ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ТЕЛЕКОМУНІКАЦІЙ

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему: «**ВИКОРИСТАННЯ АВТОКОДУВАЛЬНИКІВ ДЛЯ ПІДВИЩЕННЯ ТОЧНОСТІ ПРОГНОЗУВАННЯ ПОПИТУ**»

Виконав: студент 6 курсу, групи САДМ-61
спеціальності 124 Системний аналіз
(шифр і назва спеціальності)

Цапро І.В.
(прізвище та ініціали)

Керівник _____
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Нормоконтроль _____
(прізвище та ініціали)

Київ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ТЕЛЕКОМУНІКАЦІЙ

Кафедра Системного аналізу
Ступінь вищої освіти Магістр
Спеціальність 124 Системний аналіз
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Системного аналізу
Гордієнко Т.Б.
_____ _____ 20__ року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Цапро Ігор Вікторович

1. Тема роботи: «Використання автокодувальників для підвищення точності прогнозування попиту», керівник роботи Гордієнко Тетяна Богданівна, к.т.н., професор, затверджені наказом вищого навчального закладу від _____ 2021 року № _____.

2. Строк подання студентом роботи _____ 2021 р.

3. Вихідні дані до роботи:
   1. Машинне навчання.
   2. Нейронні мережі.
   3. Автокодувальники.
   4. Продуктова канібалізація.
   5. Прогнозування попиту.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
   1. Вплив продуктової канібалізації на попит товарів.
   2. Реалізація автокодувальника.
   3. Порівняння результатів використаних методів.
5. Графічна частина роботи представлена на _____ слайдах презентації.
6. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Пошук та збір теоретичної інформації	01.11.21 – 03.11.21	
2	Опрацювання теоретичної інформації	04.11.21 – 06.11.21	
3	Розробка гіпотез	07.11.21 – 08.11.21	
4	Перевірка гіпотез	09.11.21 – 11.11.21	
5	Підготовка до експериментів	12.11.21 – 13.11.21	
6	Проведення експериментів	14.11.21 – 16.11.21	
7	Результати експериментів	17.11.21 – 20.11.21	
8	Порівняння, висновки	21.11.21 – 21.11.21	

Студент Цапро І.В

(підпис) (прізвище та ініціали)

Керівник роботи Гордієнко Т.Б

(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина магістерської кваліфікаційної роботи: стор. 77, рис. 76, табл. 7, дж 12.

Об'єкт дослідження – застосування автокодувальників у прогнозуванні попиту.

Предмет дослідження – стискання інформації автокодувальниками.

Мета роботи – отримання більш високої точності прогнозування попиту з використанням автокодувальників, аніж стандартними підходами.

Методи дослідження – розробка та навчання нейронної мережі з архітектурою автокодувальника, навчання стандартних методів регресійного прогнозування.

У роботі проведено аналіз предметної області прогнозування попиту та продуктової канібалізації. Отримано оцінки точності прогнозування попиту стандартними методами та за допомогою автокодувальника.

Встановлено практичність застосування автокодувальника для прогнозування попиту.

НЕЙРОННА МЕРЕЖА, НЕЙРОМЕРЕЖА, КОДУВАЛЬНИК, КОДЕР, ПРИХОВАНИЙ ПРОСТІР, МАШИННЕ НАВЧАННЯ, АВТОКОДУВАЛЬНИК, РЕГРЕСІЙНИЙ АНАЛІЗ, ПРОГНОЗУВАННЯ, ПРОДУКТОВА КАНІБАЛІЗАЦІЯ, MAPE, MAE, MSE, R-SQUARED.

ЗМІСТ

ВСТУП

У сучасному світі машинне навчання має потужний науковий та прикладний потенціал, а використання нейронних мереж в задачах прогнозування, обробки природної мови та комп’ютерного зору є однією з передових і найбільш перспективною сферою досліджень.

Прогнозування попиту – складна задача. Збір інформації, аналіз даних, виявлення закономірностей у історичних продажах та прогнозування – все це фундаментально впливає на час та ціну перевезень, ефективність та доцільність роботи систем постачання товарів, розкладання продуктових одиниць на полицях тощо. Кожен фактор сильно впливає на остаточну ціну товару. Тому розробка нових алгоритмів прогнозування попиту забезпечує більш ефективний та дешевий шлях товарів від початку їх виготовлення до кінцевого споживача.

Знаходження взаємозв’язків між товарами на полиці та цей вплив на загальне споживання – є значущою задачею прогнозування попиту. У свою чергу, обробка та використання інформації про взаємодію товарів мають сильний вплив на точність прогнозування, що підкреслює значущий статус даної сфери дослідження.

1 ТЕОРЕТИЧНІ ВІДОМОСТІ

1.1 Прогнозування попиту

1.1.1 Визначення поняття

Прогнозування попиту – це процес використання прогнозного аналізу історичних даних для оцінки та прогнозування майбутнього попиту споживачів на продукт або послугу. Прогнозування попиту допомагає компанії приймати більш обґрунтовані рішення щодо поставок, які оцінюють загальний обсяг продажів і доходи на майбутнє.

За допомогою прогнозування попиту підприємства можуть оптимізувати запаси, передбачивши майбутні продажі на основі історичного аналізу даних про продажі, щоб приймати обґрунтовані бізнес-рішення щодо всього, від планування запасів і складських потреб до миттєвих продажів і очікувань клієнтів. Приклад історичних продажів на рис. 1.1.

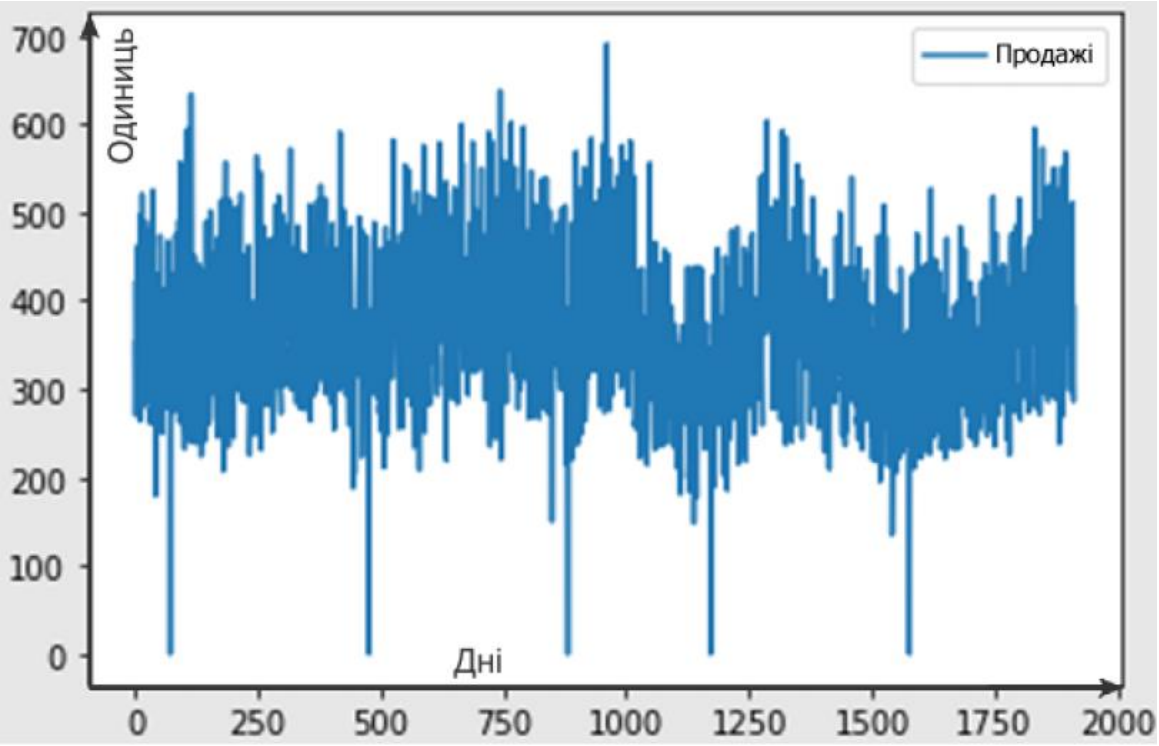


Рис. 1.1. Історичні продажі

Немає бізнесу без попиту. А без глибокого розуміння попиту підприємства не можуть приймати правильні рішення щодо витрат на маркетинг, виробництво, персонал тощо.

Прогнозування попиту ніколи не буде на 100% точним, але є кроки, які можна зробити, щоб покращити час виробництва, підвищити операційну ефективність, заощадити гроші, запустити нові продукти та забезпечити кращий досвід для клієнтів.

Прогнозування попиту допомагає зменшити ризики та приймати ефективні фінансові рішення, які впливають на прибуток, грошовий потік, розподіл ресурсів, можливості розширення, облік запасів, операційні витрати, персонал та накладні витрати. Усі стратегічні та оперативні плани формуються навколо прогнозування попиту.

Прогнозування попиту дозволяє постачати продукти, які потрібні клієнтам, коли вони цього захочуть. Прогнозування попиту вимагає синхронізації виконання замовлення з маркетингом перед запуском. Приклад прогнозування середнім значенням на рис. 1.2.

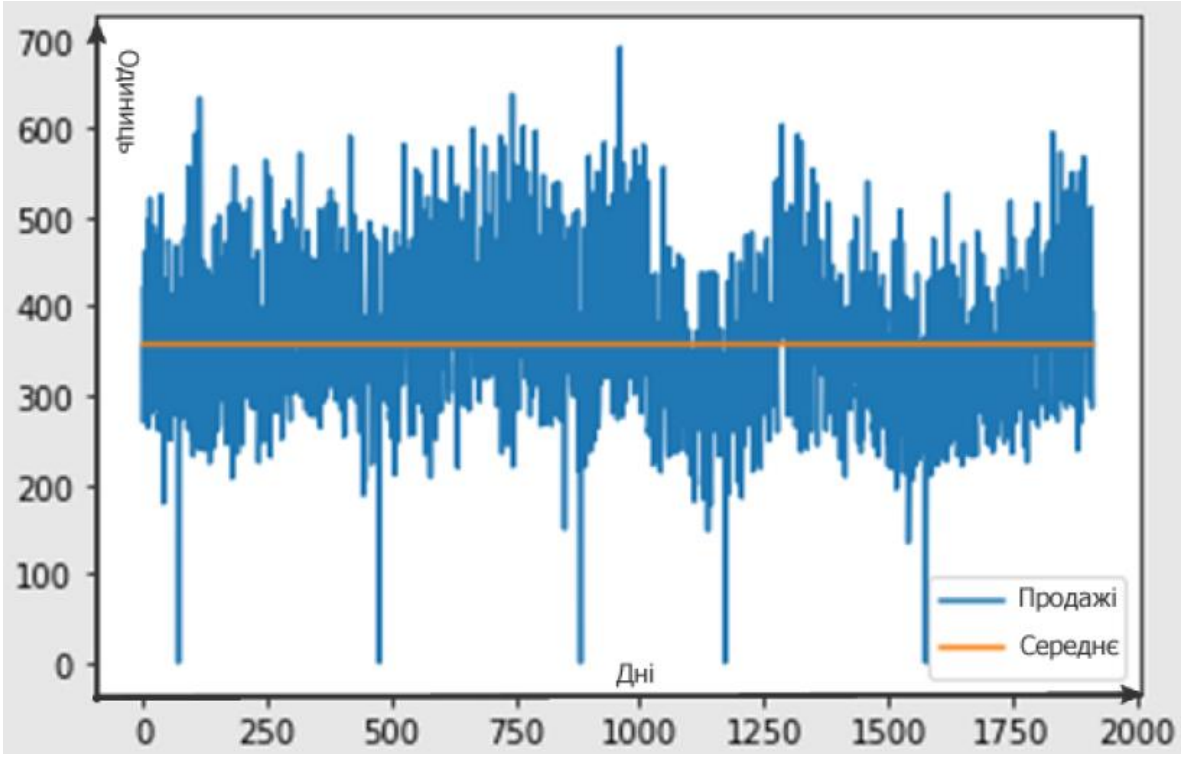


Рис. 1.2. Прогнозування середнім значенням

Ніщо не вбиває прогрес (або репутацію) швидше, ніж продажі за кілька тижнів. Належне прогнозування попиту та контроль запасів можуть допомогти упевнитися, що підприємство не купує недостатній або надлишковий запас товарів.

Прогнозування попиту може допомогти вам витрачати менше грошей як на замовлення на закупівлю запасів, так і на складування, оскільки чим більше запасів ви маєте, тим дорожче їх зберігати. Хороше управління запасами передбачає наявність достатньої кількості товару під рукою, але не надто багато. Уважне відстеження рівнів запасів дозволяє легко поповнювати запаси та прогнозувати їх з часом. Приклад з продажами товару, який неправильно спрогнозували й не поповнили склад на рис.1.3.

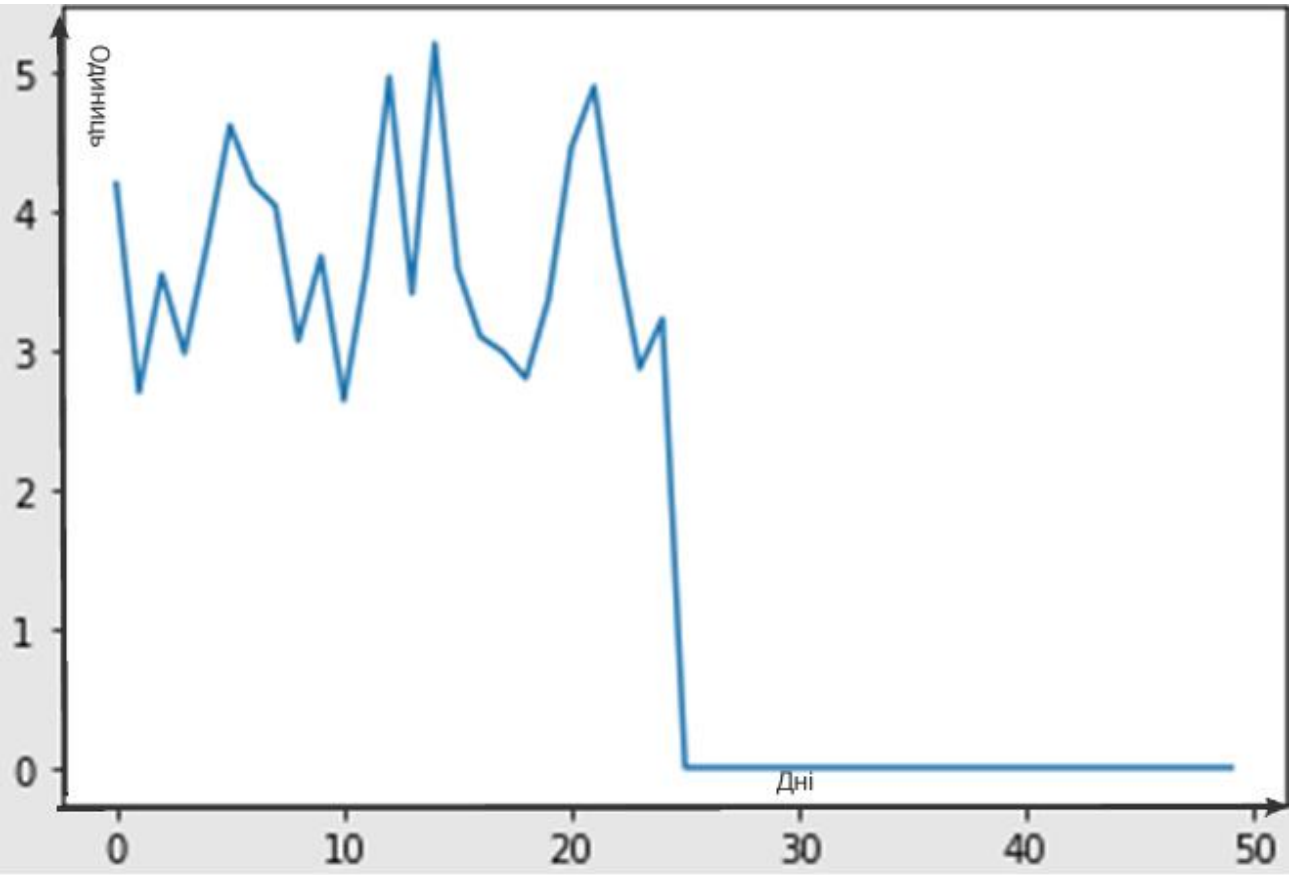


Рис. 1.3. Нульові продажі через відсутність товару на складі

Прогнозування попиту – це не тільки вдосконалення виробничого графіка підприємства, щоб задовольнити попит, але воно також має допомогти оцінювати

продукти на основі попиту. Розуміючи ринок і потенційні можливості, підприємства можуть розвиватися, формулювати конкурентоспроможні ціни, застосовувати правильні маркетингові стратегії та інвестувати в своє зростання.

Якщо вирішити знизити ціни або поставити товар на акцію, попит на цей продукт може тимчасово зрости. Без цього продажу ви, можливо, не відчули б підвищення. Приклад знижки для товару на шостому тижні продажів проілюстровано на рис. 1.4.

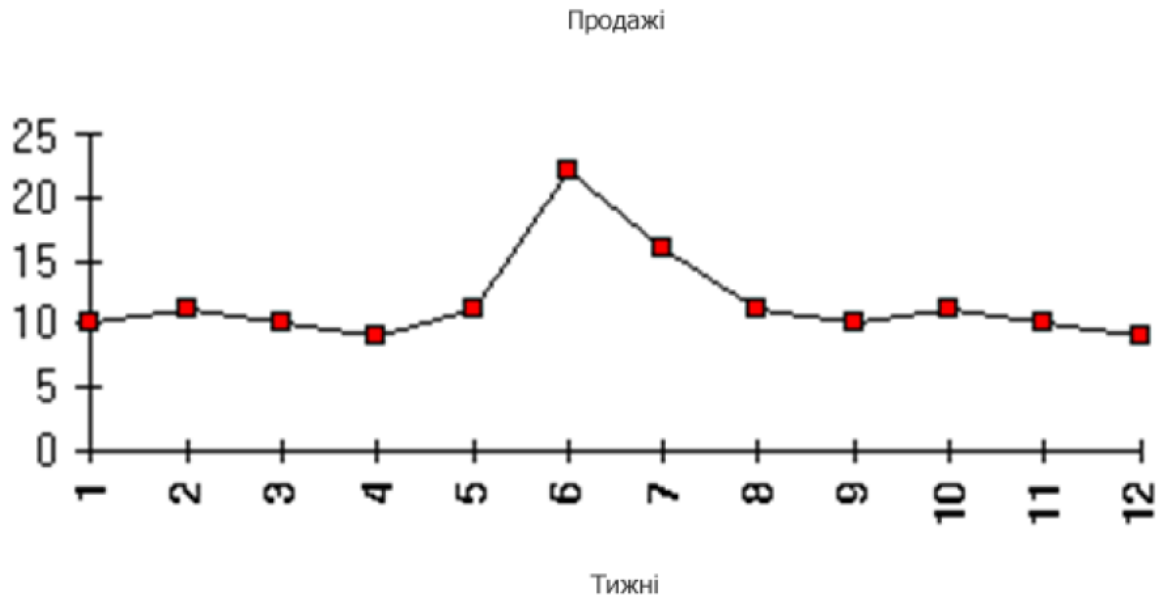


Рис. 1.4. Знижка на шостому тижні продажів

Якщо пропозиція високого попиту обмежена, ви можете використовувати принцип дефіциту для підвищення ціни як ексклюзивної пропозиції. Ви повинні стежити за новими учасниками, хоча пропозиція може збільшитися.

Малий бізнес може мати консервативний план зростання, в той час як інша компанія може масштабувати або диверсифікуватися за допомогою агресивних планів зростання.

1.1.2 Приклади прогнозування попиту

Наведені нижче приклади прогнозування попиту описують кілька різних сценаріїв:

- У продуктовому магазині розглядаються тенденції продажів за минулорічний тиждень подяки, щоб підготувати достатній рівень запасів для майбутнього сезону. Вони розглядають продажі сезонних продуктів, таких як індичка, журавлина та картопляне пюре, які почалися на цьому тижні минулого року. Для них це був чудовий святковий розпродаж. Але вісім місяців тому за чотири квартали від нього відкрився конкуруючий продуктовий магазин, тому вони не впевнені, як це вплине на попит у День подяки та чи будуть місцеві клієнти купувати інгредієнти у свого конкурента. У той же час багато сімей продовжують переїжджати в цей район, і вони все ще зростали в середньому на 1% щомісяця з моменту відкриття конкуруючої мережі. Вони планують запустити більше реклами, ніж минулого року, через канали, які в минулому підтвердили хорошу рентабельність інвестицій, а також запропонують декілька нових пропозицій, щоб позиціонувати себе як місце для Дня подяки. Їхні розрахунки прогнозують зростання продажів на 5% порівняно з минулим роком. Графік з реальними продажами на День подяки та Чорну п’ятницю зображено на рис.1.5.

Рис. 1.5. Реальні продажі на День подяки та Чорну п’ятницю

- Перспективний бренд косметики для споживачів швидко розвивається. Зараз вони продають 10 000 замовлень на місяць. На основі їхніх минулих даних про продажі, майбутніх рекламних кампаній та загальних ринкових умов у галузі, вони планують перевищити 30 000 замовлень на місяць у цей час наступного року. Наразі вони мають загальну кількість 75 000 одиниць на різних рівнях у своїх 5 артикулах. Їх обсяг замовлень коливається залежно від їхнього циклу поповнення, і вони поповнюють запаси за рівнем SKU приблизно кожні 90 днів. Середні одиниці, які вони зберігають, швидко зростуть, а частота каденції залишиться незмінною. Останній випуск їх основного артикула становив 30 000 одиниць. Вони збираються відправити ще 50 000 одиниць, а їх наступний тираж буде 75 000 одиниць. Вони планують і надалі розвиватися такими темпами, тому розглядають, чи варто купувати землю та орендувати склад, щоб не відставати від попиту.

1.1.3 Типи прогнозування попиту

Підприємства можуть прогнозувати попит різними способами (рис. 1.6.).

Рис. 1.6. Типи прогнозування попиту

Усі моделі прогнозування використовують дані й аналітику за певні періоди часу:

- Прогнозування попиту на макрорівні розглядає загальні економічні умови, зовнішні сили та інші загальні речі, які порушують торгівлю. Ці фактори дозволяють компанії бути в курсі можливостей розширення портфеля, інформації про дослідження ринку та інших змін на ринку.
- Прогнозування попиту на мікрорівні може бути специфічним для певної галузі, бізнесу або сегмента клієнтів (наприклад, вивчення попиту на натуральний дезодорант для клієнтів у Чикаго, штат Іллінойс).
- Короткострокове прогнозування попиту зазвичай робиться на період менше 12 місяців. Він розглядає попит менше ніж за рік продажів, щоб інформувати щоденно (наприклад, планувати потреби виробництва для акції «Чорна п’ятниця/Кіберпонеділок»).
- Довгострокове прогнозування попиту здійснюється на термін більше року. Це допомагає визначити та спланувати сезонність, річні закономірності, виробничі потужності та розширення на більш тривалий період часу. Це стимулює довгострокову бізнес-стратегію (наприклад, плани запустити об’єкт або магазин на міжнародному рівні та вийти на нові ринки).

1.1.4 Основні фактори у прогнозуванні попиту

Прогнозування попиту – це синтез де ланцюжок поставок бізнесу зустрічається з продажами та маркетингом.

Щоб досягти успіху, обидві сторони мають бути синхронізовані. Загалом є декілька основних факторів, що впливають на попит:

- Сезонність — це зміни обсягу замовлення протягом певного періоду часу. Високосезонний бренд може обслуговувати певний період часу, події чи сезону, що спричиняє різний рівень попиту протягом року, включаючи значні стрибки під час пікового сезону. Сезонний попит часто вимагає від бізнесу скоротити запаси в наявності протягом

тихих місяців, а потім збільшити виробництво та робочу силу під час пікового сезону. Ось чому багато циклічних компаній передають роздрібні послуги сторонній логістичній компанії, яка може зберігати запаси, вибирати предмети, упаковувати коробки та відправляти для них замовлення. Приклад сезонності на рис. 1.7.

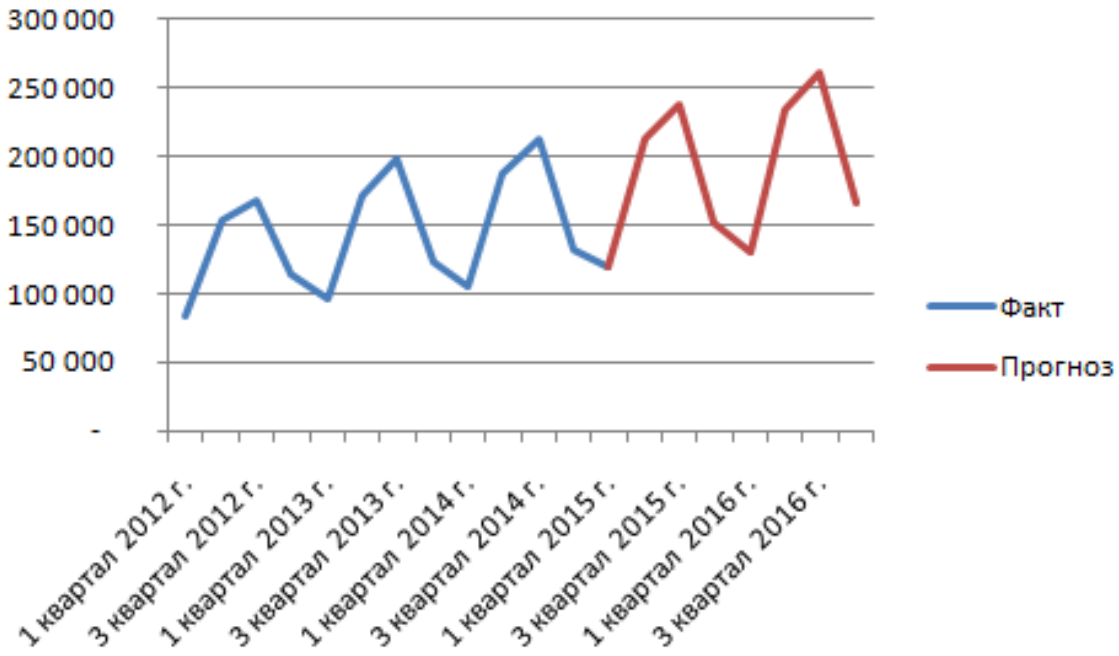


Рис. 1.7. Сезонність попиту

- Конкуренція впливає на попит, оскільки у клієнтів є більше варіантів на вибір, та багато компаній борються за їхню увагу. Коли в гру вступає конкурентна сила — чи то прямий конкурент, чи новий тип рішення, який змушує клієнта обирати між одним чи іншим — попит буде перекошеним. Це може здивувати, тому гнучка модель прогнозування попиту може допомогти швидко реагувати.
- Тип товарів. Прогнозування попиту буде дуже відрізнятися для різних продуктів і послуг — від швидкопсувних товарів, термін придатності яких швидко закінчується, до передплатних коробок, які надходять щомісяця в один і той же час. Важливо знати цінність клієнтів за весь час (загальна кількість покупок, які вони купують у різних каналах за

певний час), середню вартість замовлення (скільки вони витрачають кожен раз) і комбінації продуктів, замовлені для покращення прогнозування попиту. Використовуючи ці дані, можна зрозуміти, як згрупувати або об'єднати товари, збільшити постійний дохід і побачити, як один артикул впливає або стимулює попит на інший (наприклад, продажі заправних картриджів для бритви та леза).

