне моделювання процесів бізнесу, 2009, С. 8-20.
16. Cooper I. Asset Pricing Implications of Non-Convex Adjustment Costs of Investment / *University of Chicago*, Graduate School of Business, 2002, pp. 9-27.

ДОДАТОК А ІЛЮСТРАТИВНІ МАТЕРІАЛИ ДЛЯ ДОПОВІДІ

Тема: Моделі у формі нейронних мереж в задачах прогнозування розвитку фінансових процесів

Виконав: Студент групи КА-41
Барзій І.І.

Науковий курівник: д.т.н.б професор Бідюк П.І.

Об'єкт, предмет і мета дослідження

Об'єкт дослідження – фінансові процеси у формі часових рядів

Мета дослідження – покращення методів прогнозування розвитку фінансових процесів різного характеру з використанням моделей нейронних мереж.

Предмет дослідження – моделі нейронних мереж, методи їх побудови та знаходження параметрів, формування вхідних даних

Актуальність

- Прогнозування складних фінансових процесів – задача, що вирішується багатьма підприємствами та установами. Застосування методів машинного навчання дає гарні результати.
- В літературі та роботах мало точної інформації про результати таких досліджень через їх комерційну значущість.
- Результати отримані в роботі можуть бути використані для прогнозування часових рядів або даних іншого характеру.

Постановка задачі

- Провести аналіз найефективніших методів прогнозування динаміки процесів фінансового характеру
- Дослідити можливі реалізації архітектури нейронних мереж та обрати таку, щоб її властивості підходили для вирішення поставленої задачі
- Знайти фінансові дані та утворити з них набор характеристик для подачі в модель
- Реалізувати обрану модель, навчити її на отриманих даних
- Запропонувати методи покращення отриманих результатів прогнозу
- Порівняти результати прогнозу між собою та з іншими моделями

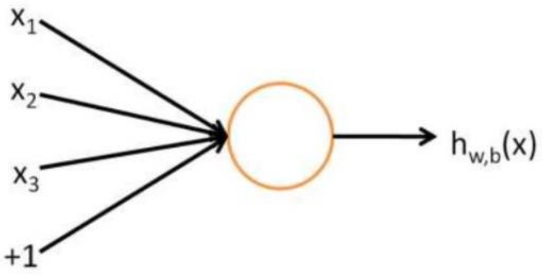
Огляд існуючих методів передбачення

1. Статистичні методи
2. Методи машинного навчання
    1. Методи опорних векторів
    2. Дерева рішень
    3. Нейронні мережі

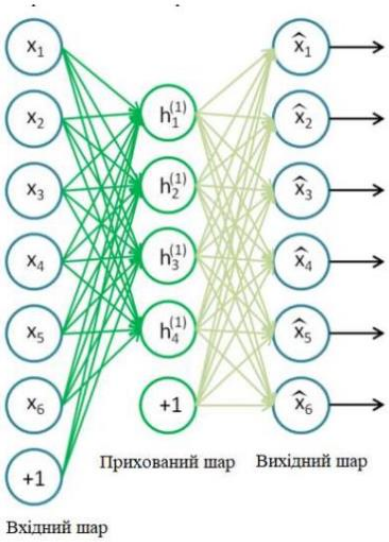
Процеси, що досліджуються

- **Курси валют**
    - EUR/USD
    - GBP/USD
    - BITCOIN/USD
    - USD/JPY
- **Акції**
    - Amazon
    - Facebook
    - Tesla
    - Apple
- **Сировина**
    - Нафта
    - Золото
    - Срібло
- **Індекси**
    - S&P 500
    - STOXX 50

Модель нейронної мережі



Нейрон



Автоенкодер

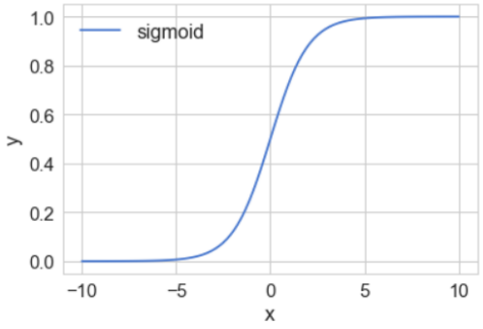
Модель нейронної мережі

Дані на вході та виході шару мережі

$$Input = x \quad Output = f(Wx + b)$$

Активаційна функція (сигмоїд)