- Географія, де проживають клієнти, а також звідки виготовляєте та відправляєте замовлення, може сильно вплинути на прогнозування запасів і швидкість, з якою можна виконувати замовлення клієнтів. Географічне розташування роздрібної мережі поставок може бути дуже стратегічним. Використання центрів виконання в місцях поблизу клієнтів може допомогти задовольнити попит клієнтів швидше та дешевше, тому доставка здійснюється з найближчого до клієнта складу. Це допомагає відстежувати, де проживають клієнти, і зберігати певні продукти в регіонах, де їх замовляють найбільше, тож не доведеться відправляти у віддалені місця.

Всі ці фактори є важливою складовою точних прогнозів та ефективної роботи системи постачання (supply chain). У комерції управління ланцюгом поставок (SCM) – це управління потоком товарів і послуг між підприємствами та місцями, а також включає переміщення та зберігання сировини, незавершеного виробництва та готової продукції, разом з виконанням замовлень від точки походження до точки споживання. Взаємопов'язані мережі, канали та вузли об'єднуються в наданні продуктів і послуг, необхідних кінцевим клієнтам у ланцюжку поставок. Управління ланцюгом поставок було визначено як проектування, планування, виконання, контроль та моніторинг діяльності ланцюга поставок з метою створення чистої вартості, створення конкурентоспроможної інфраструктури, використання логістики в усьому світі, синхронізації пропозиції з попитом та вимірювання ефективності в усьому світі. Умовна схема системи поставок зображена на рис. 1.8.

Рис. 1.8. Система постачання

1.1.5 Загальний опис процесу прогнозування попиту

Прогнозування попиту є надзвичайно складним завданням. Потрібно бути не тільки достатньо гнучкими, щоб впоратися з підвищеним або відсутнім попитом, але також використовувати довгостроковий підхід.

Для прогнозування попиту необхідно мати точну мету. По суті, це передбачає, що, скільки і коли куплять клієнти. Період часу, конкретний продукт або загальну категорію, яку переглядають. Потрібно переконатися, що це задовольнить спеціалістів з фінансового планування, маркетингу продуктів, логістики та операційних команд неупереджено. Потрібно зрозуміти цілі для правильного планування потужності попиту, що дозволить використовувати процеси прогнозування прийняття рішень, щоб краще зрозуміти поведінку споживачів.

Інтеграція всіх даних з каналів продажів може забезпечити цілісне уявлення про фактичний попит на продукцію та уявлення про прогнози продажів. Можливість бачити час і дату замовлень, замовлені артикули та канали продажів допоможе спрогнозувати зростання і прогнозувати тенденції на більш детальному рівні та оглянутися назад, щоб побачити, як прогнози відповідають дійсності. Також необхідно звернути увагу на прибуток від електронної комерції. Продукти з високим рівнем повернення слід оцінювати та коригувати. Якщо повертається 10% товарів, і можна зменшити цю кількість, можливо, також знадобиться відкоригувати виробництво. Крім історичних даних про продажі, також може знадобитися отримати інші дані, наприклад, ринкові умови. Для забезпечення надійності та точності - дані повинні бути належним чином підготовлені.

Незалежно від того, чи виконується це вручну або з використанням автоматизації та прогнозної аналітики, потрібен повторюваний процес аналізу даних. Для цього потрібно порівняти те, що передбачили, з реальними продажами, щоб допомогти адаптувати наступний прогноз. На діаграмі (рис. 1.9.) показано чотирьох різних клієнтів на одній часовій шкалі, які всі відправили 60 000 замовлень за один рік. Вимірювання цього допомагає відстежувати попит на різні продукти в різний час. Хоча кожен із них відправляє в середньому 5000 замовлень на місяць, деякі місяці набагато легші за інші. Якби бренди занижували цей обсяг, у них не було б достатньої кількості запасів для відвантаження замовлень і не вистачило б персоналу, щоб виконати всі замовлення вчасно. Якби вони завищили обсяги, вони витратили б багато грошей на запаси, які просто стоять і займають набагато більше часу, ніж очікувалося, для отримання доходу. У міру зростання можна виявити, що потрібно почати відстежувати додаткову інформацію, таку як застарілі запаси, частота відсутності та інші деталі замовлення.

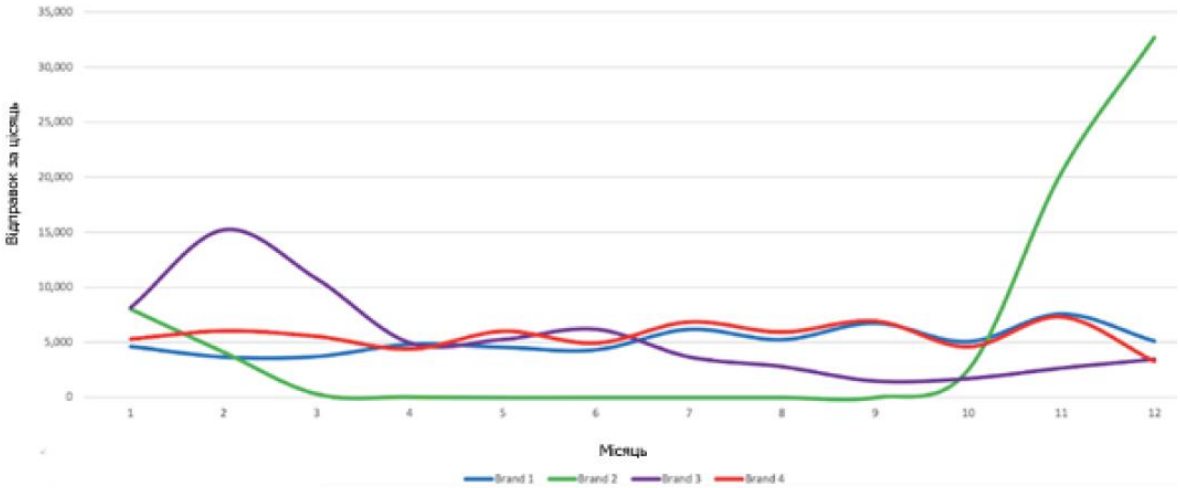


Рис. 1.9. Діаграма чотирьох клієнтів на одній часовій шкалі

Коли є цикл зворотного зв’язку, можна встановити наступний прогноз і оновити бюджет, щоб розподілити кошти, куди вони мають спрямовуватися, на основі цілей зростання. Прогнозування попиту допомагає зменшити витрати на зберігання запасів, планувати витрати на маркетинг, майбутню чисельність персоналу, потреби у виробництві та запасах і навіть нові продукти [1].

Також складності виникають, коли дані про історичні продажі не якісно зібрані або продажі товарів по своїй природі формують дуже хаотичний та складно прогнозуючий часовий ряд. Приклад погано прогнозуючого товару зображено на рис. 1.10.

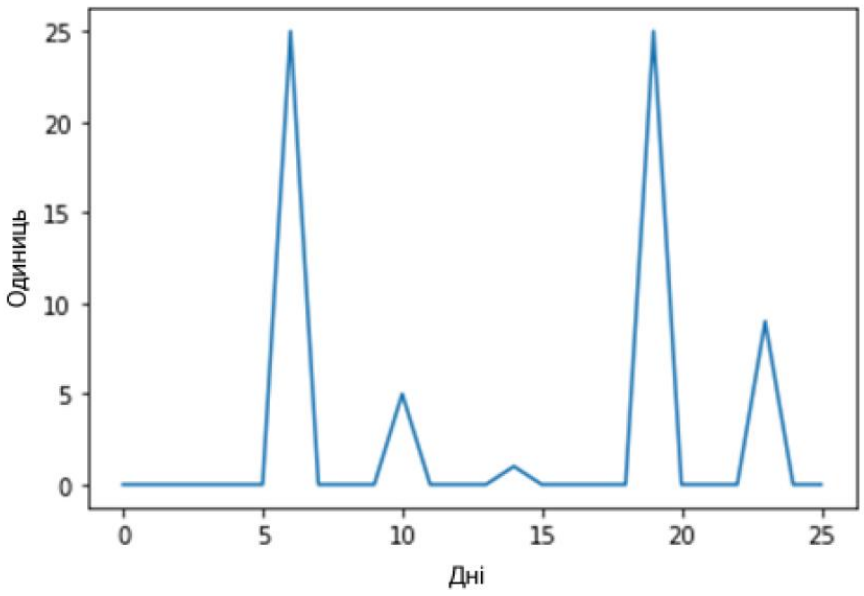


Рис. 1.10. Приклад погано прогнозуючого товару

1.2 Продуктова канібалізація

1.2.1 Основні поняття

Сприяння продажу товару, наприклад, за допомогою знижок на ціни, реклами та/або спеціальних показів, часто має величезний вплив на його продажі. Успішне виконання стимулювання збуту можливе тоді і тільки тоді, коли збільшення обсягу продажів враховується на всіх фазах ланцюга поставок. При правильному плануванні ланцюга поставок, керуючись прогнозами просування, можна задовольнити підвищений попит на рекламований продукт без псування або надлишку запасів. Однак стимулювання збуту одного товару може додатково мати значний вторинний вплив на продажі інших продуктів, які не входять до рекламної діяльності – факт, про який часто забувають або не звертають уваги. Нехтування вторинними ефектами призводить до неоптимального планування і, отже, не дозволяє досягти повного потенціалу прибутку від стимулювання збуту (рис. 1.11.).

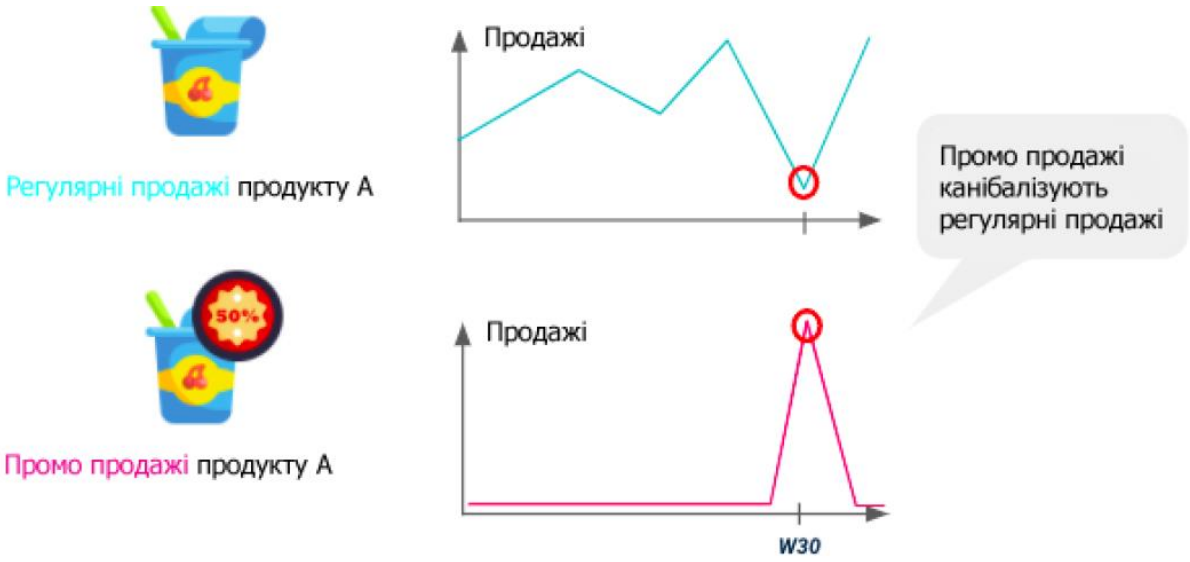


Рис. 1.11. Промо-продажі та їх канібалізація на регулярні продажі

Уявімо, що покупець стоїть перед шафою зі свіжим м’ясом у звичайному супермаркеті з нотатником. Наступний пункт у списку покупок: 2 кілограми яловичого фаршу. Дивлячись на шафу, він бачить щонайменше 15 варіантів

різних брендів із різним вмістом жиру, які відповідають невизначеним критеріям у моєму списку покупок. Деякі з варіантів є органічними, а деякі з них були легко сформовані в котлети для гамбургерів і приправлені. Чи варто вибрати ту саму марку, яку покупець використовував минулого разу, чи спробувати щось інше в надії приготувати смачніші страви? Але на полиці є продукт X і він має знижку 20%, тому скоріш за все клієнт вирішить заощадити трохи грошей та купити його.

Легко придумати приклади того, як стимулювання збуту впливає на нашу поведінку при покупці. Якщо два продукти майже ідеально замінюють один одного, а один із товарів має значну знижку через постійне стимулювання збуту, зазвичай рекламований продукт потрапляє в кошик покупця замість того, що продається в магазині за стандартною ціною. Отже, стимулювання збуту одного товару зменшує або знищує продажі подібних товарів. Як абсолютно протилежне явище, деякі продукти, як-от вода з джином і тоніком, часто купуються разом, і, отже, реклама одного продукту також може збільшити продажі його добавки. Це, у свою чергу, називається ефектом halo (ореолу).

Щоб врахувати ефекти канібалізації та ореолу при прогнозуванні попиту, вигідно з точки зору ефективності спочатку визнати відповідні відносини з, ймовірно, мільйонів можливостей. Найпростішим, але не дуже практичним способом було б покластися на здоровий глузд і перерахувати відносини вручну. Яловичий фарш з низьким вмістом жиру, ймовірно, канібалізує подібні продукти, але чи він також канібалізує стандартний яловичий фарш? Визначити всі релевантні зв’язки за допомогою здорового глузду не так просто, як може здатися на перший погляд, а крім того, вручну створювати та підтримувати список для тисяч продуктів зовсім неможливо. Отже, єдиним практичним варіантом є виявлення зв’язків на основі історичних даних за допомогою методів машинного навчання.

У роздрібному бізнесі для цієї мети часто доступні два корисних набори даних. Перший варіант полягає в тому, щоб ідентифікувати замінники продуктів і продукти, які часто купують разом, на основі транзакцій на рівні чеку та даних картки лояльності, використовуючи, наприклад, навчання правил асоціації.

Наприклад, якщо два продукти ніколи не входять до одного кошика покупок або якщо переваги одного покупця довільно різняться між двома подібними продуктами, існує висока ймовірність того, що продукти насправді є замінниками і, швидше за все, вступають у канібалізаційні стосунки один з одним. Той самий метод легко застосувати і для відносин ореолу: якщо два продукти купуються разом частіше, ніж можна було б очікувати, якби покупки були повністю незалежними, продукти, ймовірно, перебувають у відносинах ореолу.

Другий варіант — дані часових рядів на рівні SKU-магазину про продажі. Ці дані також відображають взаємозв'язки, і, зрештою, майбутній попит – це також величина, яку ми зацікавлені спрогнозувати. Якщо ми розглянемо систему, що складається лише з однієї пари продуктів, протягом звичайних періодів розпродажів, тобто поза акцій, і з обома продуктами, наявними на складі, система перебуває в певній рівновазі. Обидва продукти купуються дещо випадково, і на пропорційні продажі кожного товару впливають різні фактори, наприклад бренд, ціна. Важко, якщо взагалі неможливо, зробити якісь корисні висновки щодо канібалізації чи ореолів стосунків лише на основі рівноважних продажів. Якщо два продукти перебувають у стосунках канібалізації, стимулювання збуту першого продукту має збільшити його продажі, але в той же час зменшити продаж другого товару порівняно з рівноважним. Таким чином, вплив канібалізаційного просування на продажі двох продуктів негативно корелює. З іншого боку, ефект ореолу призведе до позитивної кореляції, тобто якщо один продукт рекламується, продажі додаткового продукту також збільшуються. Аналіз сили та значущості цих кореляцій виявляється надійним і відносно ефективним способом виявлення зв'язків канібалізації з великої маси даних про продажі. Важливою перевагою є те, що ця методика надає легко піддану кількісній оцінці інформацію про міцність і релевантність відносин, що полегшує фільтрацію неважливих відносин [2].

1.2.2 Товари новинки

Ймовірно, новий продукт «вбиває» свого попередника, бо значна кількість продажів припадає на тих самих покупців, що купували старий товар-аналог. Помилятися можуть навіть компанії-лідери з великим штатом аналітиків та бізнес-консультантів. Наприклад, компанія "Вімм-Білль-Данн" на початку 2000-х років зіткнулася з проблемою через надмірне поділ ринку соків на сегменти. Спочатку, коли асортимент компанії складався з марок J7, Rio Grande та 100old Premium, проблем не було. Але як тільки компанія додала бренд "Улюблений сад", почалося витіснення одних марок іншими, оскільки всі соки практично не відрізнялися за ціною.

Канібаліацію не завжди розглядають у негативному світлі. Деякі компанії вдаються до неї навмисно і відносять її до корисних маркетингових інструментів. Новий продукт може бути більш прибутковим. Припустимо, компанія виробляє пакети для сміття та виводить новинку – екопакети. Не біда, що частина тих споживачів, які купували звичайні пакети, тепер переключаться на ековерсію продукту, оскільки ті, хто дбає про стан навколишнього середовища, зазвичай готові купувати екопродукти дорожче.

Темп інновацій у галузі такий високий, що компанія розуміє: якщо вона не випустить новинку, то це зроблять її конкуренти. Це типова ситуація для виробників мобільних телефонів та комп'ютерної техніки. Класичний приклад - Apple, коли вона виводила на ринок планшет iPad. Це призвело до падіння продажу ноутбуків компанії, але при цьому відкрило новий ринок.

Потрібно прискорити вихід із ринку продукту, виробництво якого компанія збирається припинити. Наприклад, продавцю техніки може бути простіше запропонувати своїм існуючим клієнтам схему trade-in, ніж ще років десять випускати запчастини до старої моделі обладнання та підтримувати систему його сервісу. Компанія вірить, що новий продукт буде продаватися краще за існуюче, тому що змінилася структура попиту.

Виводячи потенційну новинку з ефектом канібалізації, компанія сподівається завдати більшої шкоди своєму конкурентові. Такі бренди називають брендами-винищувачами. Наприклад, в Індії, яка з початку 2000-х переживає бум попиту на легкові авто, компанія Maruti-Suzuki випустила на ринок нову модель Suzuki Alto у тому ж сегменті, що й інший її автомобіль – Maruti 800. Мета – не дозволити вийти в сегмент малолітражок конкуренту - Hyundai.

Запуск бренду-винищувача – дуже ризикована стратегія, оскільки в питаннях конкурентних воєн ніколи не можна бути впевненим, які наслідки матиме той чи інший крок. У будь-якому разі, до негативного сценарію фінансового плану компанії слід вносити можливе зниження продажу існуючих продуктів. А оцінюючи успіх нового продукту, варто обов'язково коригувати доходну частину проекту на величину зниження доходу від свого продукту, чию частину ви «відкусили». Хоча слід зазначити, такі підрахунки є дуже приблизними.

Існує окремий вид канібалізації – радикальна. Вона означає заміну своїх успішних товарів фундаментально новими. Стратегія розрахована на те, що конкурентам просто не вистачить знань, досвіду та ресурсів, щоби змагатися з вами. Радикальна канібілізація призводить до збільшення частки ринку, а також його зростання за рахунок представленої інновації.

Типи товарів новинок зображені на рис. 1.12.

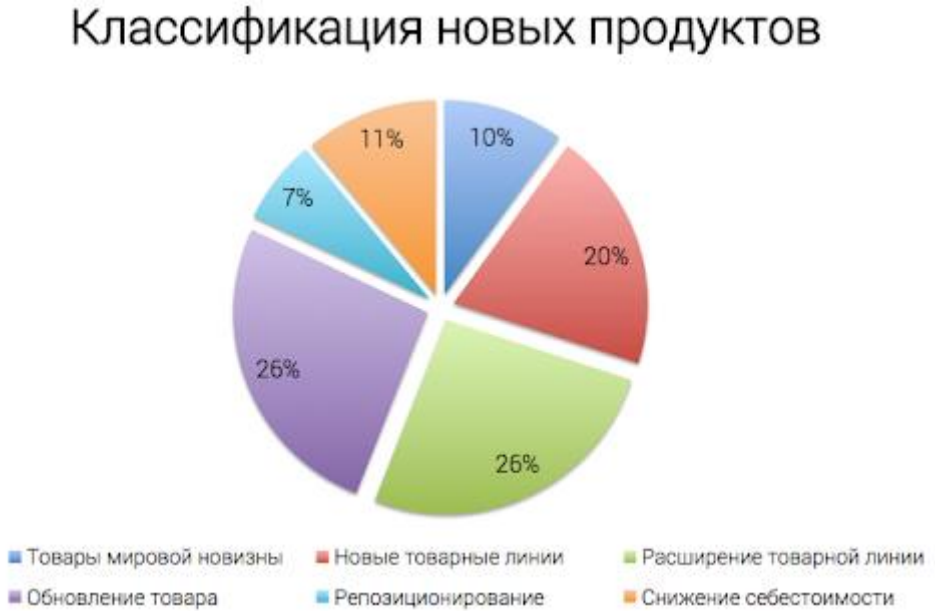


Рис. 1.12. Типи товарів новинок

1.2.3 Заключення

Після того, як відповідні зв’язки відомі, динамічні ефекти рекламних акцій можна ввести в прогнозний розрахунок, використовуючи аналогічний підхід, як і при розрахунку звичайних рекламних ефектів. У той час як звичайні рекламні заходи, як правило, збільшують продажі рекламованого продукту, ефект ореолу збільшує продажі всіх його добавок протягом того самого періоду реклами. Таким чином, ефект ореолу можна просто розглядати як особливий вид просування, який активний, коли родичі ореолу продукту проводять стимулювання збуту. Однак слід мати на увазі, що ефект ореолу слабший, ніж ефект оригінальної кампанії, якщо продукти не завжди купуються разом. Відповідно, канібалізація матеріалізується як спеціальні акції з негативним впливом на продажі. Як видно на рис. 1.13., зазвичай первинний ефект просування на продажі рекламованого продукту є набагато більш помітним, ніж вторинний ефект канібалізації, який походить від рекламної діяльності продукту-замінника. Реклама подібних продуктів (червоні періоди) знижує продажі цього продукту з замороженої картоплі та зменшує продажі на 10-25 % порівняно з наступними тижнями. Однак ефект канібалізму значно слабший, ніж ефект власних рекламних акцій продукту (синім кольором).

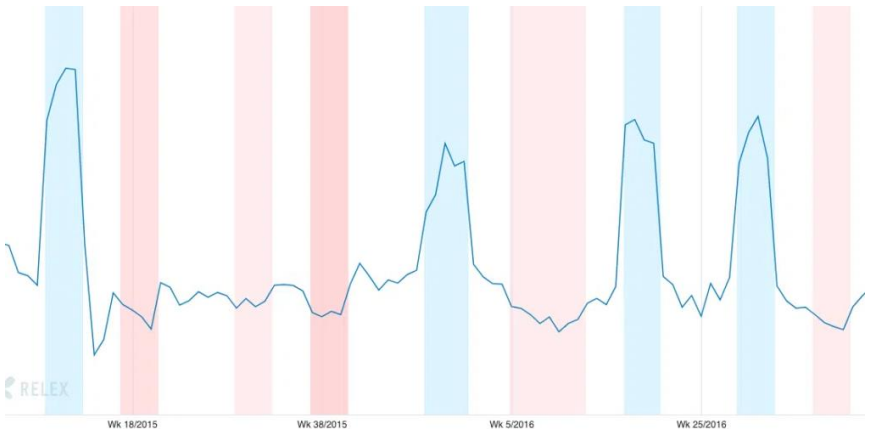


Рис. 1.13. Канібалізація та первинний ефект промо акцій (Relex)

1.3 Метрики

У даній роботі для оцінювання точності прогнозування попиту використовується чотири метрики: MAE, MSE, MAPE та R-Squared. Далі у підрозділах детально описано алгоритми розрахунків даних метрик.

1.3.1 MAE

MAE (Mean Absolute Error) - вимірює середню величину помилок у наборі передбачень, не враховуючи їх напрямок. Це середнє за тестовою вибіркою абсолютних відмінностей між прогнозом і фактичним спостереженням, де всі індивідуальні відмінності мають однакову вагу

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n} |y_j - \widehat{y_J}| \qquad (1.1)$$

де n – це кількість передбачень, y_j- прогнозне значення, $\widehat{y_J}$ - реальне значення.

Якщо абсолютне значення не береться (ознаки помилок не видаляються), середня похибка перетворюється на середню помилку зміщення (MBE) і зазвичай призначена для вимірювання середнього зміщення моделі. MBE може передавати корисну інформацію, але її слід тлумачити обережно, оскільки позитивні та негативні помилки зникнуть.

MAE виражає середню помилку прогнозування моделі в одиницях змінної. Метрики можуть коливатися від 0 до ∞ і байдужі до напрямку помилок. Чим нижче значення метрики тим більш точний прогноз [3].

Візуально метрика продемонстрована на рис. 1.14., де вісь X – це незалежна змінна, а вісь У – залежна. Пряма на графіку – прогноз. Сині точки на графіку –

реальні значення. У такому випадку MAE – це середнє значення сум абсолютних відстаней від синіх точок до прямої.

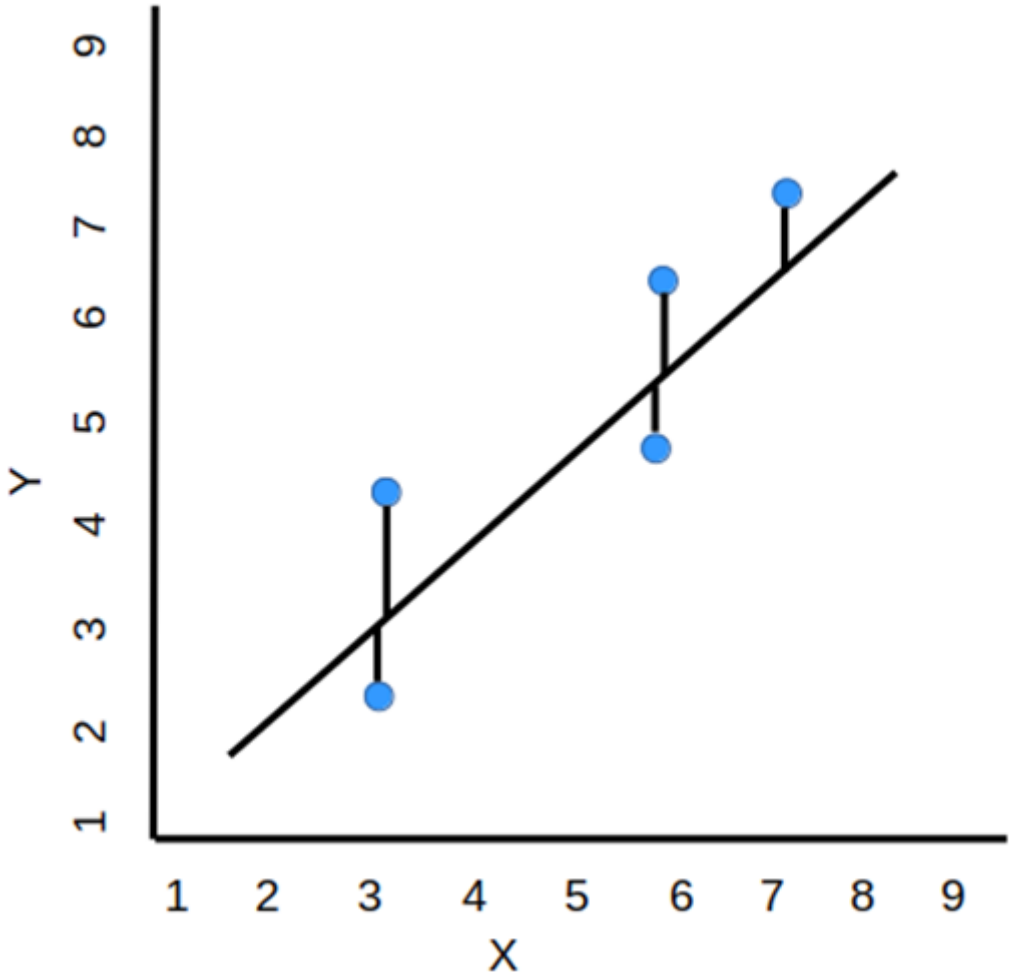


Рис. 1.14. MAE на графіку

1.3.2 MSE

MSE (Mean Squared Error) – середнє значення квадратів похибок. Алгоритм розрахунку та інтерпретація похибки дуже схожа на MAE. Середня квадратична помилка говорить вам, наскільки близька лінія регресії до набору точок. Це робиться, беручи відстані від точок до лінії регресії (ці відстані є «помилками») і зводять їх у квадрат. Зведення в квадрат необхідне, щоб прибрати негативні ознаки. Це також надає більшої ваги більшим відмінностям. Це називається середньоквадратичною помилкою, оскільки ви знаходите середнє значення набору помилок. Чим нижчий MSE, тим кращий прогноз [4].

$$\text{MSE} = \frac{1}{n}\sum_{j=1}^{n}(y_j - \widehat{y_J}) \text{ ^2} \quad (1.2)$$

де n – це кількість передбачень, y_j- прогнозне значення, $\widehat{y_J}$ - реальне значення.

Візуально метрика продемонстрована на рис. 1.15.

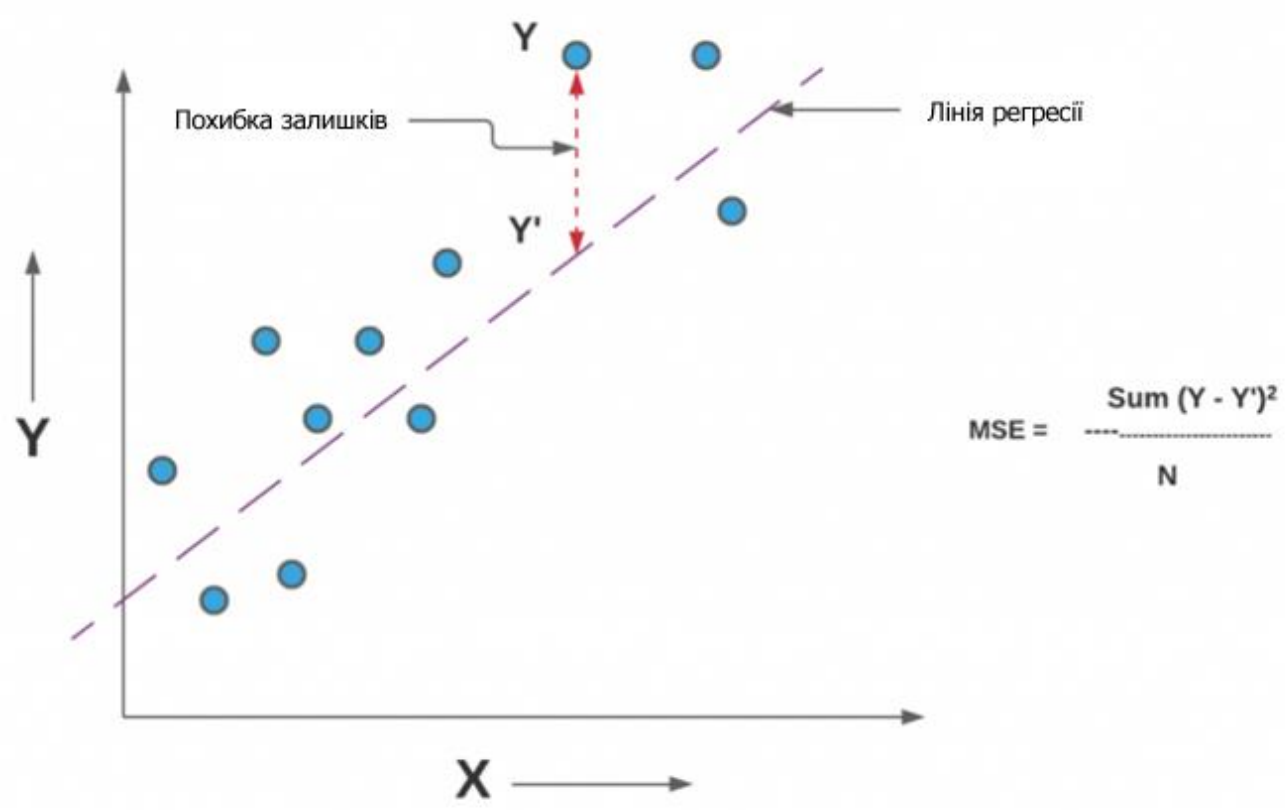


Рис. 1.15. MSE на графіку

1.3.3 MAPE

MAPE (Mean Absolute Percentage Error) - є мірою того, наскільки точною є система прогнозів. Він вимірює цю точність у відсотках і може бути розрахований як середня абсолютна відсоткова помилка для кожного періоду часу мінус фактичні значення, поділені на фактичні значення

$$\text{MAPE} = \frac{1}{\text{n}} \sum_{\text{j}=1}^{\text{n}} \left| \frac{A_t - F_t}{A_t} \right| \quad (1.3)$$

де n – це кількість передбачень, F_t - прогнозне значення, A_t - реальне значення.

Середня абсолютна відсоткова помилка є найпоширенішою мірою, яка використовується для прогнозування помилки, і найкраще працює, якщо в даних немає екстремумів (і нулів). Тому недоліком даної метрики є те, що вона не може працювати, якщо у реальних значеннях є нулі, тому під час підрахунку необхідно їх або ігнорувати або синтетично підвищувати з нульових значень до більших [5].

1.3.4 R-Squared

R-квадрат є мірою відповідності для моделей регресії. Ця метрика вказує на відсоток дисперсії залежної змінної, яку незалежні змінні пояснюють разом. R-квадрат вимірює міцність зв’язку між моделлю та залежною змінною.

Після встановлення моделі лінійної регресії вам потрібно визначити, наскільки добре модель відповідає даним. Чи добре він пояснює фактори залежної змінної?

Статистики стверджують, що регресійна модель добре відповідає даним, якщо відмінності між спостереженнями та прогнозованими значеннями невеликі та неупереджені. Неупередженість у цьому контексті означає, що підібрані значення систематично не є занадто високими або занадто низькими в будь-якому місці спостережень.

Однак, перш ніж оцінювати числові показники відповідності, наприклад R-квадрат, ви повинні оцінити залишкові графіки. Графіки залишків можуть виявити упереджену модель набагато ефективніше, ніж числові результати, відображаючи проблемні закономірності в залишках (рис. 1.16.).

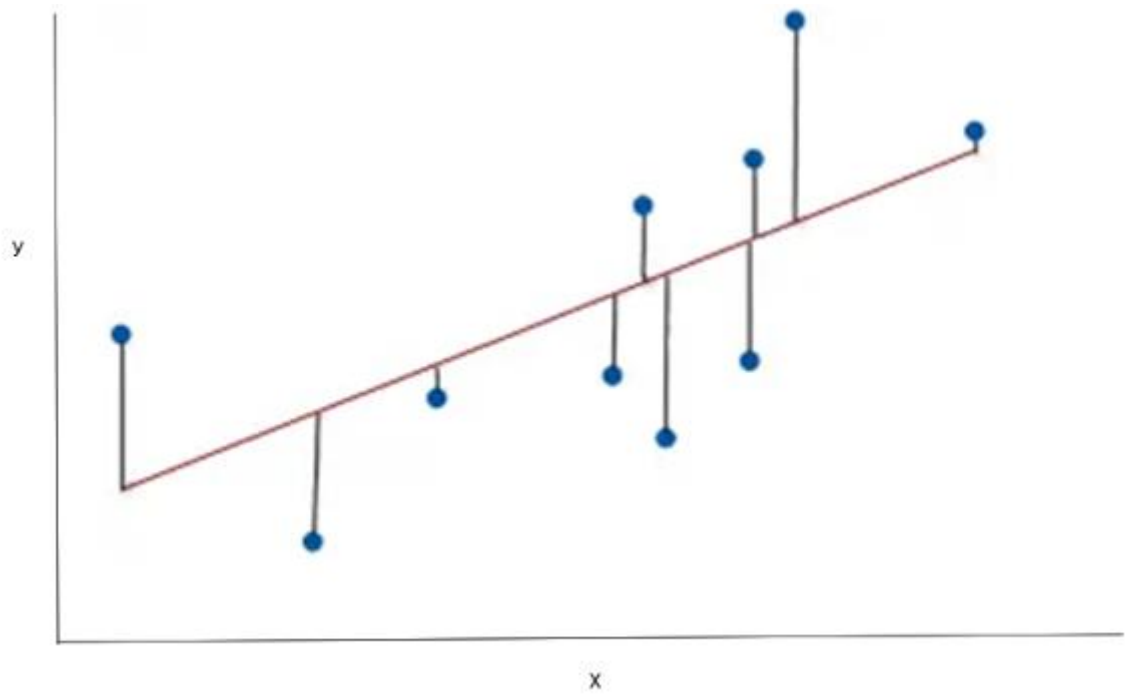


Рис. 1.16. Позитивні залишки – відстань від точок вище лінії, та негативні – відстань від точок нижче за лінію

Якщо ваша модель упереджена, ви не можете довіряти результатам. Якщо ваші залишкові графіки виглядають добре, оцініть свій R-квадрат та інші статистичні дані.

R-квадрат оцінює розкид точок даних навколо підігнаної лінії регресії. Його також називають коефіцієнтом детермінації або коефіцієнтом множинної детермінації для множинної регресії. Для того самого набору даних вищі значення R-квадрата представляють менші відмінності між спостережуваними даними та підібраними значеннями.

R-квадрат – це відсоток варіації залежної змінної, яку пояснює лінійна модель.

$$\mathrm{R-Squared} = \frac{\text{Variance explained by the model}}{\text{Total variance}} \tag{1.4}$$

Зазвичай, чим більше R-квадрат (максимум 100%), тим краще регресійна модель відповідає вашим спостереженням. Однак у цій інструкції є важливі застереження про які буде описано далі.

Щоб наочно продемонструвати, як значення R-квадрат представляють розкид навколо лінії регресії, можна побудувати підігнані значення за спостережуваними значеннями на рис. 1.17 та на рис. 1.18.

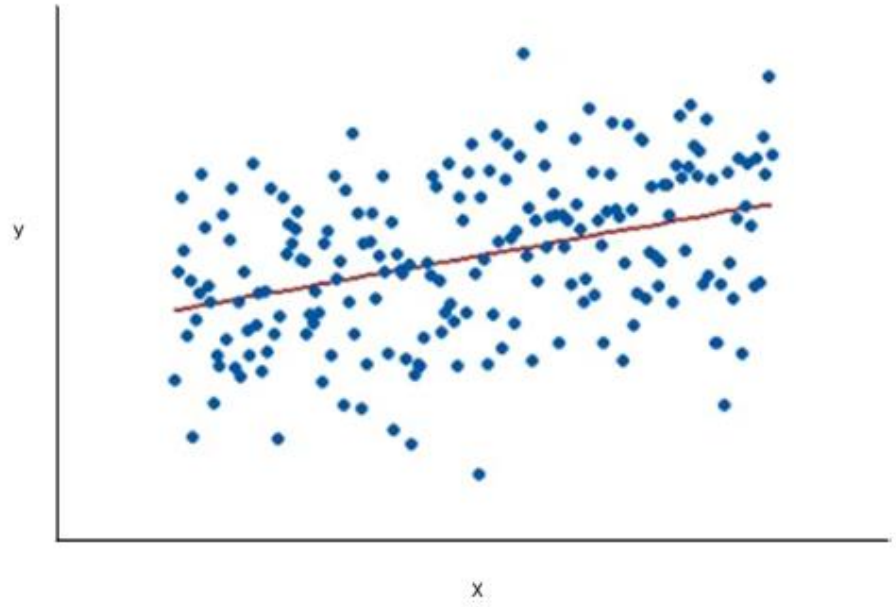


Рис. 1.17. R-squared становить 15%

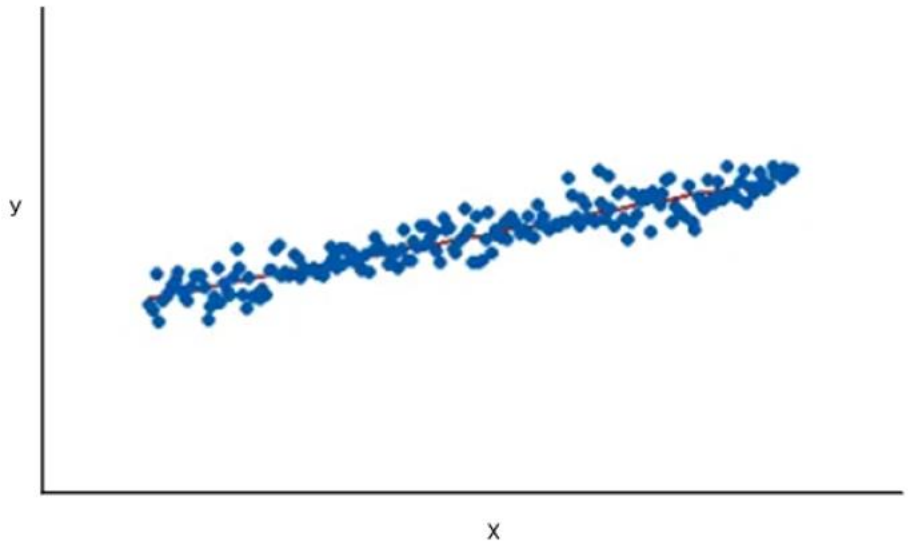


Рис. 1.18. R-squared становить 85%

R-квадрат для регресійної моделі на рис. 1.17. становить 15%, а для моделі на рис. 1.18. — 85%. Коли регресійна модель враховує більшу частину дисперсії, точки даних знаходяться ближче до лінії регресії. На практиці ви ніколи не побачите модель регресії з R2 100%. У цьому випадку підібрані значення дорівнюють значенням даних, і, отже, всі спостереження потрапляють точно на лінію регресії.

Ви не можете використовувати R-квадрат, щоб визначити, чи є оцінки коефіцієнтів і передбачення упередженими, тому ви повинні оцінити залишкові графіки. R-квадрат не вказує, чи відповідає регресійна модель вашим даним. Хороша модель може мати низьке значення R2. З іншого боку, упереджена модель може мати високе значення R2!

Регресійні моделі з низькими значеннями R-квадрата можуть бути ідеально хорошими моделями з кількох причин. Деякі галузі дослідження мають більшу кількість незрозумілих варіацій. У цих областях ваші значення R2 обов’язково будуть нижчими. Наприклад, дослідження, які намагаються пояснити поведінку людини, як правило, мають значення R2 менше ніж 50%. Людей просто важче передбачити, ніж такі речі, як фізичні процеси.

На щастя, якщо у вас низьке значення R-квадрат, але незалежні змінні статистично значущі, ви все одно можете зробити важливі висновки про зв’язки між змінними. Статистично значущі коефіцієнти продовжують являти собою середню зміну залежної змінної за умови зсуву незалежної змінної на одну одиницю. Очевидно, що вміти робити такі висновки дуже важливо.

Існує сценарій, коли малі значення R-квадрат можуть викликати проблеми. Якщо вам потрібно генерувати відносно точні прогнози (вузькі інтервали передбачення), низький R2 може бути зупинкою.

Наскільки високим має бути R-квадрат, щоб модель давала корисні прогнози? Це залежить від потрібної точності та кількості варіацій, наявних у ваших даних. Високий R2 необхідний для точних прогнозів, але він недостатній сам по собі.

Регресійна модель з високим значенням R-квадрат може мати безліч проблем. Очікувано, що високий R2 вказує на хорошу модель, але розглянемо графіки нижче на рис. 1.19. та на рис 1.20. Графік підігнаної лінії моделює зв'язок між рухливістю електронів і щільністю.

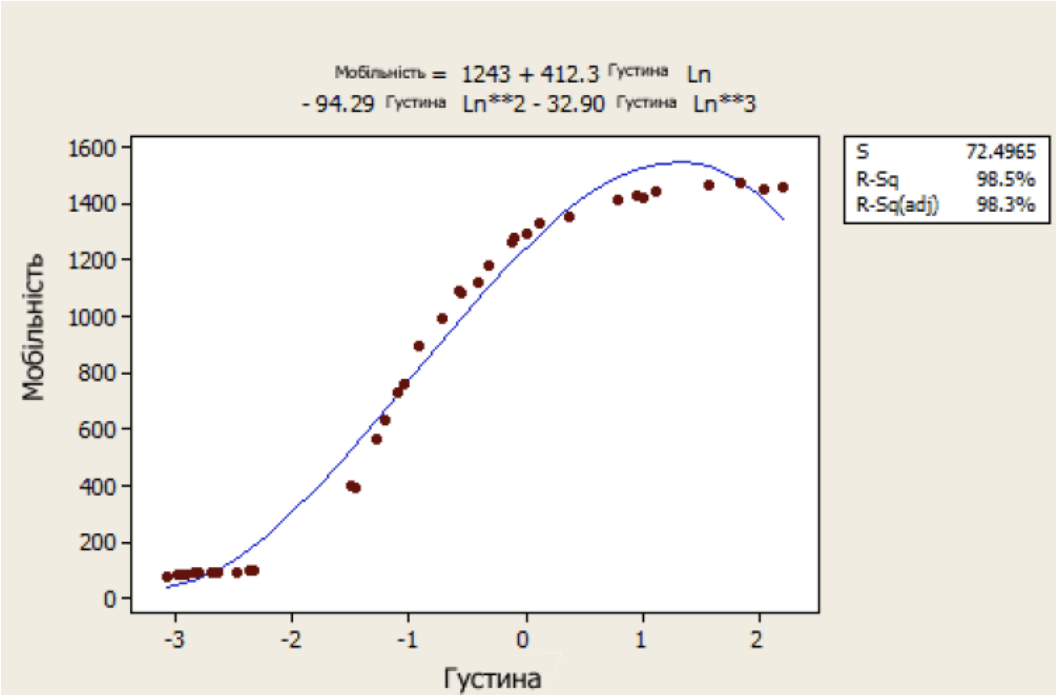


Рис. 1.19. Регресійна модель

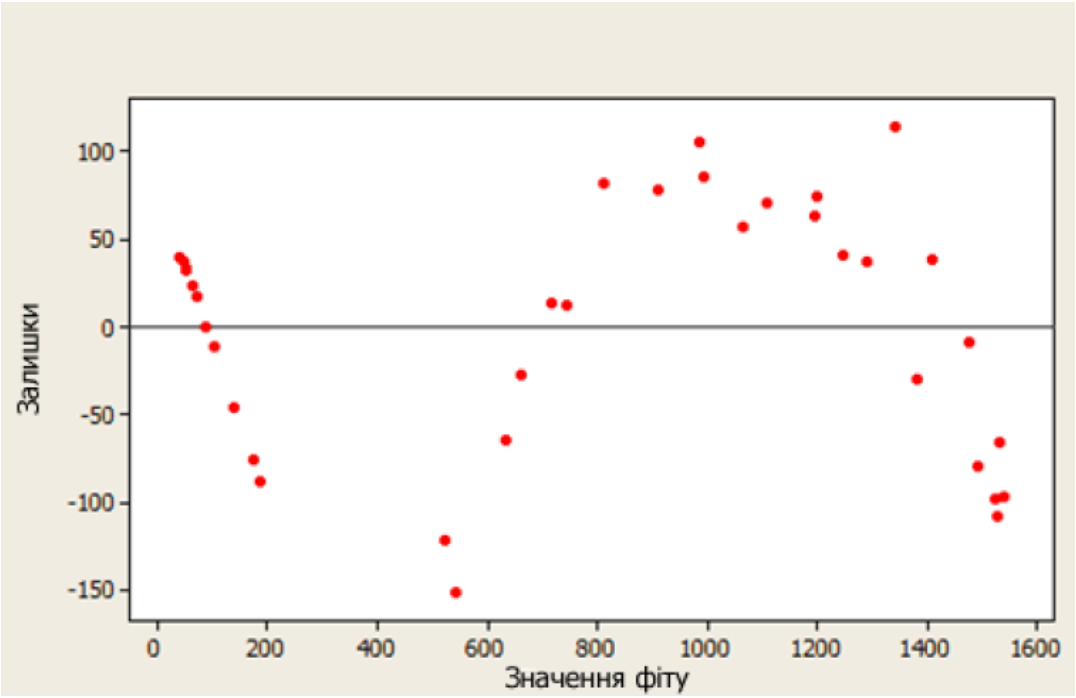


Рис. 1.20. Залишки моделі

Дані на графіку підігнаних ліній мають дуже низький рівень шуму, а R-квадрат становить 98,5%, що здається фантастичним. Однак лінія регресії постійно занижує та завищує дані вздовж кривої, що є зміщенням. Неупереджена модель має залишки, які випадковим чином розкидані навколо нуля. Невипадкові залишкові моделі вказують на погану підгонку, незважаючи на високий R2. Завжди потрібно звертати увагу на залишки.

Цей тип зміщення специфікації виникає, коли ваша лінійна модель недостатньо визначена. Іншими словами, у ньому відсутні значущі незалежні змінні, поліноміальні терміни та терміни взаємодії. Щоб отримати випадкові залишки, спробуйте додати терміни до моделі або підігнати нелінійну модель.

На перший погляд R-квадрат здається легкою для розуміння статистикою, яка вказує, наскільки добре регресійна модель відповідає набору даних. Однак це не розповідає усієї історії. Щоб отримати повну картину, необхідно розглянути значення R2 у поєднанні з графіками залишків, іншою статистикою та глибокими знаннями предметної області [6].

1.4 Нейронна мережа

1.4.1 Визначення нейронної мережі

Нейронні мережі — це набір алгоритмів, змодельованих за зразком людського мозку (рис. 1.21.), які призначені для розпізнавання шаблонів. Вони інтерпретують сенсорні дані за допомогою свого роду машинного сприйняття, маркування або групування вихідних даних. Патерни, які вони розпізнають, є числовими, містяться у векторах, у які мають бути переведені всі дані реального світу, будь то зображення, звук, текст чи часові ряди.

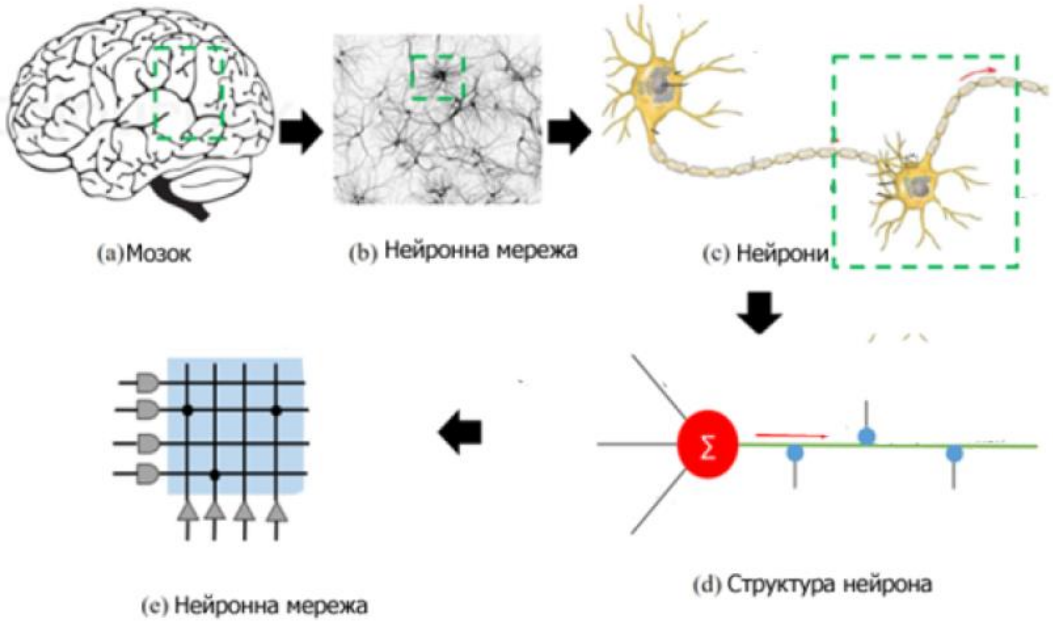


Рис. 1.21. Схожість людського мозку та штучної нейронної мережі

Нейронні мережі допомагають нам групувати та класифікувати. Ви можете розглядати їх як рівень кластеризації та класифікації поверх даних, які ви зберігаєте та керуєте. Вони допомагають групувати немарковані дані відповідно до подібності між прикладами вхідних даних, і вони класифікують дані, коли у них є позначений набір даних для навчання. (Нейронні мережі також можуть витягувати функції, які подаються в інші алгоритми для кластеризації та класифікації; тому ви можете розглядати глибокі нейронні мережі як компоненти більших програм машинного навчання, що включають алгоритми навчання з підкріпленням, класифікації та регресії. рис. 1.22.)