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$



Модель нейронної мережі

Вартісна функція: $J = -\frac{1}{m}\sum_{i=0}^{m}\sum_{j=0}^{n}\left(y_j^{(i)}\log\left(a_j^{[2](i)}\right) + (1-y_j^{(i)})\log\left(1-a_j^{[2](i)}\right)\right)$

де

a – дані на виході прихованого шару

y – вхідні дані на шар

n – кількість класів

m – кількість прикладів

Вартісна функція з регуляризацією:

$$J = -\frac{1}{m}\sum_{i=0}^{m}\sum_{j=0}^{n}\left(y_j^{(i)}\log\left(a_j^{[2](i)}\right) + (1-y_j^{(i)})\log\left(1-a_j^{[2](i)}\right)\right) + \frac{\lambda_1}{m}(\|w_1\|_1 + \|w_2\|_1) + \frac{\lambda_2}{2m}(\|w_1\|_2^2 + \|w_2\|_2^2)$$

де

λ_1, λ_2– коефіцієнти регуляризації

w_1, w_1– ваги першого та другого прихованих шарів

Модель нейронної мережі

Алгоритм forward propagation:

$$z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}$$
$$a^{[1](i)} = \sigma(z^{[1](i)})$$
$$z^{[2](i)} = W^{[2]}a^{[1](i)} + b^{[2]}$$
$$\hat{y}^{(i)} = a^{[2](i)} = \sigma(z^{[2](i)})$$
$$y^{(i)}_{prediction} = argmax(a^{[2](i)})$$

де

x – дані, що надійшли до входу мережі

W – ваги шарів

b – зміщення шарів

σ – функція сигмоїд

y – результуючі дані

Модель нейронної мережі

Алгоритм backward propagation:

$$dz^{[2]} = a^{[2]} - y$$

$$dW^{[2]} = dz^{[2]}a^{[1]T} + \frac{\lambda_1}{m}sign(W^{[2]}) + \frac{\lambda_2}{m}W^{[2]}$$

$$db^{[2]} = dz^{[2]}$$

$$dz^{[1]} = W^{[2]T}dz^{[2]} * \sigma'(z^{[1]})$$

$$dW^{[1]} = dz^{[1]}x^T + \frac{\lambda_1}{m}sign(W^{[1]}) + \frac{\lambda_2}{m}W^{[1]}$$

$$db^{[1]} = dz^{[1]}$$

де

dz – похибка мережі на шарах

dW – дельта вагів

dz – дельта зміщення

σ' – похідна функції сигмоїд

Модель нейронної мережі

Оновлення параметрів мережі:

$$\theta = \theta - \alpha\frac{\partial J}{\partial \theta}$$

де

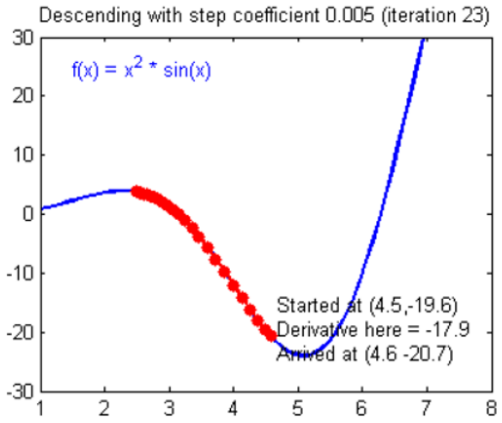
Θ – параметр мережі (dW або db)

α – швидкість навчання

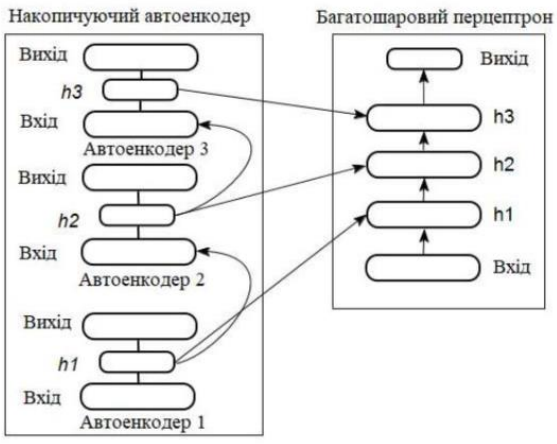
$\frac{\partial J}{\partial \theta}$ – дельта параметру (dW або db)

Модель нейронної мережі

Зміна вартісної функції:

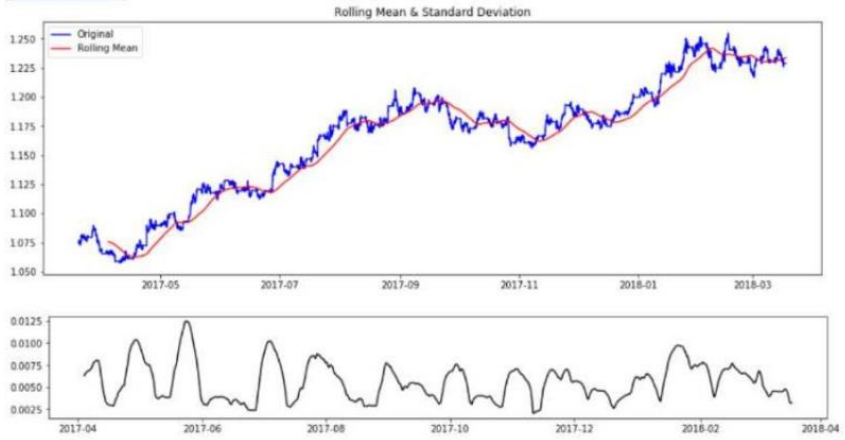


Модель нейронної мережі



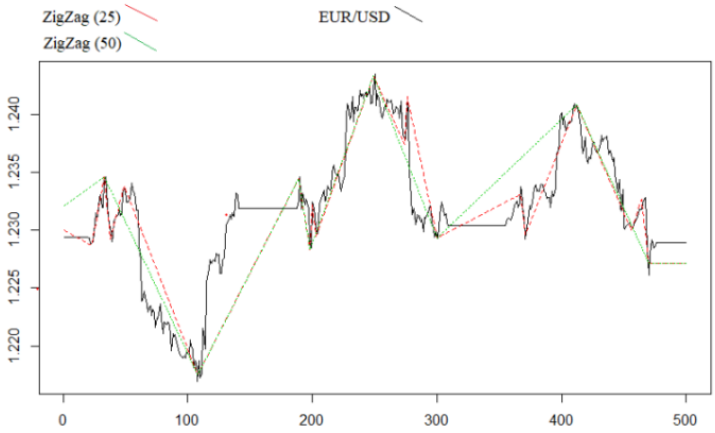
Накопичуючий автоенкодер

Вхідні дані

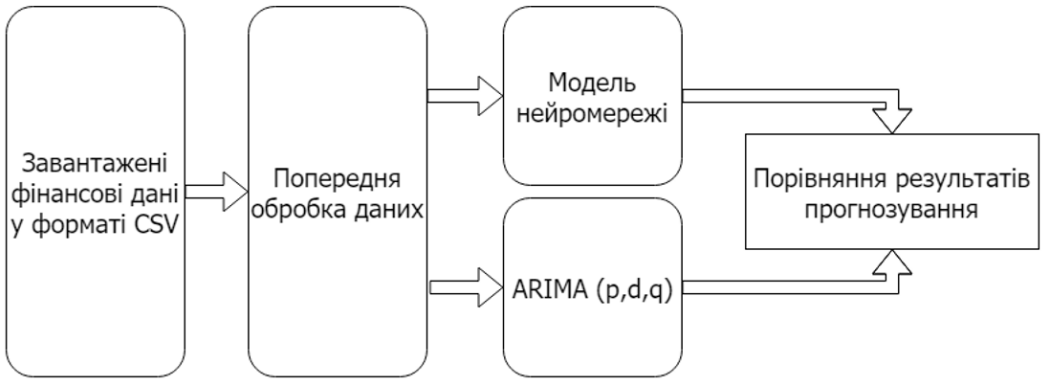


EUR/USD 1H

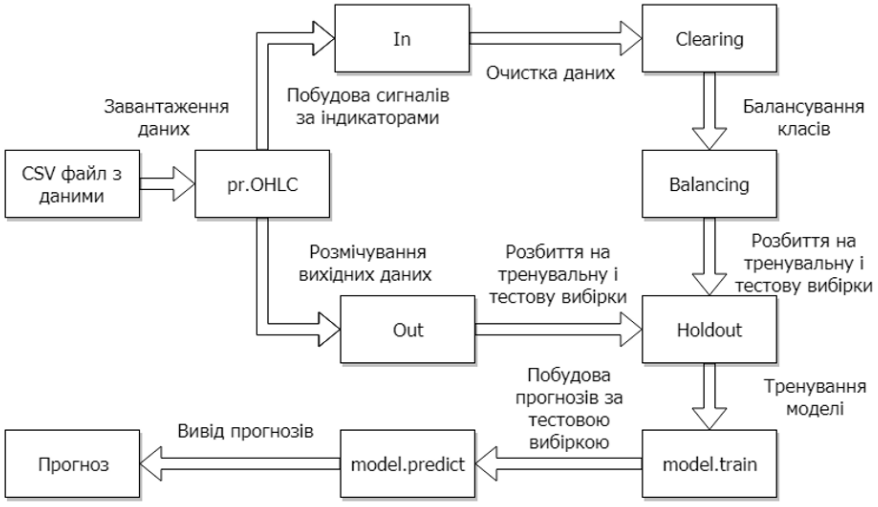
Розмічені вхідні дані



Загальний процес обробки даних у роботі



Структура створеного програмного модулю



Точність прогнозів побудованої моделі

Часовий ряд	Точність передбачення	Результат отриманої стратегії	Результат ідеальної стратегії
Євро - Долар	76%	0,20%	0,85%
Британський Фунт - Долар	42%	-2,07%	4,02%
Японська Йена - Долар	48%	0,63%	3,22%
Долар - Біткойн	42%	-3,38%	1,77%
Нафта	61%	1,05%	-2,38%
Срібло	64%	-0,42%	-0,46%
Золото	64%	-2,05%	-0,9%
Apple	71%	0,06%	1,32%
Amazon	77%	0,51%	0,99%
Facebook	54%	-0,07%	1,31%
Tesla	-	-	-
STOXX Europe 50	67%	1,6%	0,8%
S&P 500	45%	-0,3%	0,07%

Точність прогнозів моделі АРІКС

Часовий ряд	Точність передбачення	Результат отриманої стратегії	Результат ідеальної стратегії
Євро - Долар	52%	-	-
Британський Фунт - Долар	54%	-	-
Японська Йена - Долар	53%	-	-
Долар - Біткойн	45%	-	-
Нафта	52%	-	-
Срібло	60%	-	-
Золото	57%	-	-
Apple	60%	-	-
Amazon	61%	-	-
Facebook	62%	-	-
STOXX Europe 50	59%	-	-
S&P 500	45%	-	-

Дякую за увагу

ДОДАТОК Б ЛІСТИНГ ПРОГРАМИ

Функція збору даних

```
pr.OHLC <- function (o, h, l, c)
{