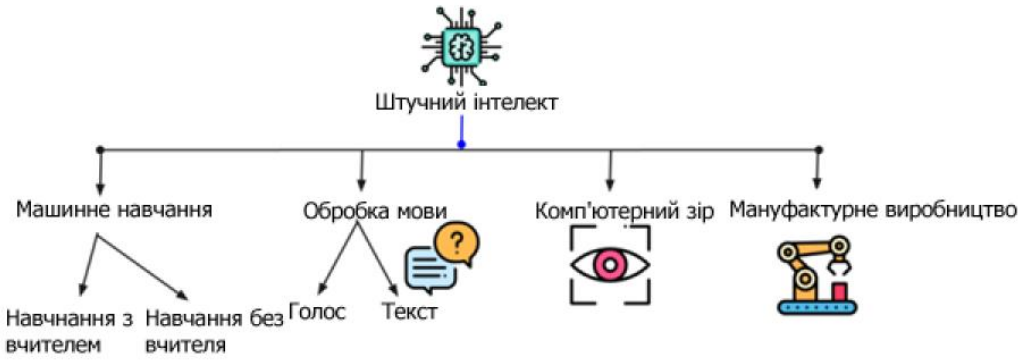


Рис. 1.22. Сфери застосувань штучних нейронних мереж

Які проблеми вирішує глибоке навчання? Щоб дізнатися відповідь, потрібно задати кілька запитань:

- Які результати турбують? У задачі класифікації ці результати є мітками, які можна застосувати до даних: наприклад, spam or not_spam у фільтрі електронної пошти, good_guy або bad_guy у виявленні шахрайства, angry_customer або happy_customer в управлінні відносинами з клієнтами. Інші типи проблем включають виявлення аномалій (корисно для виявлення шахрайства та прогнозного обслуговування виробничого обладнання) і кластеризацію, яка корисна в рекомендаційних системах, які виявляють схожість.
- Чи маємо правильні дані? Наприклад, якщо виникла проблема з класифікацією, знадобляться розмічені дані. Чи є потрібний набір даних загальнодоступним, чи можна його створити (за допомогою служби анотації даних, як-от Scale або AWS Mechanical Turk)? У цьому прикладі електронні листи зі спамом будуть позначені як спам, а мітки дозволять алгоритму відображати вхідні дані до класифікацій, які вас цікавлять. Ви не можете знати, що у вас є правильні дані, поки ви не отримаєте їх у руки. Якщо ви науковець з даних, який працює над проблемою, ви не можете довіряти комусь, щоб сказати вам, чи достатньо хороші дані. Тільки безпосереднє вивчення даних дасть відповідь на це питання.

Глибоке навчання зіставляє входи з результатами. Воно знаходить кореляції. Воно відоме як «універсальний апроксиматор», оскільки може навчитися апроксимувати невідому функцію між будь-яким входом незалежної змінної та будь-яким виходом залежної, припускаючи, що вони взагалі пов’язані (наприклад, кореляцією або причинно-наслідковим зв’язком). У процесі навчання нейронна мережа знаходить правильне функцію або правильний спосіб

перетворення незалежної змінної у залежну. Нижче наведені декілька прикладів того, що може зробити глибоке навчання (нейронні мережі).

- Класифікація (рис. 1.23.). Усі завдання класифікації залежать від розмічених наборів даних, щоб нейронна мережа дізналася кореляцію між мітками та даними. Це відомо як навчання з вчителем. Визначати обличчя, ідентифікувати людей на зображеннях, розпізнавати вирази обличчя (сердитий, радісний), ідентифікуйте об’єкти на зображеннях (знаки зупинок, пішоходів, розмітки смуги…), розпізнавати жести на відео, класифікувати текст як спам (в електронних листах) або шахрайський (у страхових виплатах); розпізнавати настрої в тексті (відгуки клієнтів), тобто будь-які мітки, які можуть створювати люди, будь-які результати, які вас цікавлять і які корелюють з даними, можуть бути використані для навчання нейронної мережі.

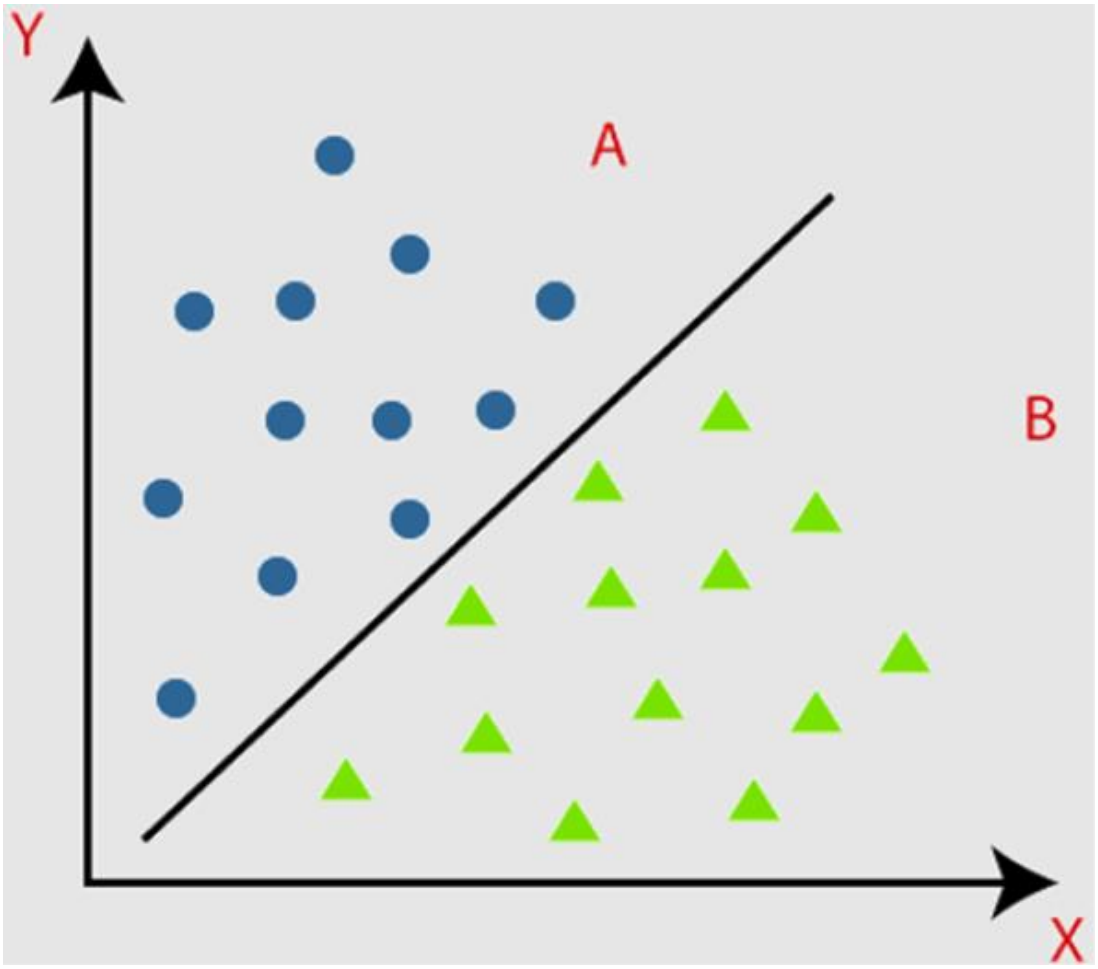


Рис. 1.23. Візуалізація задачі класифікації

- Кластеризація (рис. 1.24.). Кластеризація або групування – це виявлення подібності. Кластеризація – це навчання на не розмічених даних. Навчання без міток (не розмічені дані) називається навчанням без вчителя. Дані без міток – це більшість даних у світі. Один із законів машинного навчання: чим більше даних може тренуватися алгоритм, тим точнішим він буде. Тому навчання без нагляду має потенціал для створення високоточних моделей. Пошук: порівняння документів, зображень або звуків для виявлення схожих елементів. Виявлення аномалій. Зворотною стороною виявлення подібності є виявлення аномалій або незвичайної поведінки. У багатьох випадках незвичайна поведінка дуже корелює з речами, які ви хочете виявити та запобігти, наприклад, шахрайство.

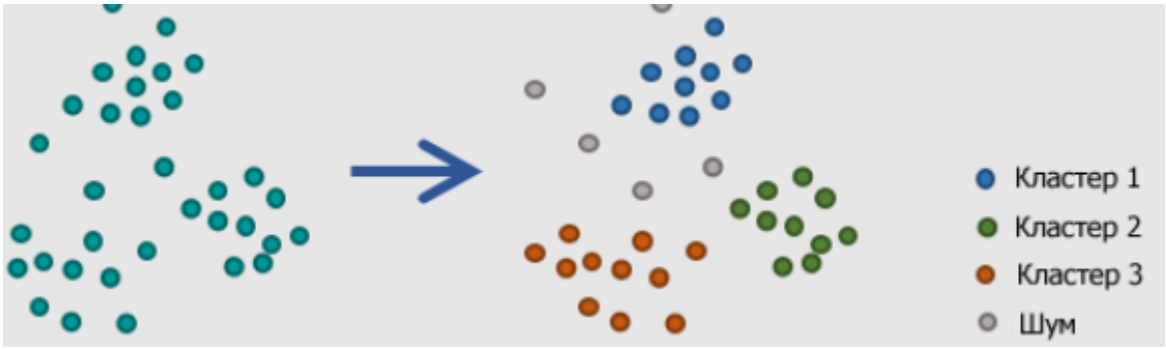


Рис. 1.24. Візуалізація задачі кластеризації

- Прогнозна аналітика: регресія (рис. 1.25.).

  Завдяки класифікації глибоке навчання здатне встановлювати кореляції між, скажімо, пікселями на зображенні та іменем людини. Ви можете назвати це статичним прогнозом. Таким же чином, отримавши достатньо правильних даних, глибоке навчання здатне встановлювати кореляції між нинішніми і майбутніми подіями. Майбутня подія в певному сенсі схожа на етикетку. Глибоке навчання не обов’язково дбає про час або той факт, що щось ще не сталося.

Враховуючи часовий ряд, глибоке навчання може прочитати рядок чисел і передбачити, яке число, найімовірніше, з’явиться наступним. Приклади: поломки обладнання (центри обробки даних, виробництво, транспорт), порушення здоров’я (інсульти, серцеві напади на основі життєво важливих статистичних даних і даних із носіїв), відтік клієнтів, текучість кадрів, чим краще ми можемо передбачити, тим краще ми зможемо запобігти та запобігти.

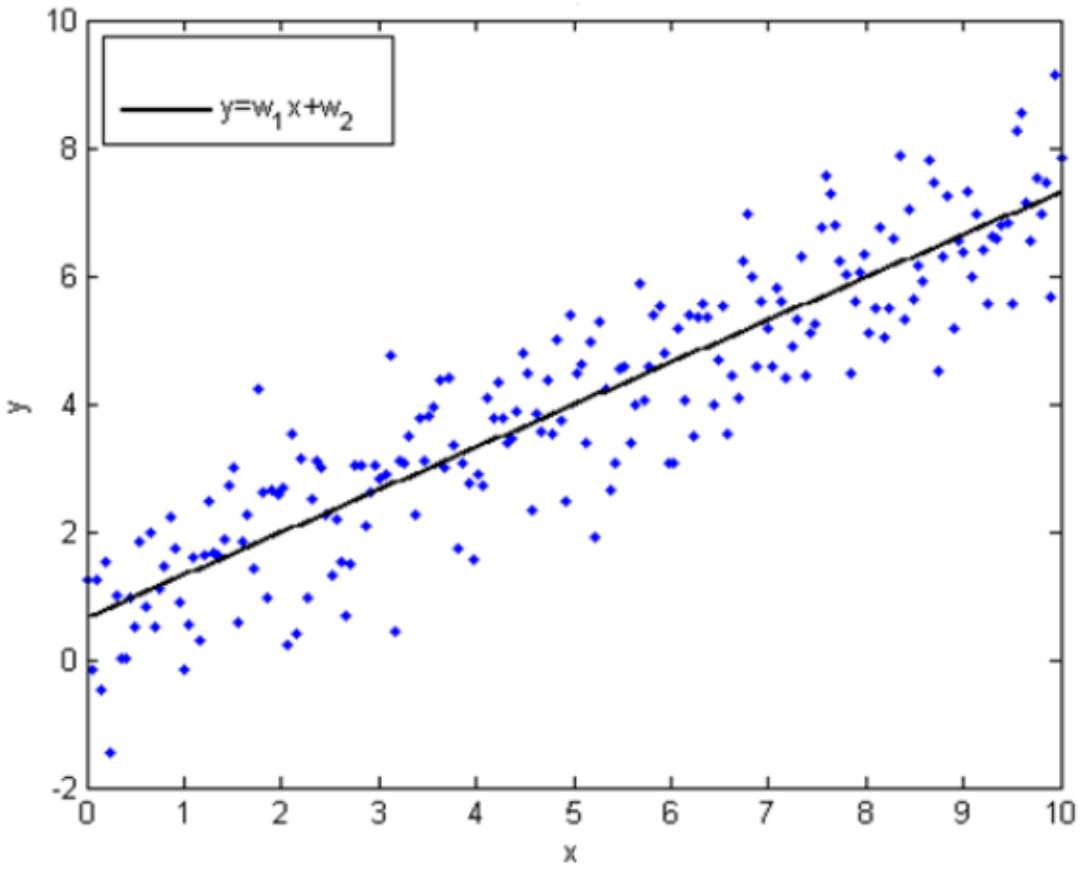


Рис. 1.25. Візуалізація задачі регресії

1.4.2 Складові елементи нейронної мережі

Глибоке навчання — це назва, яку ми використовуємо для нейронних мереж, що складаються з кількох шарів.

Шари складаються з вузлів. Вузол — це просто місце, де відбуваються обчислення. Працюють вузли схоже на нейрони в мозку людини, які

спрацьовують, коли зустрічають достатню кількість подразників. Вузол поєднує вхідні дані з набором коефіцієнтів або ваг, які або посилюють, або гасять цей вхід, тим самим призначаючи значення вхідних даних щодо завдання, яке намагається вивчити алгоритм. Ці вхідні вагові добутки підсумовуються, а потім сума передається через так звану функцію активації вузла, щоб визначити, чи повинен і в якій мірі цей сигнал просуватися далі по мережі, щоб вплинути на кінцевий результат, скажімо, акт класифікації. Якщо сигнали проходять, нейрон був «активований». Візуальне пояснення зображене на рис. 1.26.

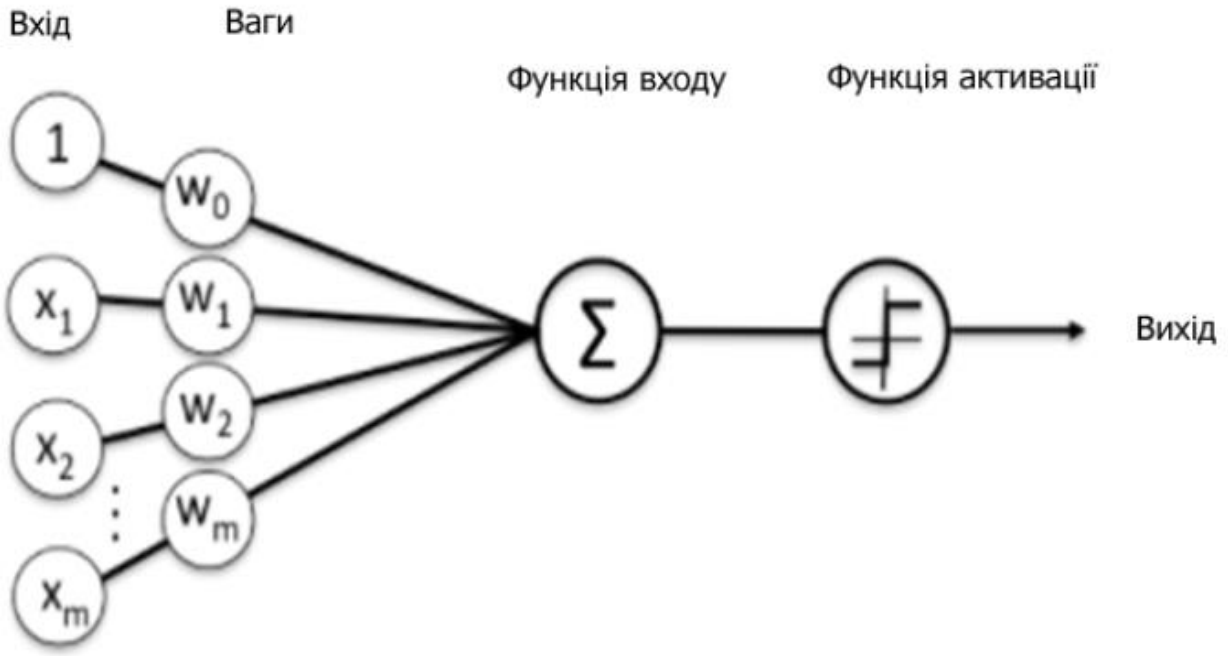


Рис. 1.26. Візуалізація вузла

Вузловий шар — це ряд тих нейроноподібних перемикачів, які вмикаються або вимикаються, коли вхід подається через мережу. Вихід кожного шару одночасно є входом наступного шару, починаючи з початкового вхідного рівня, який отримує ваші дані (рис. 1.27.).

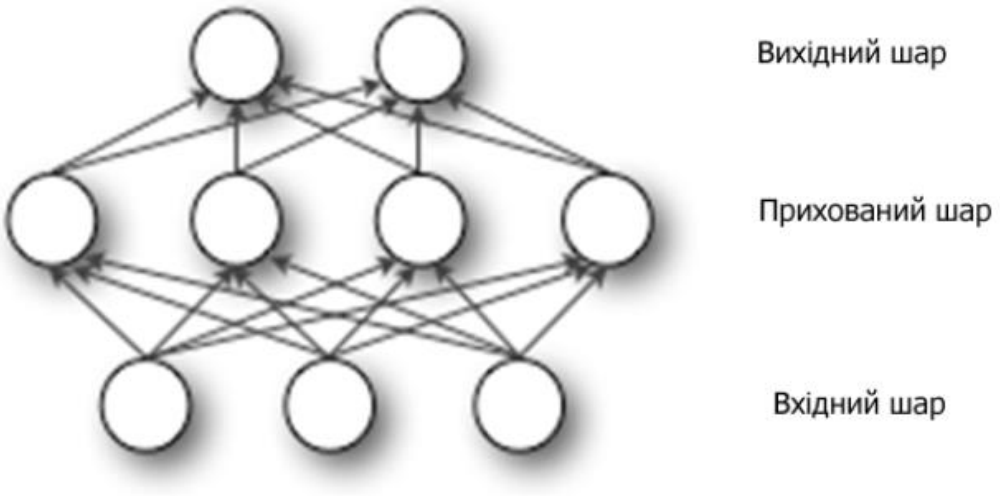


Рис. 1.27. Візуалізація шарів нейромережі

1.4.3 Ключові концепти нейронних мереж

Мережі глибокого навчання відрізняються від звичайних одношарових прихованих нейронних мереж своєю глибиною, тобто кількість шарів вузлів, через які дані повинні проходити в багатоетапному процесі розпізнавання образів. Одношарова нейронна мережа зображена на рис. 1.28., а нейронна мережа глибокого навчання на рис. 1.29.

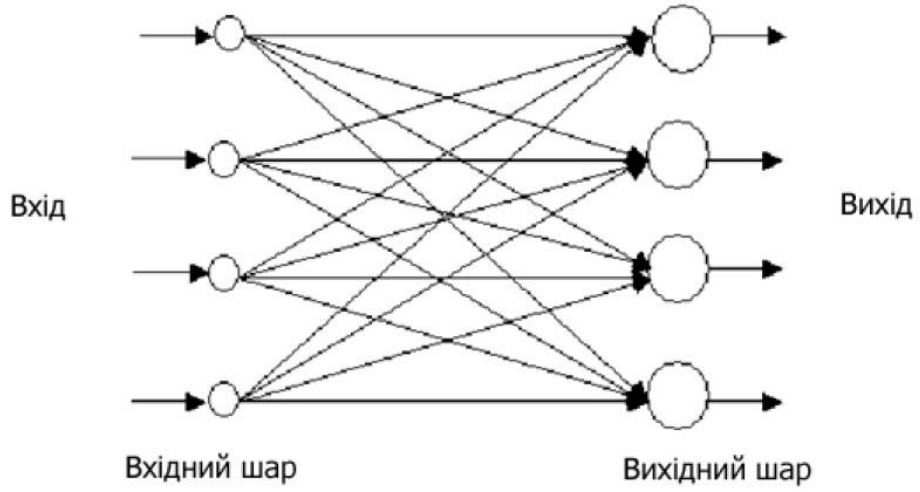


Рис. 1.28. Одношарова нейронна мережа

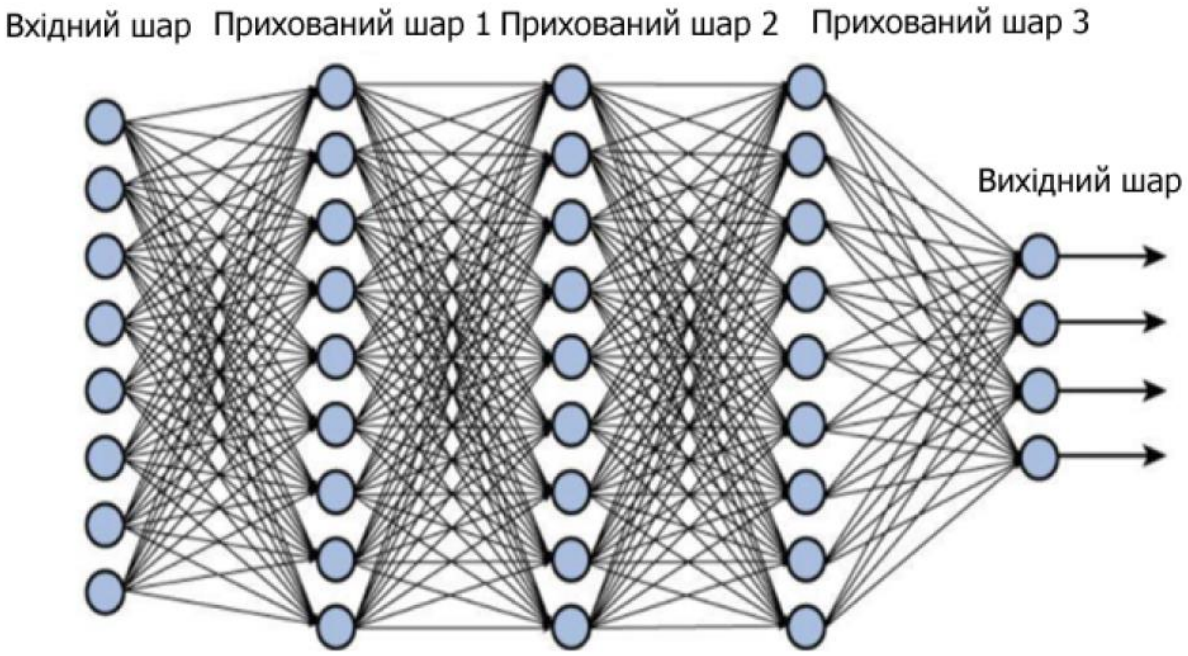


Рис. 1.29. Нейронна мережа глибокого навчання

Попередні версії нейронних мереж, такі як перші персептрони, були неглибокими, складалися з одного вхідного та одного вихідного шарів і щонайбільше одного прихованого шару між ними. Більше трьох рівнів (включаючи вхідний і вихідний) кваліфікується як «глибоке» навчання. У мережах глибокого навчання кожен рівень вузлів тренується на певному наборі функцій на основі результатів попереднього рівня. Чим далі ви просуваєтеся в нейронну мережу, тим складніші функції можуть розпізнати вузли, оскільки вони об’єднують і рекомбінують функції з попереднього шару.

Результат цього процесу відомий як ієрархія функцій, і це ієрархія зростаючої складності та абстракції (рис. 1.30.). Це робить мережі глибокого навчання здатними обробляти дуже великі масиви даних з мільярдами параметрів, які проходять через нелінійні функції.

Перш за все, ці нейронні мережі здатні виявляти приховані структури в немаркованих, неструктурованих даних, які є переважною більшістю даних у світі. Іншим словом для неструктурованих даних є необроблені медіа, тобто зображення, тексти, відео та аудіозаписи. Тому одна з проблем, яке глибоке навчання вирішує найкраще — це обробка та групування необроблених медіа без

міток, виявлення подібності та аномалій у даних, які жодна людина не впорядковувала у реляційну базу даних і не називала їх. Наприклад, глибоке навчання може обробити мільйон зображень і згрупувати їх відповідно до схожості.