 #Объединим векторы котировок в матрицу, предварительно их развернув
 #Индексация векторов таймсерий в R начинается с 1.
 #Направление индексации- от старых к новым.
 price <- cbind(Open = rev(o), High = rev(h), Low = rev(l), Close = rev(c))
 Med <- (price[, 2] + price[, 3])/2
 CO <- price[, 4] - price[, 1]
 #добавим в матрицу Med и CO
 price <- cbind(price, Med, CO)
}
```

Функції виводу індикаторів

```
library(TTR)
adx<-ADX(price, n = 16)
plot.ts(head(adx, 200))

ar<-aroon(price[ , c('High', 'Low')], n = 16)[ ,'oscillator']
plot(head(ar, 200), t = "l")
abline(h = 0)

cci<-CCI(price[ ,2:4], n = 16)
plot.ts(head(cci, 200))
abline(h = 0)
```

```
chv<-chaikinVolatility(price[ , 2:4], n = 16)
summary(chv)
plot(head(chv, 200), t = "l")
abline(h = 0)

cmo<-CMO(price[ ,'Med'], n = 16)
plot(head(cmo, 200), t = "l")
abline(h = 0)

macd<-MACD(price[ ,'Med'], 12, 26, 9)[ ,'macd']
plot(head(macd, 200), t = "l")
abline(h = 0)

osma<-macd - MACD(price[ ,'Med'],12, 26, 9)[ ,'signal']
plot(head(osma, 200), t = "l")
abline(h = 0)

rsi<-RSI(price[ ,'Med'], n = 16)
plot(head(rsi, 200), t = "l")
abline(h = 50)

stoh<-stoch(price[ ,2:4], 14, 3, 3)
plot.ts(head(stoh, 200))

smi<-SMI(price[ ,2:4],n = 13, nFast = 2, nSlow = 25, nSig = 9)
plot.ts(head(smi, 200))
```

```
vol<-volatility(price[ ,1:4],n = 16,calc = "yang.zhang", N =96)
 plot.ts(head(vol, 200))
```

Функція вирахування матриці вхідних даних

```
In<-function(p = 16){
  adx<-ADX(price, n = p);
  ar<-aroon(price[ ,c('High', 'Low')], n=p)[ ,'oscillator'];
  cci<-CCI(price[ ,2:4], n = p);
  chv<-chaikinVolatility(price[ ,2:4], n = p);
  cmo<-CMO(price[ ,'Med'], n = p);
  macd<-MACD(price[ ,'Med'], 12, 26, 9)[ ,'macd'];
  osma<-macd - MACD(price[ ,'Med'],12, 26, 9)[ ,'signal'];
  rsi<-RSI(price[ ,'Med'], n = p);
  stoh<-stoch(price[ ,2:4],14, 3, 3);
  smi<-SMI(price[ ,2:4],n = p, nFast = 2, nSlow = 25, nSig = 9);
  vol<-volatility(price[ ,1:4],n = p,calc="yang.zhang", N=96);
  In<-cbind(adx, ar, cci, chv, cmo, macd, osma, rsi, stoh, smi, vol);
  return(In)
}
```

Функція формування вихідних даних

```
Out<-function(ch=0.0037){
  # ЗигЗаг имеет значения (определен) на каждом баре а не только в вершинах
  zz<-ZigZag(price[ ,'Med'], change = ch, percent = F, retrace = F, lastExtreme = T);
```

```
  n<-1:length(zz);
  # На последних барах неопределенные значения заменим на последние
известные
  for(i in n) { if(is.na(zz[i])) zz[i] = zz[i-1];}
  #Определим скорость изменения ЗигЗага и сдвинем на один бар в будущее
  dz<-c(diff(zz), NA);
  #Если скорость >0 - сигнал = 0(Buy), если <0, сигнал = 1 (Sell) иначе NA
  sig<-ifelse(dz>0, 0, ifelse(dz<0, 1, NA));
  return(sig);
}
```

Функція балансування класів

```
Balancing<-function(DT){
  #Вычисляем таблицу с количеством классов
  cl<-table(DT[ ,ncol(DT)]);
  #Если разбаланс меньше 15%, возвращаем исходную матрицу
  if(max(cl)/min(cl)<= 1.15) return(DT)
  #Иначе балансируем в большую сторону
  DT<-if(max(cl)/min(cl)> 1.15){
        upSample(x = DT[ ,-ncol(DT)],y = as.factor(DT[ , ncol(DT)]), yname = "Y")
       }
  #Преобразуем y (фактор) в число
  DT$Y<-as.numeric(DT$Y)
  #Перекодируем y из 1,2 в 0,1
  DT$Y<-ifelse(DT$Y == 1, 0, 1)
  #Преобразуем датафрейм в матрицу
```

```
 DT<-as.matrix(DT)
 return(DT);
   }
```

Створення моделі SAE

```
sae.dnn.train(x, y, hidden = c(10), activationfun = "sigm", learningrate = 0.8,
momentum = 0.5, learningrate_scale = 1, output = "sigm", sae_output = "linear",
  numepochs = 3, batchsize = 100, hidden_dropout = 0, visible_dropout = 0)
```

Перевірка часу навчання моделі

```
system.time(SAE<-sae.dnn.train(x= x.tr, y= y[t$tr], hidden=c(100,100,100),
activationfun = "tanh", learningrate = 0.6, momentum = 0.5, learningrate_scale =
1.0, output = "sigm", sae_output = "linear", numepochs = 10, batchsize = 100,
hidden_dropout = 0, visible_dropout = 0))
```

Перевірка точності отриманої моделі

```
pr<-ifelse(pr.sae>mean(pr.sae), 1, 0)
confusionMatrix(y[t$ts], pr)
```