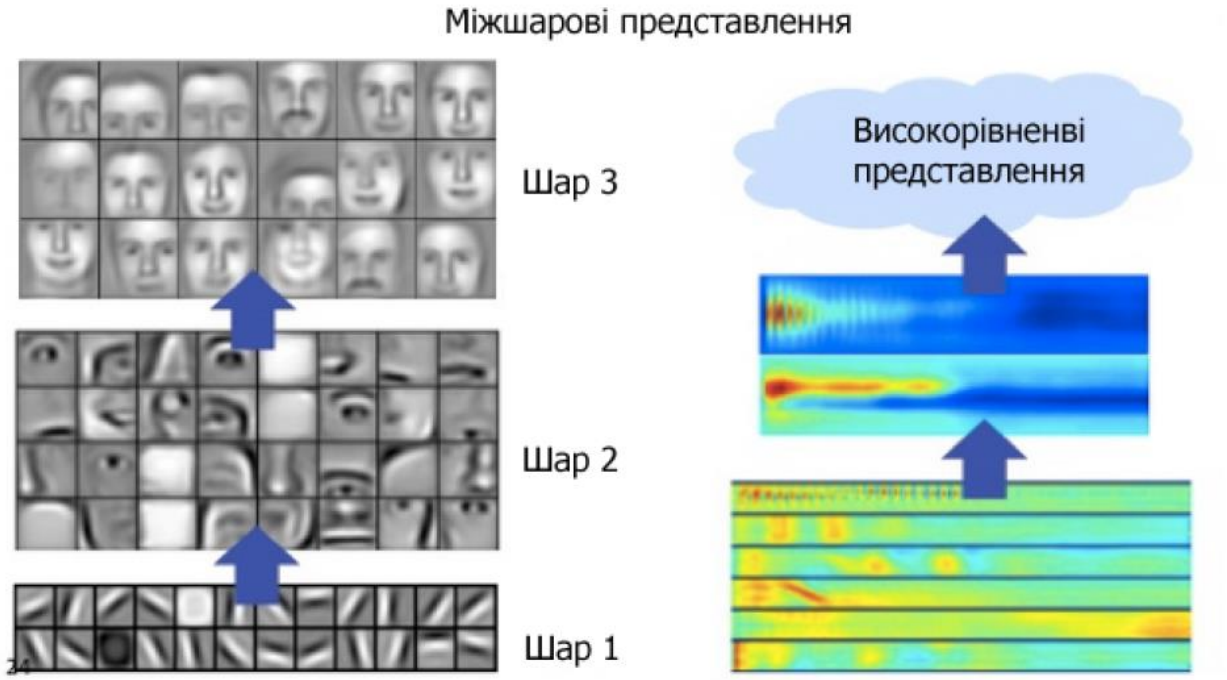


Рис. 1.30. Ієрархічні та абстракті образи нейромережі на основі людських фото

Ту саму ідею можна застосувати до інших типів даних: глибоке навчання може об’єднувати необроблений текст, наприклад електронні листи чи статті новин. Листи, наповнені гнівними скаргами, можуть з’являтися в одному кутку векторного простору, а задоволені клієнти або повідомлення-спам-боти – в інших. Це основа різних фільтрів обміну повідомленнями і може використовуватися в управлінні відносинами з клієнтами (CRM). Те ж саме стосується голосових повідомлень.

З часовими рядами дані можуть групуватися навколо нормальної/здорової поведінки та аномальної/небезпечної поведінки. Якщо дані часового ряду генеруються за допомогою смартфона, це дасть уявлення про здоров’я та звички

користувачів; якщо його генерує автозапчастина, його можна використовувати для запобігання катастрофічних поломок.

На відміну від більшості традиційних алгоритмів машинного навчання, мережі глибокого навчання виконують автоматичний вибір факторів без втручання людини. З огляду на те, що вибір факторів — це завдання, на виконання якого командам науковців з даних можуть знадобитися роки, глибоке навчання — це спосіб обійти обмеженість експертів. Це збільшує повноваження невеликих команд науки про дані, які за своєю природою не масштабуються.

Під час навчання на немаркованих даних кожен рівень вузла в глибокій мережі автоматично вивчає особливості, багаторазово намагаючись відновити вхідні дані, з яких він бере свої вибірки, намагаючись мінімізувати різницю між припущеннями мережі та розподілом ймовірностей самих вхідних даних (рис. 1.31.). Обмежені машини Больцмана, наприклад, створюють так звані реконструкції.

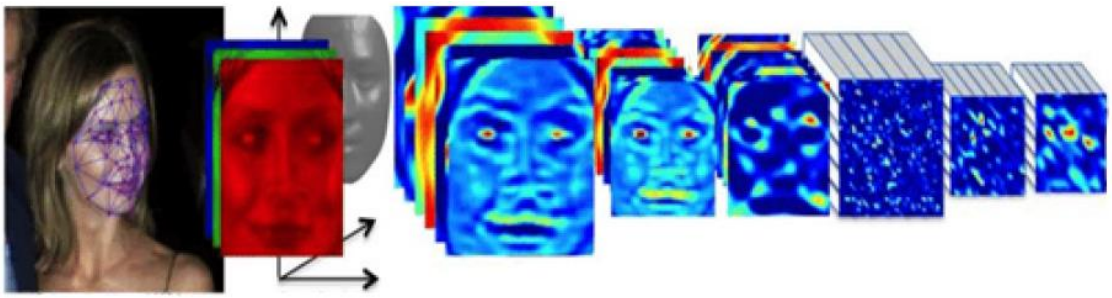


Рис. 1.31. Автоматичний вибір факторів нейронною мережею

У процесі ці нейронні мережі вчаться розпізнавати кореляції між певними релевантними ознаками та оптимальними результатами – вони створюють зв’язки між сигналами ознак і тим, що ці функції представляють, чи то повна реконструкція, чи з позначеними даними.

Мережу глибокого навчання, навчену на позначених даних, можна потім застосувати до неструктурованих даних, надаючи їй доступ до набагато більшої кількості вхідних даних, ніж мережі машинного навчання. Це рецепт вищої продуктивності: чим більше даних може тренуватися мережа, тим точнішою вона

буде. (Погані алгоритми, навчені на великій кількості даних, можуть перевершити хороші алгоритми, навчені на дуже малій кількості.) Здатність глибокого навчання обробляти та вчитися на величезній кількості немаркованих даних дає йому явну перевагу перед попередніми алгоритмами. Мережі глибокого навчання закінчуються вихідним рівнем: логістичним, або softmax, класифікатором, який призначає ймовірність певного результату або мітки. Ми називаємо це прогнозним, але воно є прогнозним у широкому сенсі. Враховуючи вихідні дані у вигляді зображення, мережа глибокого навчання може вирішити, наприклад, що вхідні дані на 90 відсотків представляють людину.

1.4.4 Нейронна мережа прямого поширення

Мета нейронної мережі — якомога швидше прийти до точки найменшої помилки. Початок навчання — це стан, в якому ініціалізовані наші ваги, а кінець навчання — стан тих параметрів, коли вони здатні давати достатньо точні класифікації та прогнози.

Саме навчання включає багато кроків, і кожен з цих кроків нагадує кроки до і після. Кожен крок для нейронної мережі передбачає припущення, вимірювання помилок і невелике оновлення її ваг, поступове коригування коефіцієнтів, оскільки вона повільно вчиться звертати увагу на найважливіші характеристики.

Колекція ваг, незалежно від того, чи знаходяться вони в початковому або кінцевому стані, також називається моделлю, оскільки це спроба змоделювати зв’язок даних з мітками основної істини, щоб зрозуміти структуру даних. Зазвичай моделі починаються з поганої точності і закінчуються менш поганою, змінюючись з часом, коли нейронна мережа оновлює свої параметри.

Причина поганої точності на початку навчання – це те, що ваги, зазвичай, випадковим шляхом ініціалізуються, звичайно ж без розуміння які значення для тих або інших вузлів будуть найбільш сприятливими. Нейронна мережа не знає, які ваги найкраще переведуть вхідні дані, щоб зробити правильні припущення. Мережа повинна почати з припущення, а потім спробувати зробити кращі здогади

послідовно, навчаючись на своїх помилках (можна думати про нейронну мережу як про мініатюрне втілення наукового методу, перевірку гіпотез і повторну спробу – тільки це науковий метод із зав'язаними очима. Або як дитина: вони народжуються, не знаючи багато, і через вплив життєвий досвід, вони повільно вчаться вирішувати проблеми у світі. Для нейронних мереж дані – це єдиний досвід.)

Нижче наведено просто пояснення того, що відбувається під час навчання за допомогою нейронної мережі з прямим зв'язком, найпростішої для пояснення архітектури.

Вхід входить в мережу. Коефіцієнти, або вагові коефіцієнти, відображають вхідні дані в набір припущень, які мережа робить в кінці

$$\text{guess} = \text{input} * \text{weight} \tag{1.5}$$

Зважений вхід дозволяє припустити, що це за вхід. Потім нейронна система робить припущення і порівнює їх з фактичною правдою про дані, фактично запитуючи експерта: «Чи правильно я зрозумів це?»

$$\text{error} = \text{ground truth} - \text{guess} \tag{1.6}$$

Різниця між припущенням мережі та основною істиною полягає в її помилці. Мережа вимірює цю помилку і повертає помилку по своїй моделі, коригуючи ваги в тій мірі, в якій вони сприяли виникненню помилки.

$$\text{adjustment} = \text{error} * \text{weiths contribution error} \tag{1.7}$$

Три формули, наведені вище, абстрактно враховують три ключові функції нейронних мереж: підрахунок введених балів, підрахунок втрат і застосування оновлення до моделі – щоб почати триетапний процес знову. Нейронна мережа — це коригувальний цикл зворотного зв'язку, який нагороджує ваги, які

підтримують її правильні припущення, і карають вагові показники, які призводять до помилок.

1.4.5 Градієнтний спуск

Градієнтний спуск – це часто використовуваної функції оптимізації, яка коригує ваги відповідно до помилки, яку вони викликали. Градієнт – це вектор, що визначає напрямок найскорішого росту функції у точці, а функція представляє собою нейронну мережу. Завдяки використанню від'ємного градієнта (як основа алгоритму оптимізації – градієнтного спуску) змінюється помилка під час коригування ваги (рис. 1.32.).

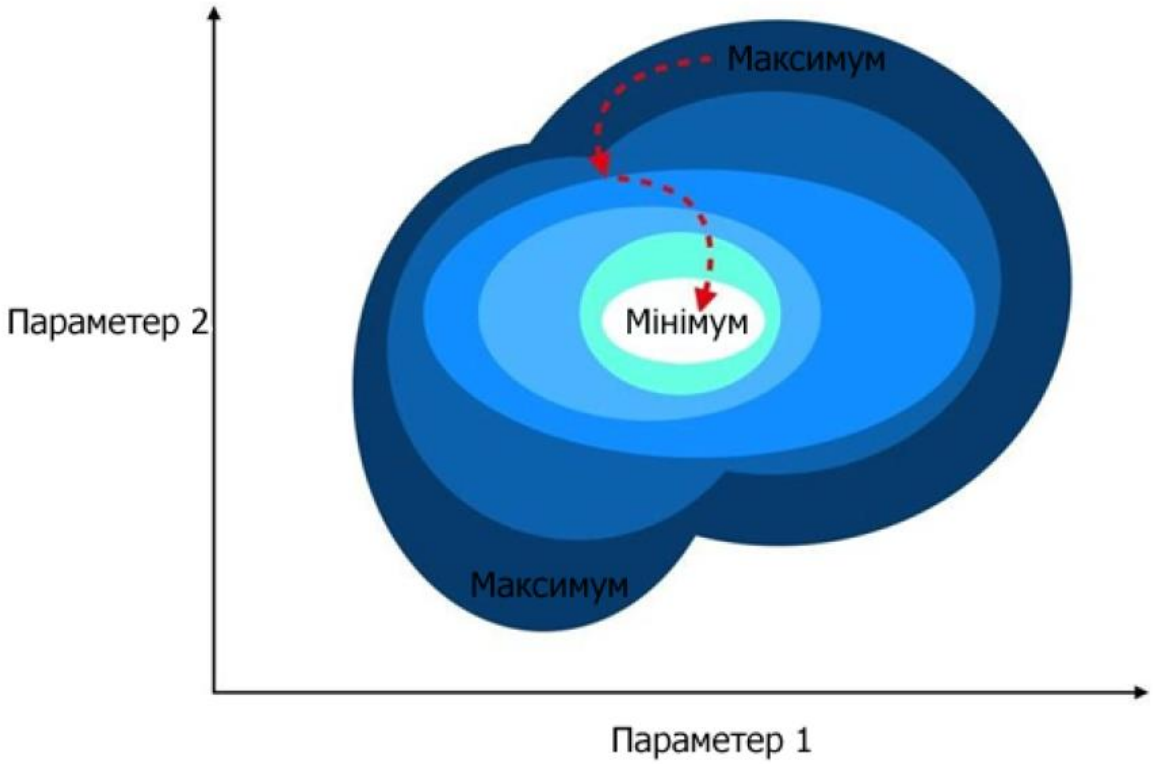


Рис. 1.32. Візуалізація градієнтного спуску

Коли нейронна мережа вчиться, вона повільно коригує багато ваг, щоб вони могли правильно зіставити сигнал із значенням. Зв'язок між мережевою

помилкою та кожним із цих ваг є похідною, яка вимірює ступінь, до якої незначна зміна ваги викликає незначну зміну помилки.

Кожна вага є лише одним фактором у глибокій мережі, яка включає багато перетворень; Сигнал ваги проходить через активації та суми на кількох рівнях, тому ми використовуємо ланцюгове правило обчислення, щоб повернутися назад через активації та виходи мережі і, нарешті, прийти до відповідної ваги та її зв’язку із загальною помилкою.

Правило ланцюга в обчисленні стверджує, що

$$\frac{dz}{dx} = \frac{dz}{dy} * \frac{dy}{dx} \tag{1.8}$$

де dz – похибка, dx – ваги, dy – активації.

Суть глибокого навчання полягає не більше, ніж у коригування ваги моделі у відповідь на помилку, яку вона створює, доки ви не зможете зменшити помилку [7].

1.4.6 Метод зворотного розповсюдження похибки

У машинному навчанні зворотне поширення є широко використовуваним алгоритмом для навчання нейронних мереж із прямим зв’язком (рис.1.33.). Узагальнення зворотного поширення існують для інших штучних нейронних мереж (ШНМ) і для функцій взагалі. Усі ці класи алгоритмів у загальному вигляді називаються "зворотним поширенням". При підборі нейронної мережі зворотне поширення обчислює градієнт функції втрат відносно ваг мережі для одного прикладу введення-виведення, і робить це ефективно, на відміну від наївного прямого обчислення градієнта щодо кожної ваги окремо (рис.1.34.). Ця ефективність робить можливим використання градієнтних методів для навчання багатошарових мереж, оновлюючи ваги для мінімізації втрат. Зазвичай використовуються градієнтний спуск або такі варіанти, як стохастичний

градієнтний спуск. Алгоритм зворотного поширення працює шляхом обчислення градієнта функції втрат щодо кожної ваги за ланцюговим правилом, обчислення градієнта по одному шару за раз, ітерації назад від останнього шару, щоб уникнути зайвих обчислень проміжних членів у правилі ланцюга – це приклад динамічного програмування [8].

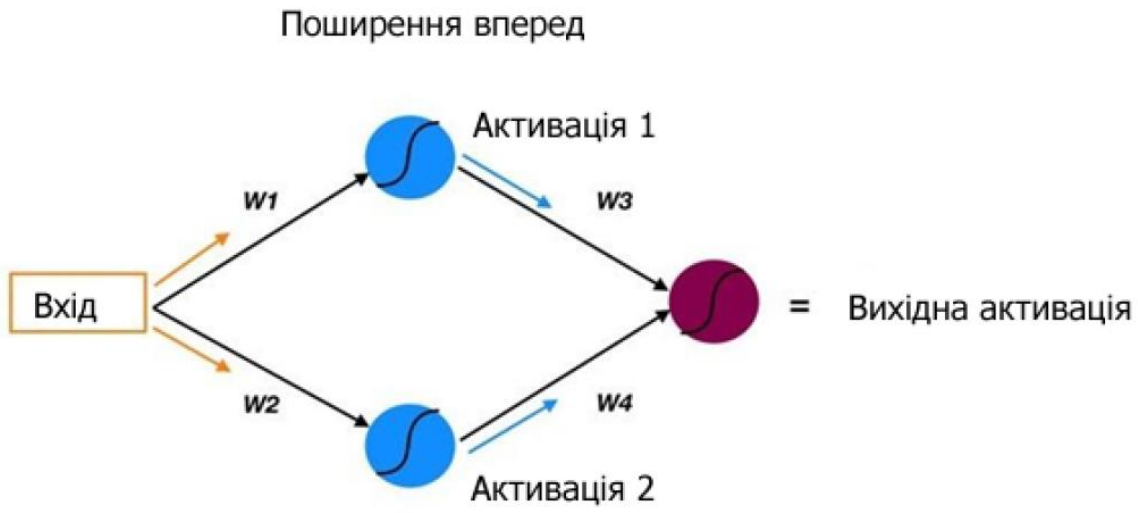


Рис. 1.33. Візуалізація поширення вперед

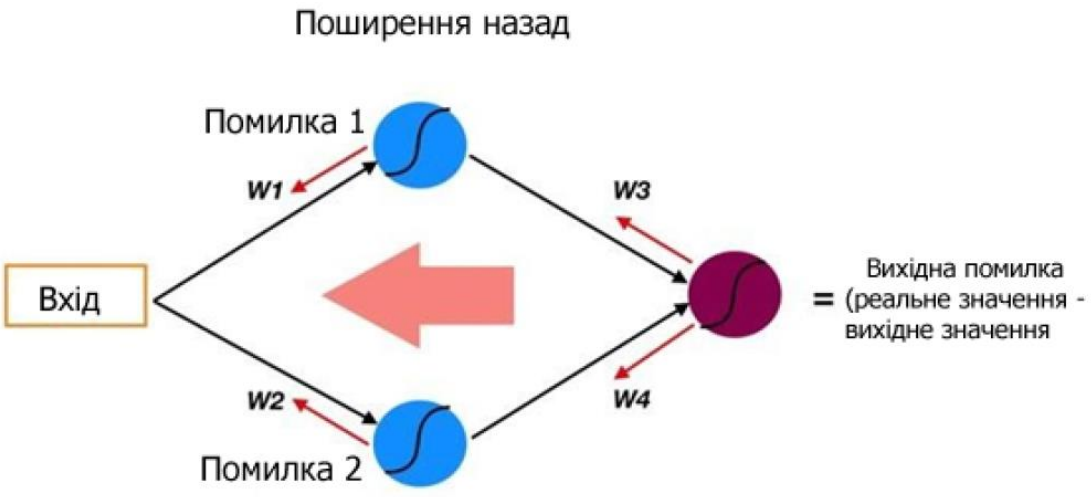


Рис. 1.34. Візуалізація метода зворотного розповсюдження похибки

Величина помилки конкретного нейрона (відносно помилок усіх інших нейронів) прямо пропорційна впливу виходу цього нейрона (активації) на нашу функцію вартості. Таким чином, помилка кожного нейрона є проксі для часткової похідної функції вартості відносно вхідних даних цього нейрона. Часткові похідні для кожної ваги та зміщення є окремими елементами, які складають вектор градієнта нашої функції вартості.

Таким чином, в основному обернене поширення дозволяє обчислити помилки, що приписуються кожному нейрону, а це, у свою чергу, дозволяє обчислити часткові похідні і, зрештою, градієнт, щоб ми могли використовувати градієнтний спуск.

1.5 Автокодувальник

1.5.1 Визначення автокодувальника

Автокодувальники (або автокодери) — це специфічний тип нейронних мереж із прямим зв’язком, де вхідні дані збігаються з виходом. Вони стискають вхідні дані в простір нижчого розміру, а потім відновлюють вихідні дані з цього представлення. Простір є компактним «зведенням» або «стисненням» вхідних даних, яке також називають представленням латентного простору.

Автокодувальник складається з 3 компонентів: кодера (encoder), латентного простору (code) та декодера (decoder). Кодер стискає вхід та створює латентний простір, декодер потім відновлює вхід лише за допомогою цього простору (рис. 1.35.).

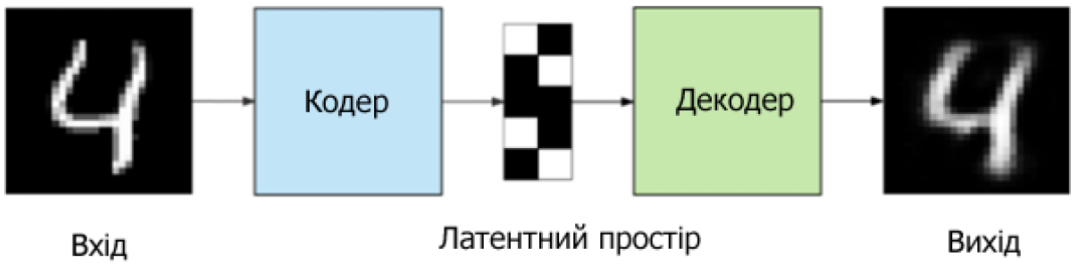


Рис. 1.35. Візуалізація роботи автокодувальника

Щоб створити автокодер, нам потрібні 3 речі: метод кодування, метод декодування та функція втрат для порівняння результату з цільовим. Автокодери – це в основному алгоритм зменшення розмірності (або стиснення) з кількома важливими властивостями:

- Специфічні дані: автокодери здатні лише суттєво стискати дані, подібні до того, на чому вони були навчені. Оскільки вони вивчають особливості, характерні для навчальних даних, вони відрізняються від стандартного алгоритму стиснення даних, такого як gzip. Тому ми не можемо очікувати, що автокодер, навчений рукописним цифрам, стискатиме пейзажні фотографії.
- Втрати: вихід автокодера не буде точно таким же, як вхід, це буде близьке, але погіршене представлення. Якщо ви хочете стиснення без втрат, вони не підходять.
- Навчання без учителя: щоб навчити автокодер, нам не потрібно робити нічого незвичайного, просто кидайте в нього вихідні дані. Автокодери вважаються технікою навчання без вчителя, оскільки їм не потрібні явні мітки для навчання. Але якщо бути точніше, вони перебувають під самоконтролем, оскільки вони створюють власні мітки з даних навчання.

1.5.2 Архітектура автокодувальника

Як кодер, так і декодер є повністю пов’язаними нейронними мережами з прямим зв’язком, по суті. Латентний простір – це один шар мережі з розмірністю за нашим вибором.

Спочатку вхідний сигнал проходить через кодер, щоб створити латентний простір. Декодер, який має подібну структуру, потім виробляє вихід тільки за допомогою латентного простору. Мета – отримати вихід, ідентичний входу. Потрібно зауважити, що архітектура декодера є дзеркальним відображенням кодера. Це не обов’язкова вимога, але зазвичай це так. Єдина вимога полягає в

тому, що розміри вхідних і вихідних даних повинні бути однаковими. Більш детальна візуалізація автокодувальника наведена на рис. 1.36.

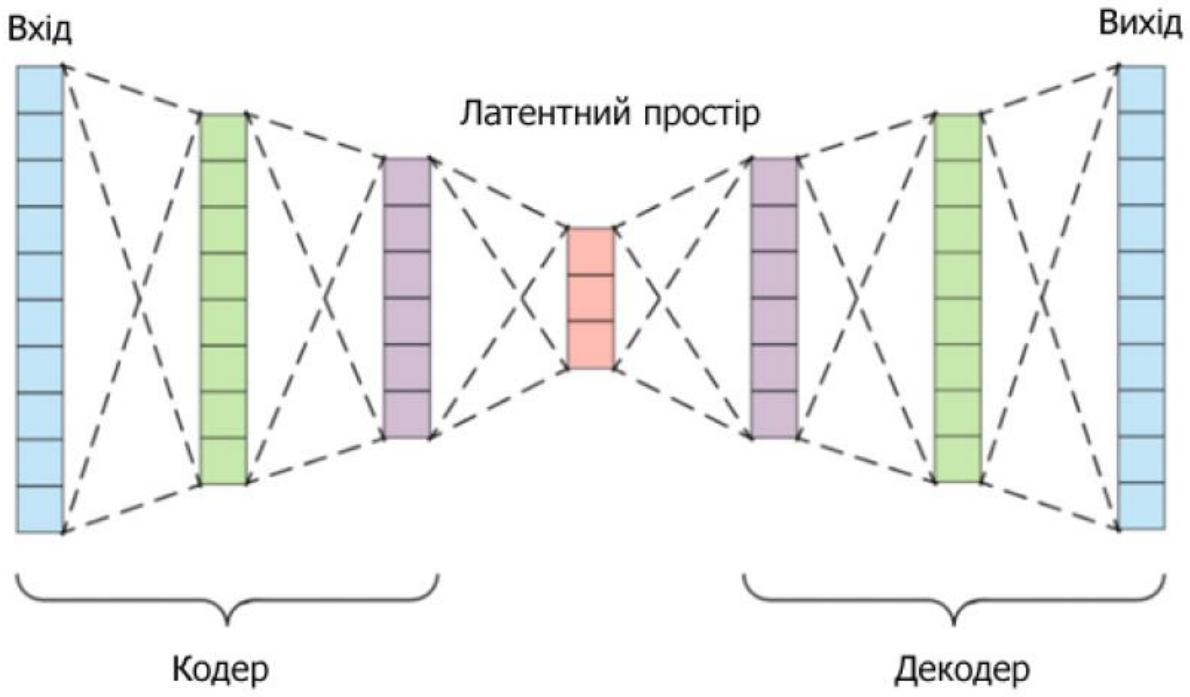


Рис. 1.36. Візуалізація архітектури автокодувальника

Існують 4 гіперпараметри, які нам потрібно встановити перед навчанням автокодера:

- Розмір латентного простору: кількість вузлів у середньому шарі. Менший розмір призводить до більшого стиснення.
- Кількість шарів: автокодер може бути настільки глибоким, як нам подобається. На рис.1.36. (вище) ми маємо 2 шари як в кодері, так і в декодері, не враховуючи вхід і вихід.
- Кількість вузлів на шар: архітектура автокодера, над якою ми працюємо, називається автокодером із стеком, оскільки шари укладаються один за одним. Зазвичай складені автокодери виглядають як «сендвіч». Кількість вузлів на шар зменшується з кожним наступним шаром кодера і збільшується назад у декодері. Також декодер симетричний кодеру з точки зору структури шару. Як

зазначалося вище, це не обов’язково, і ми маємо повний контроль над цими параметрами.

- Функція втрат: ми використовуємо або середню квадратичну помилку (MSE), або двійкову кросентропію. Якщо вхідні значення знаходяться в діапазоні [0, 1], ми зазвичай використовуємо кросентропію, інакше ми використовуємо середню квадратичну помилку.

Автокодери навчаються так само, як і звичайні штучні нейронні мережі – за допомогою зворотного поширення.

Загалом, під час реконструкції даних частина інформації втрачається. На рис. 1.37. зображено втрату інформації під час реконструкції, але загалом результат вважається дуже ефективним.

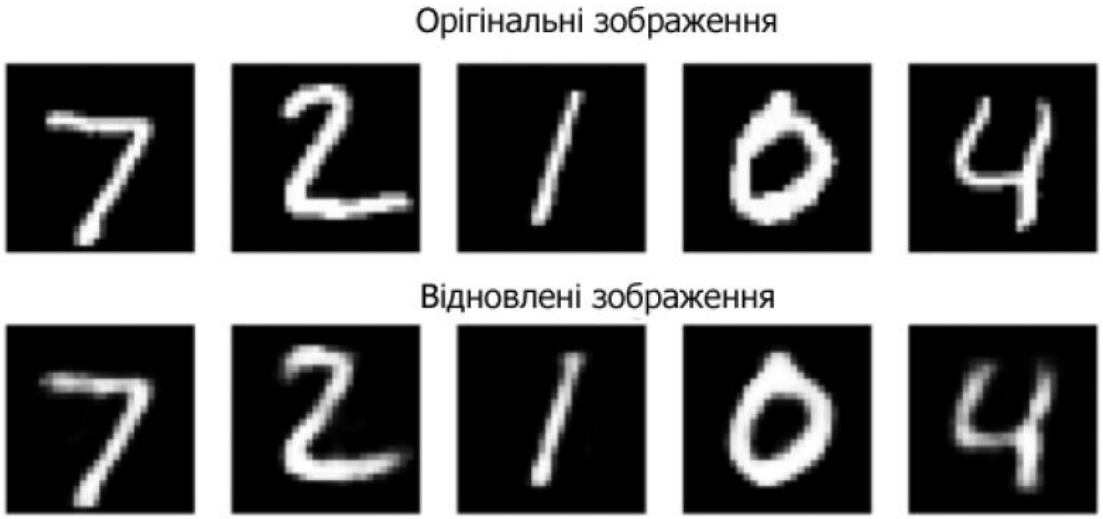


Рис. 1.37. Візуалізація реконструкції даних

Вони дійсно дуже схожі, але не зовсім однакові. Більш чітко це видно на останній цифрі «4».

Автокодувальник можна зробити дуже потужним, збільшивши кількість шарів, вузлів на шар і, головне, розмір латентного простору. Збільшення цих гіперпараметрів дозволить автокодеру вивчати більш складні кодування. Але потрібно бути обережними, щоб не зробити його занадто потужним. Інакше

автокодер просто навчиться копіювати свої вхідні дані у вихідні дані, не вивчаючи жодного змістовного представлення. Це буде просто імітування функції ідентифікації. Автокодер ідеально відновить навчальні дані, але він буде без можливості узагальнювати нові екземпляри.

Ось чому потрібно навмисно зберігати невеликий розмір коду. Оскільки рівень кодування має меншу розмірність, ніж вхідні дані, автокодер вважається недоповненим. Він не зможе безпосередньо копіювати свої вхідні дані у вихідні дані, і буде змушений вивчати інтелектуальні функції. Якщо вхідні дані мають шаблон, наприклад, цифра «1» зазвичай містить дещо пряму лінію, а цифра «0» є круговою, він дізнається цей факт і закодує його в більш компактній формі. Якщо вхідні дані були повністю випадковими без будь-якої внутрішньої кореляції або залежності, то недоповнений автокодер не зможе їх повністю відновити.

Збереження невеликого латентного простору змусило наш автокодер навчитися розумному представленню даних. Існує ще один спосіб змусити автокодер навчитися корисним функціям – додавати випадковий шум до його вхідних даних і змушувати відновлювати вихідні дані без шумів (рис. 1.38.). Таким чином, автокодер не може просто скопіювати вхід на свій вихід, оскільки вхід також містить випадковий шум. Ми просимо його відняти шум і створити основні значущі дані. Це називається автокодером, що знімає шуми.

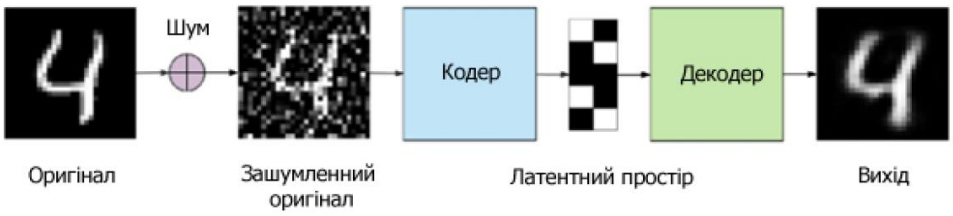


Рис. 1.38. Додавання випадкового шуму до даних

На рис. 1.39. верхній ряд містить оригінальні зображення. Ми додаємо до них випадковий гауссовий шум, і зашумлені дані стають вхідними для автокодера. Автокодер взагалі не бачить оригінальне зображення. Але тоді ми очікуємо, що автокодер відновить вихідне зображення без шумів.

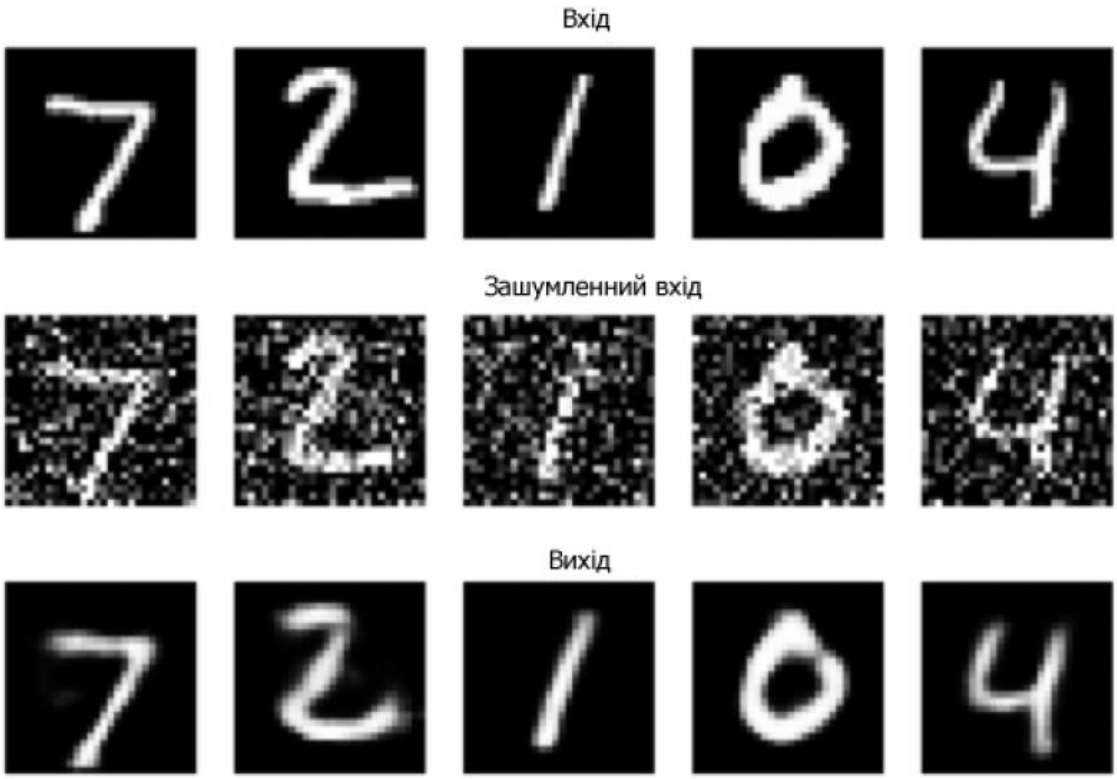


Рис. 1.39. Візуалізація реконструкції даних з шумом

На жаль, автокодувальники не широко використовуються в реальних програмах. Як метод стиснення, вони не працюють краще, ніж його альтернативи, наприклад, jpeg стискає фото краще, ніж автокодер. А той факт, що автокодери є специфічними для даних, робить їх непрактичними як загальну техніку. Однак у них є 3 поширені випадки використання:

- Зниження шуму даних: ми бачили приклад цього на зображеннях.
- Зменшення розмірності: візуалізація високорозмірних даних є складною задачею. t-SNE є найбільш часто використовуваним методом, але він бореться з великою кількістю вимірів (зазвичай понад 32). Таким чином, автокодери використовуються як етап попередньої обробки для зменшення розмірності, і це стиснуте представлення використовується t-SNE для візуалізації даних у 2D просторі.
- Варіаційні автокодери (VAE): це більш сучасний і складний варіант використання автокодерів. VAE вивчає параметри розподілу

ймовірностей, моделюючи вхідні дані, замість того, щоб вивчати довільну функцію у випадку звичайних автокодерів. Вибираючи точки з цього розподілу, ми також можемо використовувати VAE як генеративну модель [9].

Автокодувальники є дуже корисною технікою зменшення розмірності, тому в даній роботі вони будуть використовуватися як основа для підвищення точності прогнозування попиту.

2 ФОРМУВАННЯ ТА ПРОВЕДЕННЯ ЕКСПЕРИМЕНТІВ

2.1 Розробка гіпотези

У теоретичній частині роботи було розглянуто поняття продуктової канібалізації та її суттєвий вплив на формування попиту. Також доволі широко розкриті поняття та застосування нейронних мереж, та особливості використання автокодувальників (основним чином) для стискання даних.

Автокодери можна використовувати для попереднього навачання, наприклад, коли є завдання на класифікацію, а позначених пар занадто мало. Або зменшити розмірність даних для подальшої візуалізації. Або як частина архітектур більш складних нейронних мереж. Або коли потрібно просто навчитися розрізняти корисні властивості вхідного сигналу.

Сама по собі здатність автокодерів стискати дані використовується рідко, оскільки вони зазвичай працюють гірше, ніж рукописні алгоритми для певних типів даних, таких як звуки або зображення. Однак автокодери широко використовуються в областях, де вкрай важко або майже неможливо створити спеціалізований алгоритм стиснення даних. Крім того, добре навчений автокодер здатний перевершити метод головних компонентів (PCA) у стисканні структурованих і неструктурованих даних та зменшенні кількості мультиколінеарних факторів.

Таким чином, автокодери можна використовувати як допоміжний інструмент для підвищення точності алгоритмів навчання з викладачем шляхом введення на додаток до навчальної вибірки векторів просторового представлення даних. Особливість використання цього алгоритму, в контексті даної проблеми, вимагає детального розуміння проблеми, що подається до вирішення. Наприклад, при роботі з часовими рядами необхідно сформувати навчальну вибірку та побудувати архітектуру автокодера таким чином, щоб змусити нейронну мережу шукати найважливіші властивості сигналу в контексті змін над час. Тобто використовувати повторювані шари чи механізми уваги. Також можна

сформувати для алгоритму додаткову задачу прогнозування наступного значення часового ряду, щоб похибка від прогнозного та реального значення вплинула на мінімізацію функції загальної вартості.

Особливості використання автокодерів явно залежать від завдання. Тому застосування цього алгоритму може суттєво допомогти у вирішенні задач, де неможливо або недоцільно використовувати класичні методи стиснення даних [10].

Таким чином, у даній роботі представлений метод прогнозування попиту використовуючи актокодувальник для стискання даних, а потім використання цих векторів у якості додаткових показників під час навчання з учителем. На рис. 2.1. продемонстрована загальна блок-схема процесу прогнозування попиту з використанням автокодувальників.

Рис. 2.1. Блок-схема експерименту

2.2 Навчальні дані

Для проведення експерименту необхідно здобути дані, що підходять для задачі прогнозування попиту. Платформа для Data Science змагань “Kaggle” містить дуже підходящий датасет «M5 Forecasting – Accuracy» [11]. Ціль змагання – це спрогнозувати попит у мережі гіпермаркетів «Walmart».

Дані складаються з 5 файлів (рис. 2.2.).

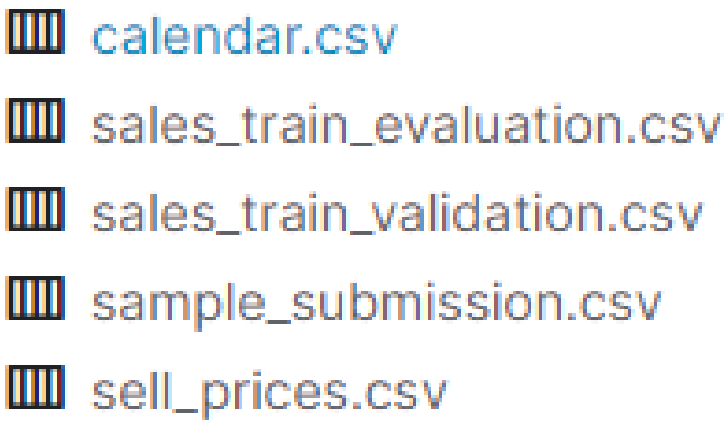


Рис. 2.2. Початкові дані (оригінальний набір даних «Walmart»)

Для наших цілей необхідні тільки три з них:

- calendar.csv містить інформацію про дати продажів, різні промо, івенти та свята.
- sales_train_validation.csv містить загреговані до днів продажі товарів. Усього 10 магазинів по 3049 товарів.
- sell_prices.csv містить інформацію про ціну товару.

У свою чергу потрібно зауважити, що для навчання автокодувальника буде використовуватись тільки продажі товарів, а інша інформація як ціна, категорія групи товару та інші данні будуть у нагоді на етапі фінального навчання разом з векторами латентного простору після навчання автокодувальника.

На рис. 2.3., рис. 2.4., та рис. 2.5. зображено декілька історичних продажів.

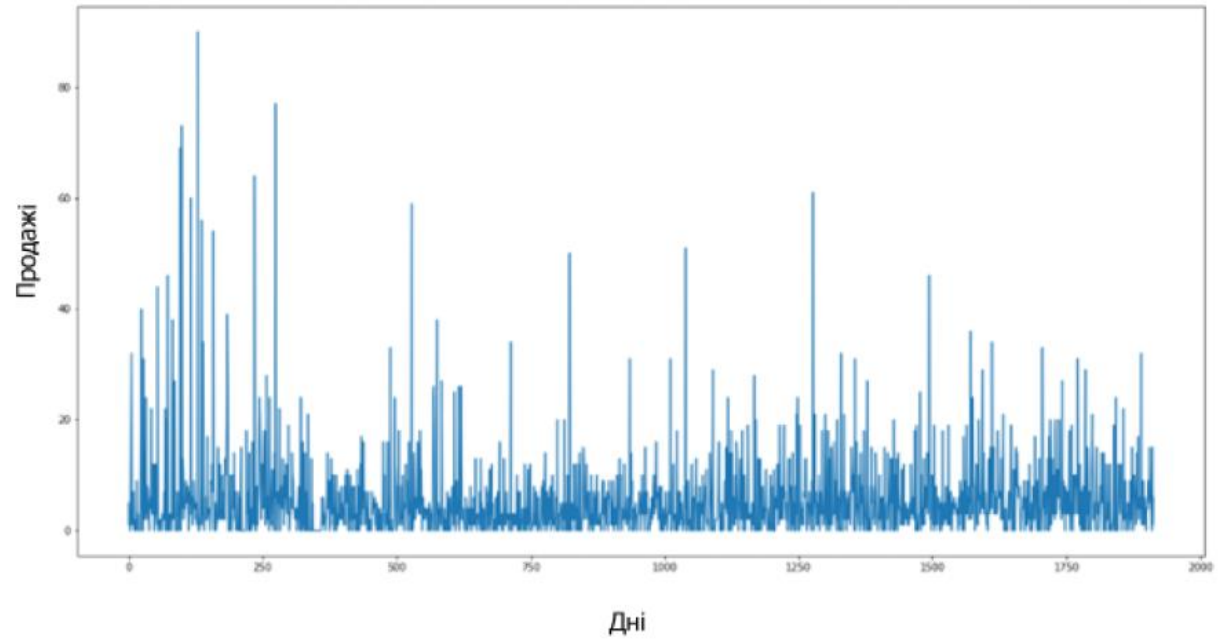


Рис. 2.3. Продажі товару 1

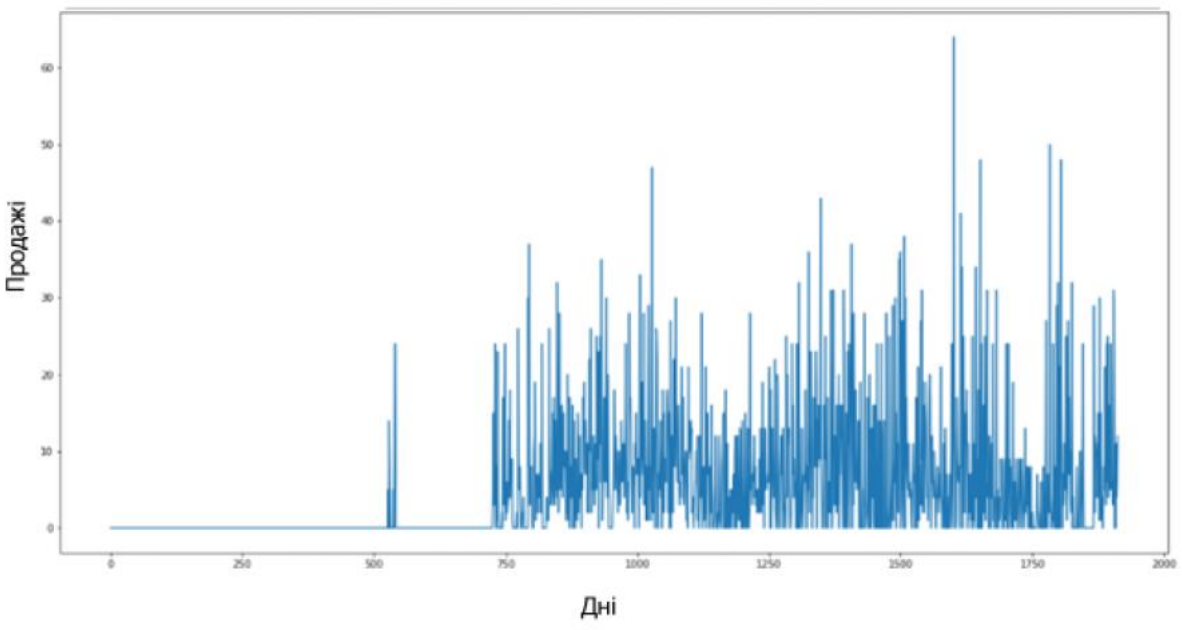


Рис. 2.4. Продажі товару 2

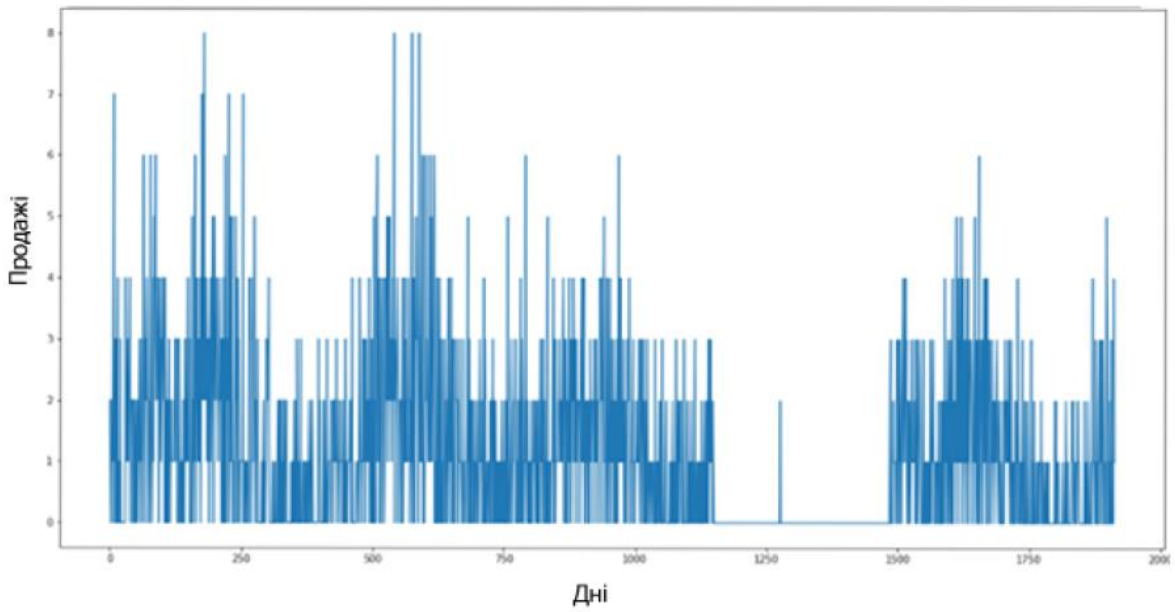


Рис. 2.5. Продажі товару 3

На рис. 2.6. зображено сумарні продажі магазину CA_1.

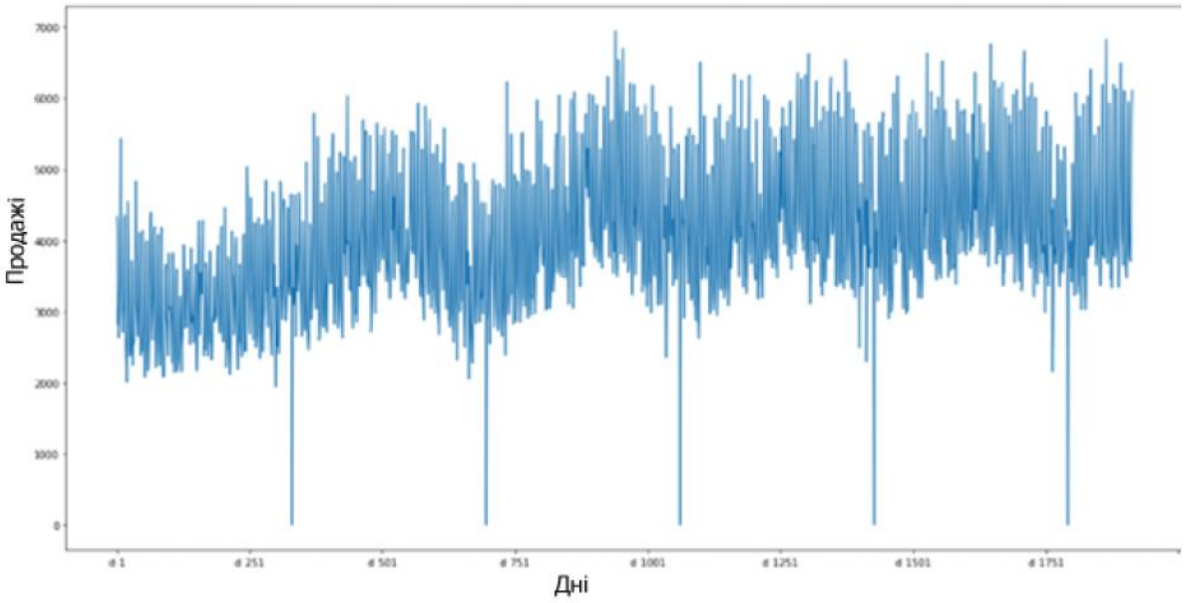


Рис. 2.6. Сумарні продажі магазину CA_1

На рис. 2.7. зображено розподіл сумарних продажів всіх товарів із магазину CA_1.

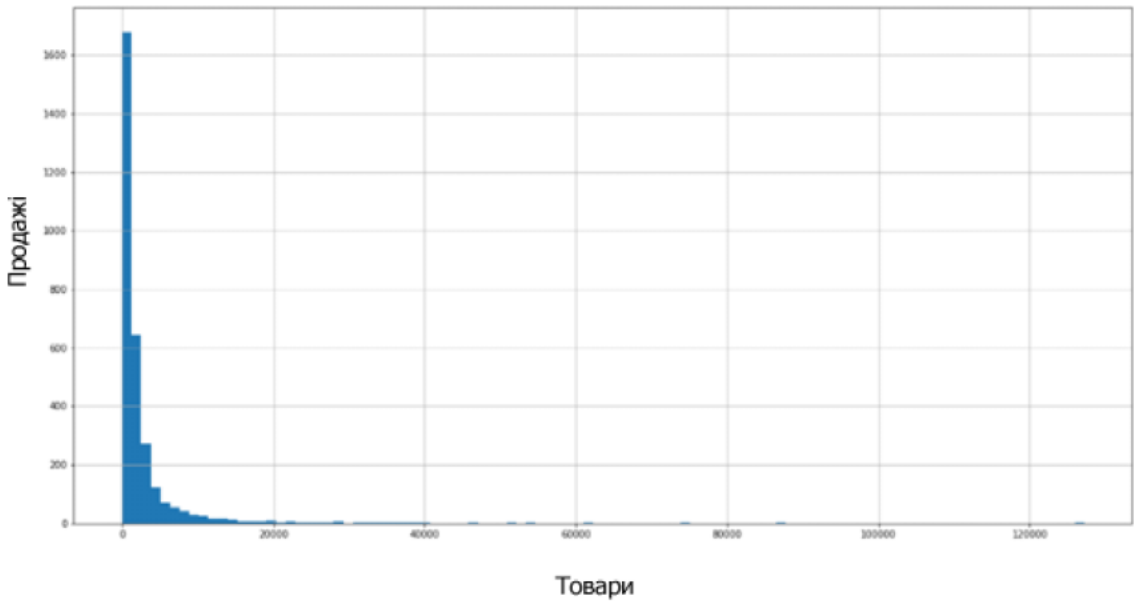


Рис. 2.7. Розподіл сумарних продажів всіх товарів із магазину CA_1

Загалом формується 10 датасетів, де кожен датасет – це продажі одного магазину. Автокодувальник навчається 10 разів по 1 магазину.

Для навчання кожен датасет розбивається на три вибірки: датасет для тренування, для валідації та для тестування. Датасет для тренування – це ті дані, що автокодувальник намагається відновити та на основі цих даних змінює ваги у нейронній мережі. Валідаційний датасет – це дані, що напряму не впливають на зміну ваг, але на основі цього сабсету алгоритм навчання буде вирішувати, коли завершити процес навчання та підбирати гіперпарамтери. Тестові данні – це ті дані, що нейронна мережа ніколи не бачила ні при навчанні ні при валідації. Тому дані для тестування будуть використані для оцінки точності та загалом ефективності навчання автокодувальника.

Датасет для тренування починається з 29 січня 2011 року та триває до 25 вересня 2015 року. Датасет для валідації починається з з 26 вересня 2015 року та закінчується 3 січня 2016 року. Дані для тестування починаються з 4 січня 2016 року та закінчуються разом з усім датасетом, тобто до 19 червня 2016 року. Дана розбивка на піддатасети однакова як і для тренування автокодувальника так і для регресійної моделі.