Знаходження кількості пунктів, котрі модель змогла передбачити після навчання

```
new.x<-predict(spSign,tail(dt[ ,-ncol(dt)], 500))
pr.sae1<-nn.predict(SAE, new.x)
pr.sig<-ifelse(pr.sae1>mean(pr.sae1), -1, 1)
new.y<-ifelse(tail(dt[  , ncol(dt)], 500) == 0, 1, -1)
cm1<-confusionMatrix(new.y, pr.sig)
cm1
```

Тіло програми

```
eur <- "C:/Users/illya/Diploma data/EURUSD_H1.csv"
gbp <- "C:/Users/illya/Diploma data/GBPUSD_H1.csv"
btc <- "C:/Users/illya/Diploma data/Bitcoin_H1.csv"
jpy <- "C:/Users/illya/Diploma data/USDJPY_H1.csv"
amz <- "C:/Users/illya/Diploma data/Amazon_H1.csv"
apl <- "C:/Users/illya/Diploma data/Apple_H1.csv"
oil <- "C:/Users/illya/Diploma data/Brent_H1.csv"
```

```
gld <- "C:/Users/illya/Diploma data/Gold_H1.csv"
slv <- "C:/Users/illya/Diploma data/Silver_H1.csv"
e50 <- "C:/Users/illya/Diploma data/EUS_H1.csv"
snp <- "C:/Users/illya/Diploma data/USA500_H1.csv"
tsl <- "C:/Users/illya/Diploma data/Tesla_H1.csv"
fcb <- "C:/Users/illya/Diploma data/Facebook_H1.csv"


pric <- read.csv( amz, header=TRUE)
pric <- read.csv( fcb, header=TRUE)
pric <- read.csv( tsl, header=TRUE)
pric <- read.csv( snp, header=TRUE)


price <- pr.OHLC(pric[,2],pric[,3],pric[,4],pric[,5])
#pric = pric / 100000

library(TTR)

X <- In()

Y  <- Out()
table(Y)
#table(Y)

dt<-Clearing(X,Y); nrow(dt)

dt.b<-Balancing(dt)
x<-dt.b[ ,-ncol(dt.b)]
y<-dt.b[ , ncol(dt.b)]

library('rminer')

t<-holdout(y, ratio = 8/10, mode = "order")

library("caret")

spSign<-preProcess(x[t$tr, ], method = "spatialSign")
tax.tr<-predict(spSign, x[t$tr, ])
x.tr<-predict(spSign, x[t$tr, ])
x.ts<-predict(spSign, x[t$ts, ])
```

```
#Testing itself

library('deepnet')

sae.dnn.train(x, y, hidden = c(10), activationfun = "sigm", learningrate = 0.8, momentum = 0.5, learningrate_scale = 1, output = "sigm", sae_output = "linear",
              numepochs = 3, batchsize = 100, hidden_dropout = 0, visible_dropout = 0)

#Time of testing

system.time(SAE<-sae.dnn.train(x= x.tr, y= y[t$tr], hidden=c(100,100,100), activationfun = "tanh", learningrate = 0.6, momentum = 0.5, learningrate_scale = 1.0, output = "sigm", sae_output = "linear", numepochs = 10, batchsize = 100, hidden_dropout = 0, visible_dropout = 0))

#Checking out the results

pr.sae<-nn.predict(SAE, x.ts);
#summary(pr.sae)

pr<-ifelse(pr.sae>mean(pr.sae), 1, 0)
confusionMatrix(y[t$ts], pr)

#Now checking the possible profit of those trades

new.x<-predict(spSign,tail(dt[ ,-ncol(dt)], 500))
pr.sae1<-nn.predict(SAE, new.x)
pr.sig<-ifelse(pr.sae1>mean(pr.sae1), -1, 1)
#table(pr.sig)
#pr.sig


new.y<-ifelse(tail(dt[  , ncol(dt)], 500) == 0, 1, -1)
#table(new.y)
#new.y


cm1<-confusionMatrix(new.y, pr.sig)
#cm1
```

```
bal<-cumsum(tail(price[ , 'CO'], 500) * pr.sig)
plot(bal, t = "l")
abline(h = 0)
table(tail(bal))
#Looking at the ideal possible result

#bal.zz<-cumsum(tail(price[ , 'CO'], 500) * new.y)
#plot(bal.zz,  t = "l")
#lines(bal,  col = 2)

bal.zz<-cumsum(tail(price[ , 'CO'], 500) * new.y)
plot(bal.zz,  t = "l")
lines(bal,  col = 2)

table(tail(bal.zz))
```