Також важливо зауважити, що для навчання автокодувальника всі дані будуть нормалізовані від 0 до 1 (рис. 2.8.).

```
1 from sklearn.preprocessing import MinMaxScaler
2
3 scaler = MinMaxScaler().fit(train_data)
4 train_data = scaler.transform(train_data)
5 val_data = scaler.transform(val_data)
6 test_data = scaler.transform(test_data)
```

Рис. 2.8. Нормалізація даних (програмний код)

Фінальні датасети містять 1700 записів для тренування, 100 для валідації та майже 150 для тестування. У кожному датасеті 3049 колонок, що представляють собою 3049 товарів для кожного з магазинів. Розмірність датасету для тренування 1700 рядків на 3049 стовпців. Саме ці 3049 товарів з продажами автокодувальник буде відтворювати та формувати латентні вектори. Результуючий датасет для навчання автокодувальника зображено на рис. 2.9.

id	HOBBIES_1_001_CA_1_validation	HOBBIES_1_002_CA_1_validation	HOBBIES_1_003_CA_1_validation	H
d_1	0	0	0	
d_2	0	0	0	
d_3	0	0	0	
d_4	0	0	0	
d_5	0	0	0	
...	...	...	...	
d_1909	1	1	1	
d_1910	3	0	0	
d_1911	0	0	1	
d_1912	1	0	1	
d_1913	1	0	1	

1913 rows × 3049 columns

Рис. 2.9. Датасет для навчання автокодувальника (вивід фреймворку візуалізації)

Датасет для навчання регресійної моделі буде відрізнятись тим, що у навчанні будуть приймати участь одразу всі магазини, сутності продуктів перейдуть з колонок до рядків та у навчанні будуть приймати інші фактори як ціна. Також до навчального датасету додаються латентні вектори, що будуть отримані від автокодувальника відповідні до магазинів. Результуючий датасет для навчання регресійної моделі зображено на рис. 2.10. Фактори зображені на рис. 2.11.

	id	item_id	dept_id	cat_id	store_id	state_id	d	Target	Date	wm_yr_wk	weekday	wday	month	yea
1	HOBBIES_1_034_CA_1_validation	HOBBIES_1_034	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0	2011-01-29	11101	Saturday	1	1	201
2	HOBBIES_1_037_CA_1_validation	HOBBIES_1_037	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0	2011-01-29	11101	Saturday	1	1	201
3	HOBBIES_1_041_CA_1_validation	HOBBIES_1_041	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0	2011-01-29	11101	Saturday	1	1	201
4	HOBBIES_1_045_CA_1_validation	HOBBIES_1_045	HOBBIES_1	HOBBIES	CA_1	CA	d_1	0	2011-01-29	11101	Saturday	1	1	201
5	HOBBIES_1_047_CA_1_validation	HOBBIES_1_047	HOBBIES_1	HOBBIES	CA_1	CA	d_1	1	2011-01-29	11101	Saturday	1	1	201

Рис. 2.10. Датасет для навчання регресійної моделі (вивід фреймворку візуалізації)

```
Data columns (total 30 columns):
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   id            621724 non-null  object
 1   item_id       621724 non-null  object
 2   dept_id       621724 non-null  object
 3   cat_id        621724 non-null  object
 4   store_id      621724 non-null  object
 5   state_id      621724 non-null  object
 6   d             621724 non-null  object
 7   Target        621724 non-null  int64
 8   Date          621724 non-null  datetime64[ns]
 9   wm_yr_wk      621724 non-null  int64
 10  weekday       621724 non-null  object
 11  wday          621724 non-null  int64
 12  month         621724 non-null  int64
 13  year          621724 non-null  int64
 14  event_name_1  621724 non-null  object
 15  event_type_1  621724 non-null  object
 16  event_name_2  621724 non-null  object
 17  event_type_2  621724 non-null  object
 18  snap_CA       621724 non-null  int64
 19  snap_TX       621724 non-null  int64
 20  snap_WI       621724 non-null  int64
 21  sell_price    621724 non-null  float64
 22  EV_1          621724 non-null  float64
 23  EV_2          621724 non-null  float64
 24  EV_3          621724 non-null  float64
 25  EV_4          621724 non-null  float64
 26  EV_5          621724 non-null  float64
 27  EV_6          621724 non-null  float64
 28  EV_7          621724 non-null  float64
 29  EV_8          621724 non-null  float64
```

Рис. 2.11. Фактори для регресійної моделі (вивід фреймворку візуалізації)

2.3 Навчання автокодувальника

Мета навчання автокодувальника – це після завершення тренування отримати вектори латентного простору, що об’єднати їх з датасетом для навчання регресійної моделі.

Для побудови автокодувальника використовується бібліотека глибокого навчання Keras.

Архітектура автокодувальника містить лише два шари з параметрами, розмірність латентного простору – 8 значень. На рис. 2.12. зображено код для побудови автокодувальника. Дана нейронна мережа складається з вхідного шару, шару нормалізації, шару активації, та шару, що допомагає нейромережі уникнути перенавчання – «Dropout».

```
# Encoder
input_img  = keras.layers.Input(shape=(train_data.shape[1]))

x = keras.layers.Dense(encoding_dim)(input_img)
x = keras.layers.BatchNormalization()(x)
x = keras.layers.Activation('relu')(x)
encoded = keras.layers.Dropout(dropout_rate)(x)


# Decoder
decoded = keras.layers.Dense(train_data.shape[1])(encoded)


# AE model
model = keras.Model(input_img, decoded)

# Encoder model
encoder = keras.Model(input_img, encoded)

# Create the decoder model
encoded_input = keras.Input(shape=(encoding_dim,))
decoder_layer = model.layers[-1]
decoder = keras.Model(encoded_input, decoder_layer(encoded_input))


model.compile(optimizer=keras.optimizers.Adam(), loss=loss, metrics=metrics)
model.summary()
```

Рис. 2.12. Keras-код автокодувальника (програмний код)

Графічне зображення автокодувальника продемонстроване на рис. 2.13.

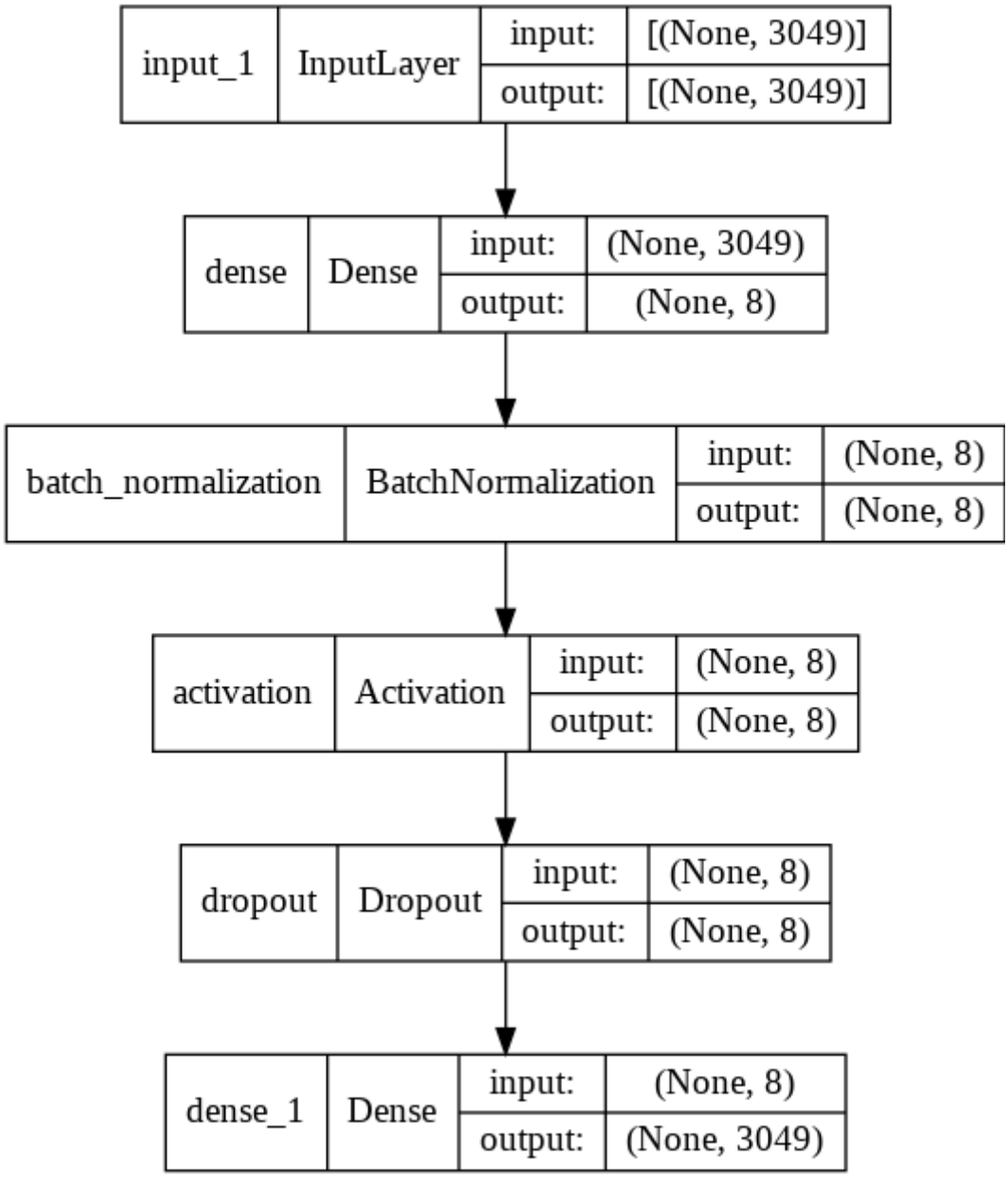


Рис. 2.13. Архітектура автокодувальника (вивід фреймворку візуалізації)

У якості функції витрат та метрикою виступає MSE (Mean Squared Error). Кількість ітерацій навчання 250 з умовою побудови функції, що автоматично зупинить навчання упродовж заданого параметра, якщо функція витрат на валідаційному датасеті не буде зменшуватись, а на тренувальному продовже падати. Таке явище має назву «перенавчання», тобто коли модель має чудову точність на даних в тренувальному датасеті, та погану на валідаційному та

тестовому. У такому випадку модель не шукає закономірності у даних, а лише вивчає тренувальну вибірку. На рис. 2.14. зображено код, що створює функцію для автоматичної зупинки навчання («EarlyStopping»).

```
path_checkpoint = "model_checkpoint.h5"
es_callback = keras.callbacks.EarlyStopping(monitor="val_loss", min_delta=0, patience=50)

modelckpt_callback = keras.callbacks.ModelCheckpoint(
    monitor="val_loss",
    filepath=path_checkpoint,
    verbose=1,
    save_weights_only=True,
    save_best_only=True,
)

history = model.fit(
    x=train_data, y=train_data,
    epochs=epochs,
    validation_data=(val_data, val_data),
    callbacks=[es_callback, modelckpt_callback],
)
```

Рис. 2.14. Keras-код early sropping (програмний код)

Кожна ітерація навчання супроводжується перевіркою точності на валідаційному датасеті, на початку точність буде покращуватись, але з часом модель все більше й більше буде вчити тренувальні дані та точність на валдації не буде покращуватись. Це не є проблемою даної нейронної мережі або задачі. Проблема перенавчання в цілому є викликом в усьому машинному навчанні, особливо у глибокому навчанні. На рис. 2.15. зображено процес навчання.

```
Epoch 1/250
51/54 [==========================>..] - ETA: 0s - loss: 0.0196 - mse: 0.0196
Epoch 00001: val_loss improved from inf to 0.02316, saving model to model_checkpoint.h5
54/54 [==============================] - 3s 8ms/step - loss: 0.0194 - mse: 0.0194 - val_loss: 0.0232 - val_mse: 0.0232
Epoch 2/250
50/54 [==========================>...] - ETA: 0s - loss: 0.0153 - mse: 0.0153
Epoch 00002: val_loss improved from 0.02316 to 0.02278, saving model to model_checkpoint.h5
54/54 [==============================] - 0s 5ms/step - loss: 0.0152 - mse: 0.0152 - val_loss: 0.0228 - val_mse: 0.0228
Epoch 3/250
50/54 [==========================>...] - ETA: 0s - loss: 0.0148 - mse: 0.0148
Epoch 00003: val_loss improved from 0.02278 to 0.02246, saving model to model_checkpoint.h5
54/54 [==============================] - 0s 5ms/step - loss: 0.0148 - mse: 0.0148 - val_loss: 0.0225 - val_mse: 0.0225
Epoch 4/250
48/54 [=========================>....] - ETA: 0s - loss: 0.0148 - mse: 0.0148
Epoch 00004: val_loss improved from 0.02246 to 0.02230, saving model to model_checkpoint.h5
54/54 [==============================] - 0s 5ms/step - loss: 0.0147 - mse: 0.0147 - val_loss: 0.0223 - val_mse: 0.0223
Epoch 5/250
49/54 [==========================>...] - ETA: 0s - loss: 0.0146 - mse: 0.0146
Epoch 00005: val_loss did not improve from 0.02230
```

Рис. 2.15. Ітеративний процес навчання нейронної мережі (вивід фреймворку візуалізації)

На рис. 2.16. зображено графіки значень функції витрат для тренувального та валідаційного датасетів під час навчання.

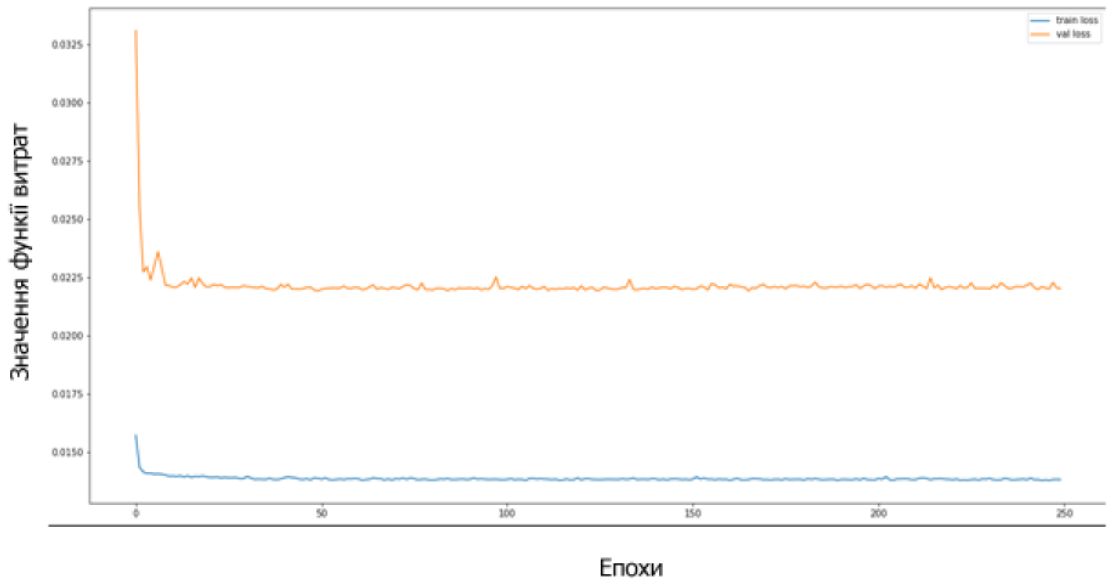


Рис. 2.16. Значення функцій витрат

На рис. 2.17. синім зображено вхідні дані, а помаранчевим – реконструкція одного з часових рядів.

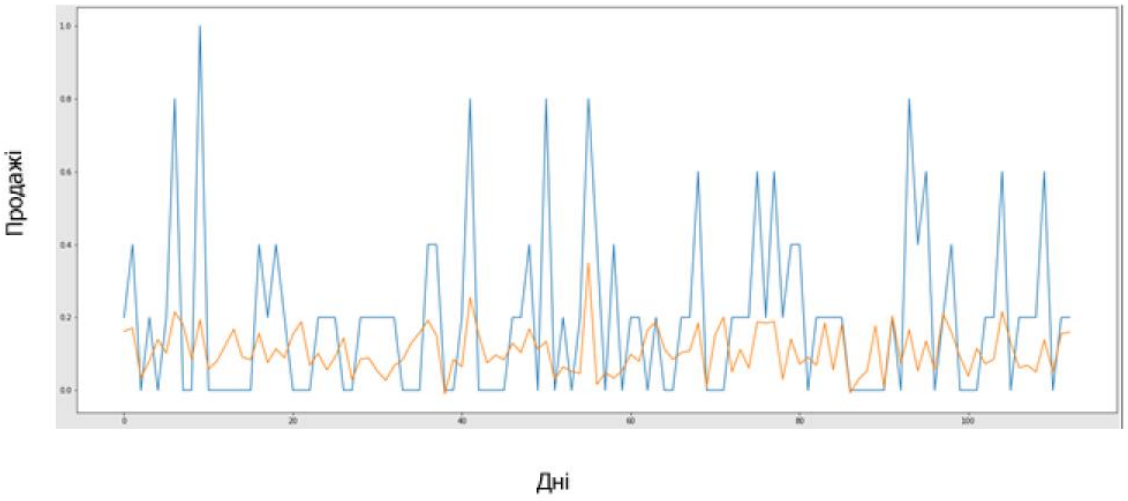


Рис. 2.17. Оригінальний часовий ряд (синім) та його реконструкція (помаранчевий)

Після навчання необхідно отримати значення латентних векторів (рис. 2.18.)

```
encoded_train_data = encoder.predict(train_data)
encoded_val_data = encoder.predict(val_data)
encoded_test_data = encoder.predict(test_data)
```

Рис. 2.18. Інференс значень латентного простору (програмний код)

2.4 Навчання регресійної моделі

Як вже написано у розділі з підготовки даних – для регресійної моделі структура датасету виглядає дещо по іншому, однак межі тренувального, валідаційного та тестового датасетів однакові з тими, що були при навчанні автокодувальника.

Для обробки категоріальних факторів використовується LabelEncoder (рис. 2.19.)

```
cat_encoder_dict = {}
cat_cols = ['item_id', 'dept_id', 'cat_id', 'store_id', 'state_id',
            'event_name_1', 'event_type_1', 'event_name_2', 'event_type_2']
for col in cat_cols:
    encoder = LabelEncoder().fit(train_x[col])
    train_x[col] = encoder.transform(train_x[col])
    val_x[col] = encoder.transform(val_x[col])
    test_x[col] = encoder.transform(test_x[col])

    cat_encoder_dict[col] = encoder
```

Рис. 2.19. LabelEncoder (програмний код)

Перед навчанням регресійної моделі датасет виглядає наступним чином, що зображено на рис. 2.20.

	item_id	dept_id	cat_id	store_id	state_id	wday	month	year	event_name_1	event_type_1	event_name_2	event_type_2	sell_price	EV_1	EV_2	EV_3	EV_4	EV_5	EV_6	EV_7	EV_8
1	154	3	1	0	0	1	1	2011	30	4	4	2	0.00	0.0	0.000000	0.313939	1.343737	1.247311	0.0	0.0	0.000000
2	155	3	1	0	0	1	1	2011	30	4	4	2	0.00	0.0	0.000000	0.313939	1.343737	1.247311	0.0	0.0	0.000000
3	156	3	1	0	0	1	1	2011	30	4	4	2	0.00	0.0	0.000000	0.313939	1.343737	1.247311	0.0	0.0	0.000000
4	157	3	1	0	0	1	1	2011	30	4	4	2	0.00	0.0	0.000000	0.313939	1.343737	1.247311	0.0	0.0	0.000000
5	158	3	1	0	0	1	1	2011	30	4	4	2	1.56	0.0	0.000000	0.313939	1.343737	1.247311	0.0	0.0	0.000000

Рис. 2.20. Датасет навчання регресійної моделі (вивід фреймворку візуалізації)

У даній роботі буде проведено один експеримент на дві моделі машинного навчання: лінійна регресія (LinearRegression) та легкий градієнтний бустінг (LightGBM).

Лінійна регресія виступає в ролі моделі апроксиматичного класу моделей, а градієнтний бустінг – деревовидних ансамблів. Також враховуючи те, що LightGBM на сьогоднішній день є найбільш точним алгоритмом машинного навчання серед більшості датасетів та задач пов’язаних з табличними даними, то дана комбінація двох моделей дає досить точне уявлення про ефективність гіпотези з використанням латентного простору автокодувальника.

На рис. 2.21. зображено код навчання лінійної регресії.

```
model = LinearRegression()
model = model.fit(train_x, train_y)
```

Рис. 2.21. Навчання лінійної регресій (програмний код)

На рис. 2.22. зображено код для навчання градієнтного бустінгу.

```
params = {
    'boosting_type': 'gbdt',
    'num_leaves': 64,
    'max_depth': 10,
    'learning_rate': 0.01,
    'n_estimators': 1000,
    'random_state': 1,
    'n_jobs': -1,
    'lambda_l1': 1.0,
}

model = lightgbm.LGBMRegressor(**params)
model = model.fit(train_x, train_y, eval_set=(val_x, val_y),
                  categorical_feature=cat_cols,
                  early_stopping_rounds=50)
```

Рис. 2.22. Навчання градієнтного бустінгу (програмний код)

3 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТІВ

3.1 Результати навчання автокодувальника

Навчання автокодувальника виявилась дуже складною задачею. У табл. 3.1. приведені результати навчання автокодувальника використовуючи метрику MSE.

Таблиця 3.1

Результати навчання автокодувальника за MSE

Датасет	MSE
Тренувальний	0.0115
Валідаційний	0.02
Тестовий	0.021

Як можна бачити з результатів навчання, автокодувальник має дещо гірші результати на валідаційному на тестовому датасетах.

Візуально результати навчання на випадковому продукті зображено на рис. 3.1.

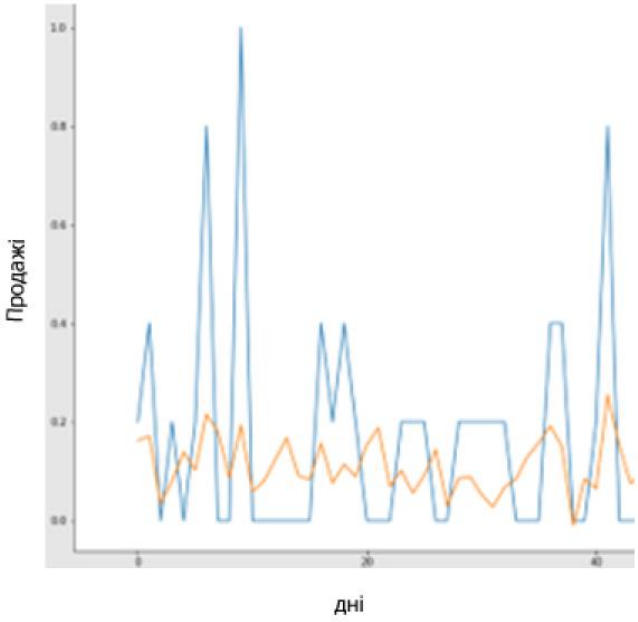


Рис. 3.1. Порівняння оригінального на відновленого рядів

Загалом видно, що автокодувальник не зміг досить точно відновити часовий ряд, але на графіку чітко видно ознаки кореляції між двома рядами.

Отримані латентні вектори будуть далі використані як додаткові ознаки при побудові регресійної моделі. Всі 8 векторів візуально продемонстровані на рис. 3.2., рис.3.3., рис. 3.4., рис. 3.5., рис. 3.6., рис. 3.7., рис. 3.8. та рис. 3.9.

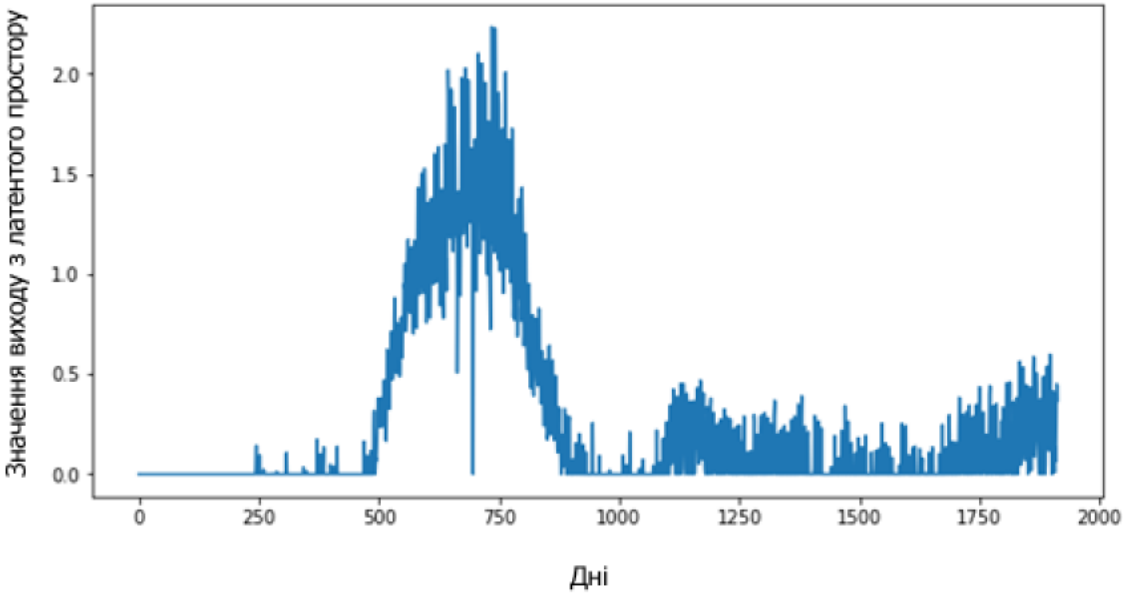


Рис. 3.2. Латентний вектор 1

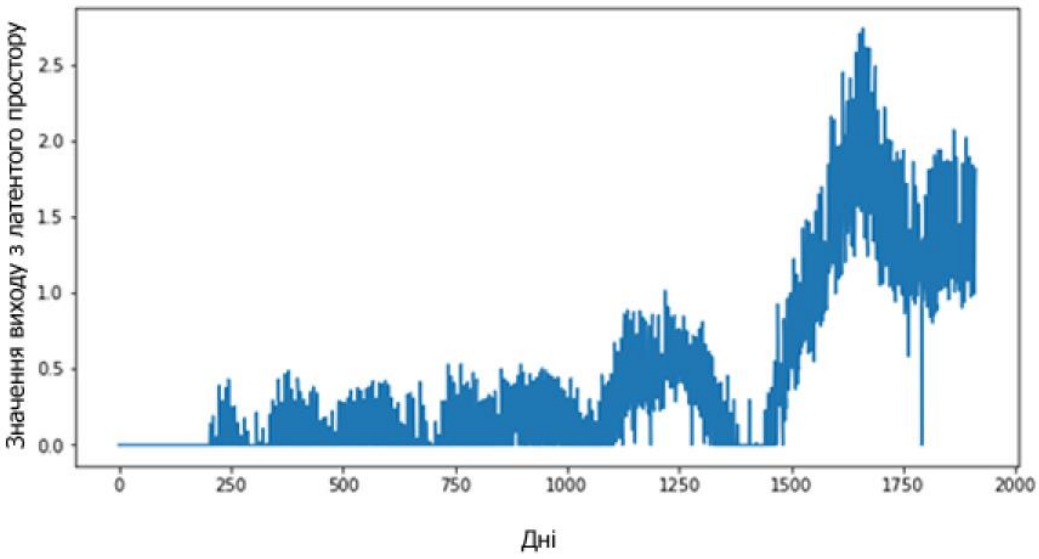


Рис. 3.3. Латентний вектор 2

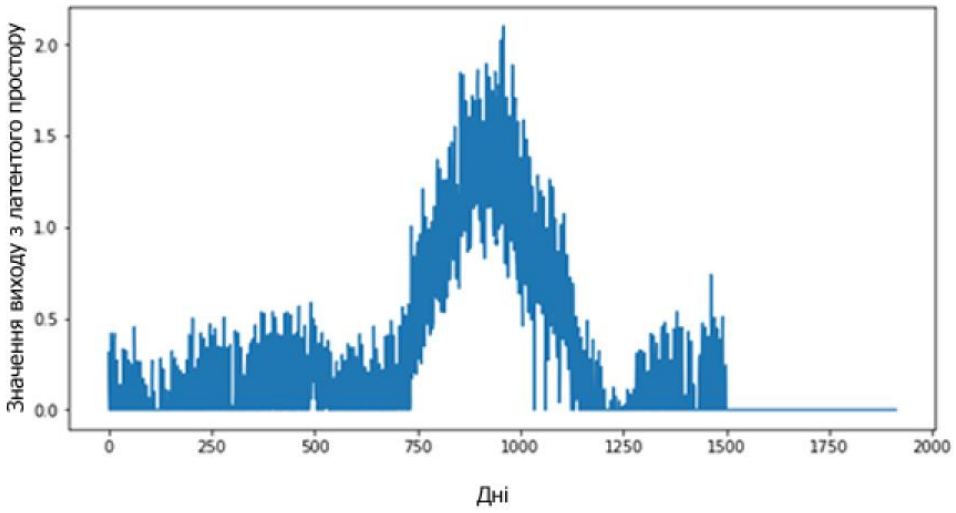


Рис. 3.4. Латентний вектор 3

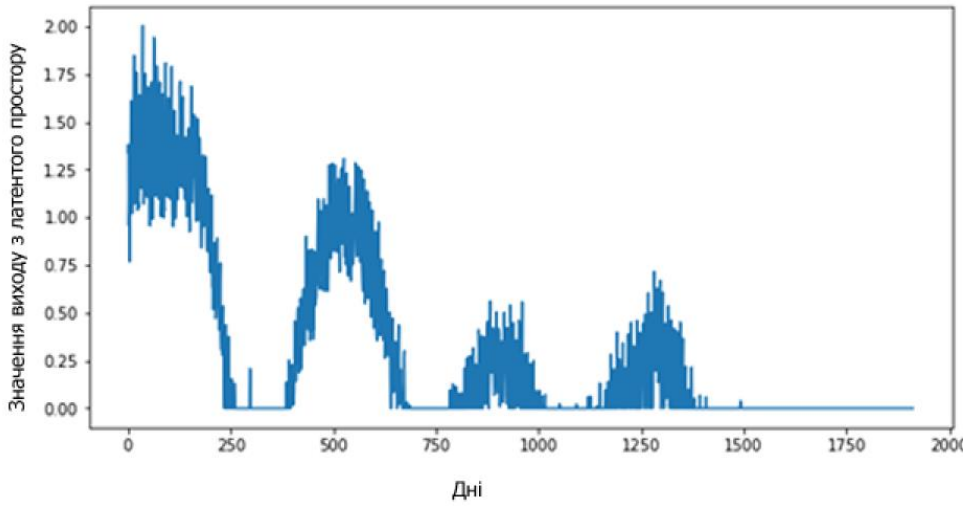


Рис. 3.5. Латентний вектор 4

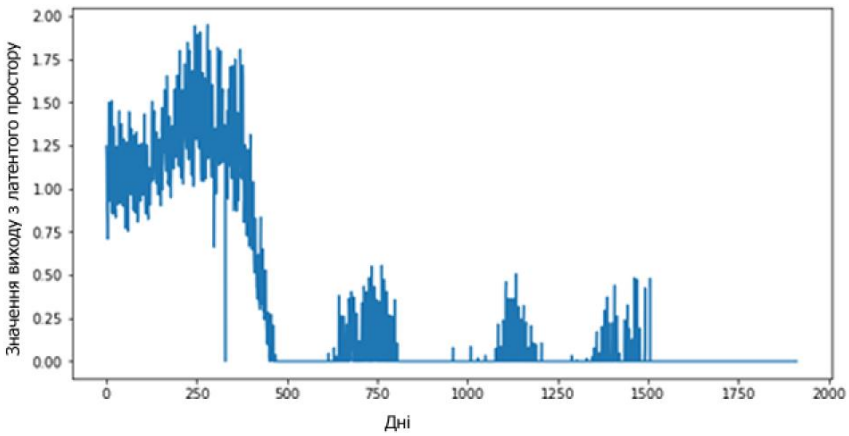


Рис. 3.6. Латентний вектор 5

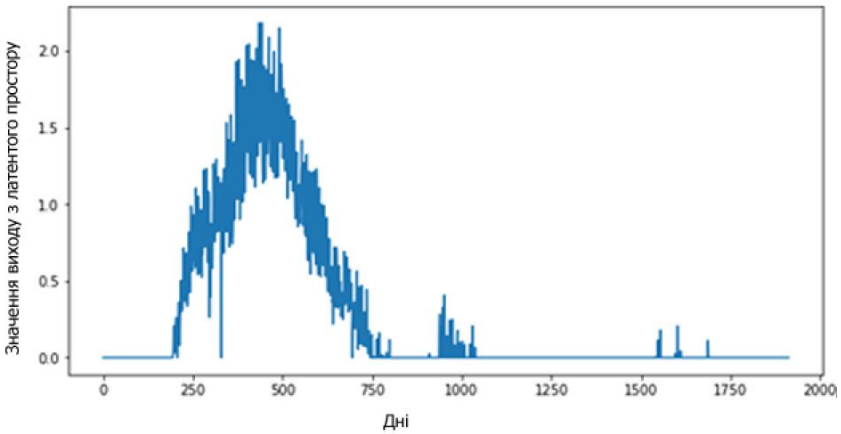


Рис. 3.7. Латентний вектор 6

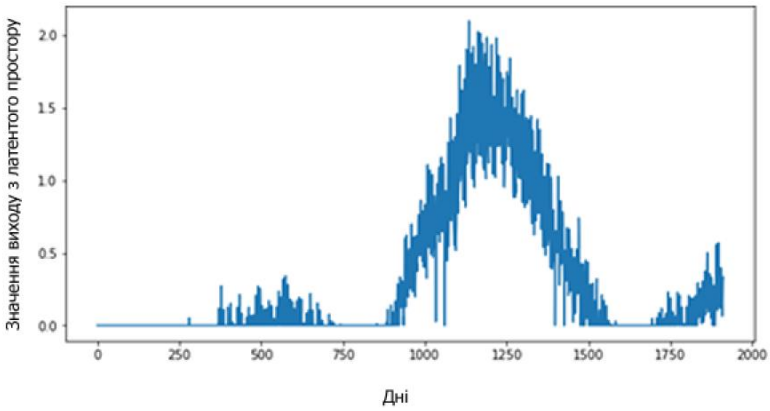


Рис. 3.8. Латентний вектор 7

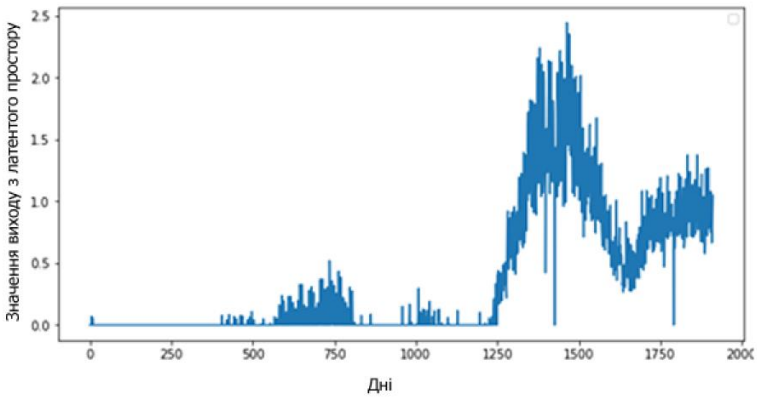


Рис. 3.9. Латентний вектор 8

3.2 Результати навчання регресійних моделей

Усього проведено два експеримента: один без прихованих векторів та один разом з прихованими векторами. У експериментах прийняли участь два алгоритма: лінійна регресія (LinearRegression) та легкий градієнтний бустінг (LightGBM).

Суть регресійної задачі у тому, щоб спрогнозувати продажі кожної із пар магазин-продукт в на рівні часової агрегації – день. Загалом горизонт прогнозування становить 1 день. Тобто алгоритмам необхідно спрогнозувати продажі для пари магазин-продукт на один день вперед. Суть прогнозного горизонту відносно історичних продаж зображено на рис. 3.10.

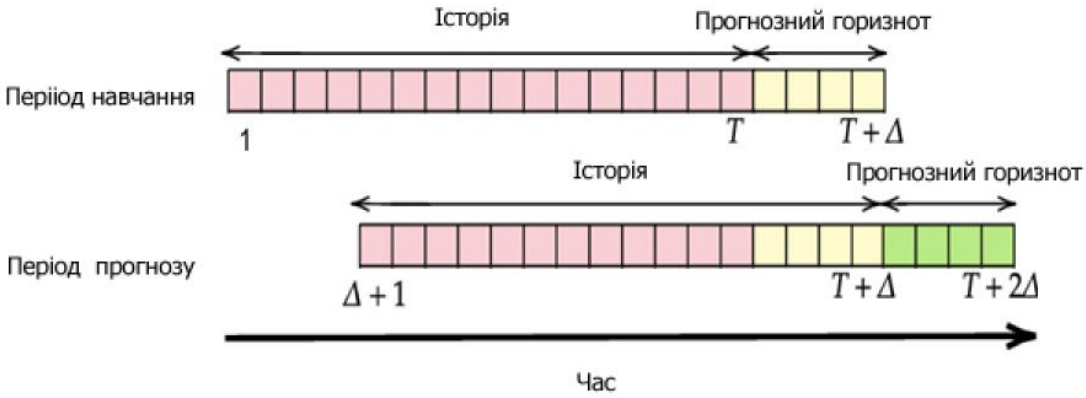


Рис. 3.10. Прогнозний горизонт

Результати експериментів без прихованих векторів наведені у табл. 3.2. для лінійної регресії та у табл. 3.3. для LightGBM.

Таблиця 3.2

Результати експерименту Лінійної регресії без прихованих векторів

Dataset	MAE	MSE	MAPE	R-Squared
Train	1.4507	10.681	-	0.029
Validation	1.4377	6.4696	-	0.019
Test	1.4707	6.8927	-	0.036

Таблиця 3.3

Результати експерименту LightGBM без прихованих векторів

Dataset	MAE	MSE	MAPE	R-Squared
Train	0.8852	3.3812	-	0.693
Validation	1.1134	3.6755	-	0.443
Test	1.0788	3.4064	-	0.524

По результатах експерименту на датасеті без латентних векторів можна побачити, що лінійна регресія справилась сильно гірше ніж градієнтний бустінг. У прогнозуванні попиту існує тенденція щодо домінації у точностях саме деревовидних моделей, тому сильний відрив у LightGBM не несе неочікуваний характер.

Результати експериментів разом з прихованими векторами наведені у табл. 3.4. для лінійної регресії та у табл. 3.5. для LightGBM.

Таблиця 3.4

Результати експерименту Лінійної регресії разом з прихованими векторами

Dataset	MAE	MSE	MAPE	R-Squared
Train	1.4301	9.351	-	0.033
Validation	1.4477	6.8696	-	0.018
Test	1.4656	6.5964	-	0.035

Таблиця 3.5

Результати експерименту LightGBM разом з прихованими векторами

Dataset	MAE	MSE	MAPE	R-Squared
Train	0.8280	3.1810	-	0.71
Validation	1.0436	3.4194	-	0.48
Test	1.0360	3.3488	-	0.53

Із результатів експерименту видно, що лінійна регресія слабко відреагувала на нові фактори, тому можна зробити висновок, що даний алгоритм не зміг у достатній мірі покращити точність прогнозу.

У випадку з другим експериментом, LightGBM показав невеликий приріст у точностях, але достатній щоб зробити висновок, що латентні вектори зіграли позитивну роль у прогнозуванні попиту. Порівняльні значення змін у метриках після експерименту з прихованими векторами для лінійної регресії представлені у табл. 3.6, а для LightGBM у табл. 3.7.

Таблиця 3.6

Зміни в похибках для Лінійної регресії (%)

Dataset	MAE	MSE	MAPE	R-Squared
Train	-1.42	-12.4	-	13.8
Validation	0.695	6.2	-	-5.3
Test	-0.34	-4.2	-	2.8

Таблиця 3.7

Зміни в похибках для LightGBM (%)

Dataset	MAE	MSE	MAPE	R-Squared
Train	-6.4	-5.9	-	2.4
Validation	-6.2	-6.97	-	8.35
Test	-3.9	-1.69	-	1.5

Як можна бачити LightGBM має сильно кращі прирости у точності та більш стабільні ніж лінійна регресія. Також слід зауважити, щоб за результатами експериментів не вдалося заспокоїти MAPE, бо через наявність нулів у цільовій змінній не вдалося отримати метрики.

На рис. 3.11. зображено історичні продажі (синім) та прогноз (помаранчевим) для лінійної регресії та на рис. 3.12. для LightGBM.

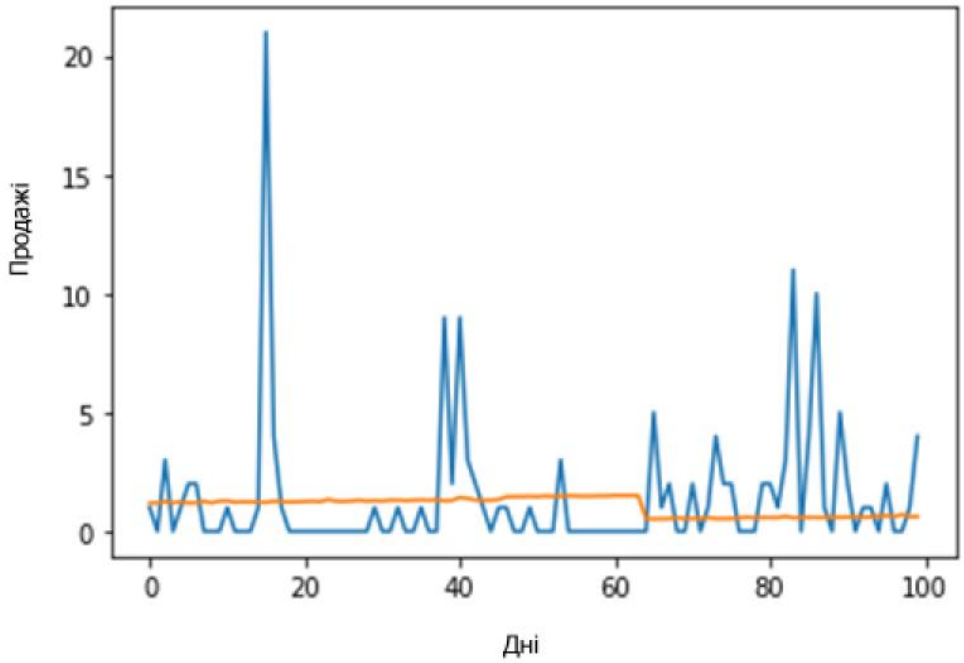


Рис. 3.11. Прогноз лінійною регресією (помаранчевий)

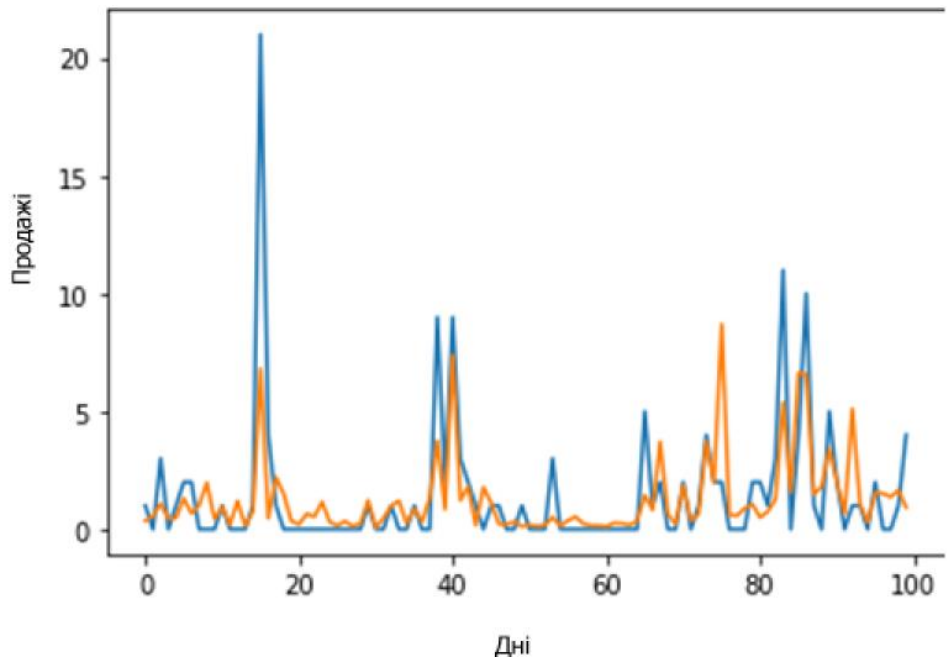


Рис. 3.12. Прогноз LightGBM (помаранчевий)

На рис. 3.13. зображено важливість показників для LightGBM. Як можна бачити приховані вектори (EV_1, EV_2, EV_3, EV_4, EV_5, EV_6, EV_7, EV_8) мають доволі сильний вплив на прийняття рішень алгоритмом.

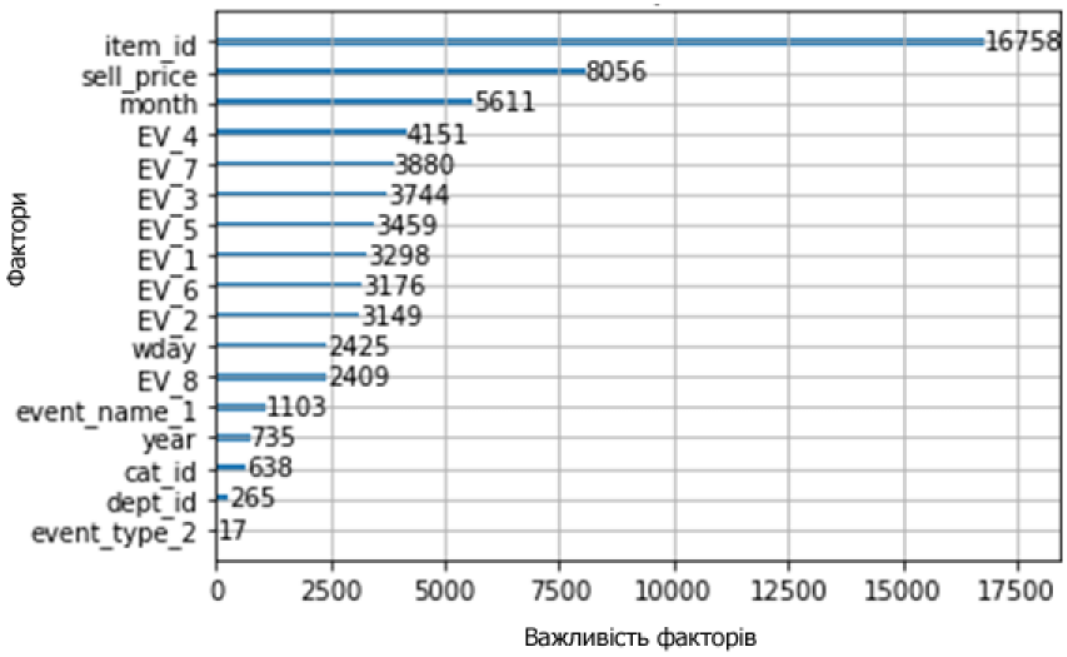


Рис. 3.13. Важливість показників для LightGBM (вивід фреймворку)

Отже, приховані вектори, що були отримані з автокодувальника, доволі слабо, але підвищили точність прогнозування попиту.

3.3 Рекомендації до отриманих результатів

Проведені експерименти показали приріст у точності прогнозування попиту. Слід зазначити, що існують ряд шляхів щодо теоретичного покращення точності прогнозування.

Представлений у роботі автокодувальник має просту архітектуру. Збільшення кількості вагів та шарів, вдосконалення структурного розміщення вузлів у нейронній мережі та інші екстенсивні методи більш ніж можуть покращити якість стиснутого латентного простору.

Однак, окрім стандартних методів покращення архітектури, існують більш потужні покращення - варіаційний автокодувальник.

Регулярність латентного простору для автокодерів є складним моментом, який залежить від розподілу даних у початковому просторі, розмірності прихованого простору та архітектури кодувальника. Таким чином, досить важко (якщо взагалі неможливо) гарантувати, апріорі, що кодувальник організує прихований простір розумним чином, сумісним із генеративним процесом.

Високий ступінь свободи автокодера, що дозволяє кодувати та декодувати без втрати інформації (незважаючи на низьку розмірність латентного простору), призводить до серйозного переобладнання, що означає, що деякі точки прихованого простору дадуть безглуздий вміст. Якщо цей одновимірний приклад був добровільно обраний як досить екстремальний, ми можемо помітити, що проблема регулярності прихованого простору автокодерів є набагато більш загальною і заслуговує на особливу увагу.

Отже, щоб мати можливість використовувати декодер нашого автокодера для генеративної мети, ми повинні бути впевнені, що прихований простір достатньо регулярний. Одним з можливих рішень для отримання такої регулярності є введення явної регуляризації під час навчального процесу. Таким чином, як ми коротко згадували у вступі до цієї публікації, варіаційний автокодер можна визначити як автокодер, навчання якого упорядковано, щоб уникнути перенавчання та гарантувати, що латентний простір має хороші властивості, які дозволяють генерувати процес.

Як і стандартний автокодер, варіаційний автокодер - це архітектура, що складається з кодера і декодера, яка навчена мінімізувати помилку реконструкції між закодованими-декодованими даними та вихідними даними. Однак, щоб ввести деяку регуляризацію прихованого простору, ми переходимо до невеликої модифікації процесу кодування-декодування: замість кодування вхідних даних як єдиної точки, ми кодуємо його як розподіл у прихованому просторі.

Потім модель навчається наступним чином:

- по-перше, вхідні дані кодуються як розподіл у прихованому просторі
- по-друге, з цього розподілу відбирається точка з латентного простору
- по-третє, вибіркова точка декодується, і помилка реконструкції може бути обчислена

Візуально різницю між звичайним автокодувальником та варіаційним продемонстровано на рис. 3.14.

Рис. 3.14. Різниця між простим та варіаційним автокодувальником

На практиці закодовані розподіли вибираються за нормальним розподілом, щоб кодер можна було навчити повертати середнє значення та матрицю коваріації. Причина, чому вхідні дані кодуються як розподіл з певною дисперсією замість однієї точки, полягає в тому, що це дає можливість дуже природно виразити регулярізацію прихованого простору: розподіли, які повертає кодер, змушені бути близькими до стандартного нормального розподілу.

Таким чином, функція втрат, яка мінімізується під час навчання, складається з «реконструкції», який прагне зробити схему кодування-декодування якомога ефективнішою, і «регуляризації», який має тенденцію регулювати організацію прихованого простору, роблячи розподіли, які повертає кодер, близькими до стандартного нормального розподілу. Цей термін регуляризації виражається як розбіжність Кульбака-Лейблера між поверненим розподілом і стандартним гауссовим. Ми можемо помітити, що розбіжність Кульбака-

Лейблера між двома гауссовими розподілами має замкнену форму, яка може бути безпосередньо виражена в термінах середніх і коваріаційних матриць двох розподілів (рис. 3.15.).

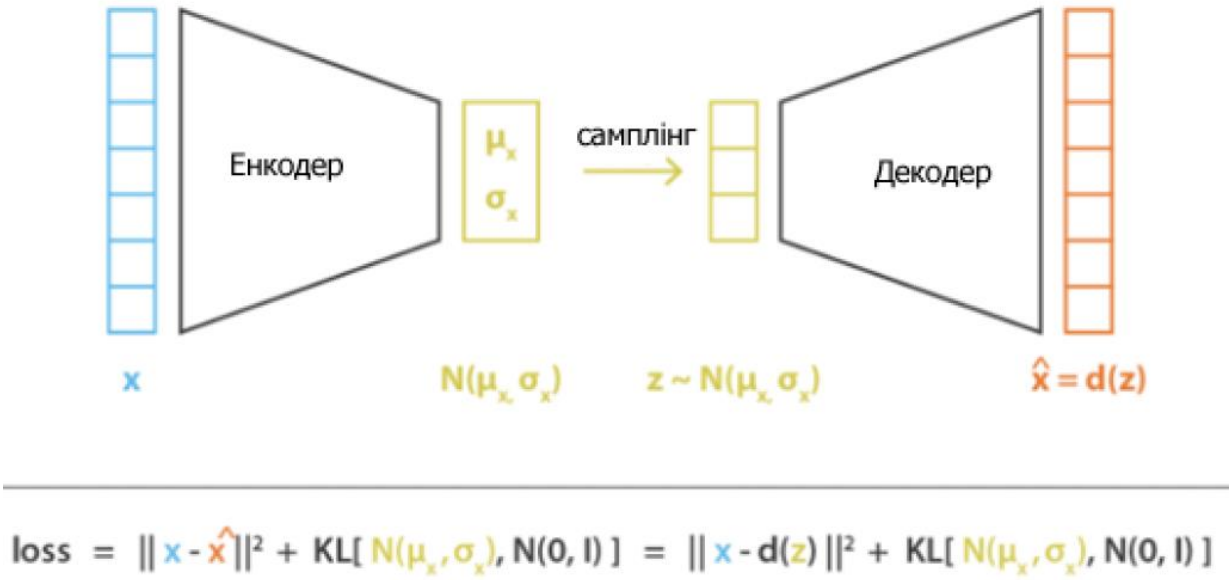


Рис. 3.15. Варіаційний автокодувальник

Таким чином, варіаційні автокодери (VAE) — це автокодери, які вирішують проблему нерівномірності латентного простору, змушуючи кодувальник повертати розподіл по прихованому простору замість однієї точки та додаючи до функції втрат термін регуляризації над цим поверненим розподілом, щоб забезпечити кращу організацію латентного простору [12].

ВИСНОВКИ

У роботі розкрито питання прогнозування попиту та продуктової канібалізації. Проведено аналіз архітектур нейронних мереж та автокодувальника. Отримано точності реконструкції часових рядів автокодувальником та похибки прогнозування попиту регресійними моделями.

У ході виконання даної роботи були досягнуті наступні результати:

– досліджено питання продуктової канібалізації;

– розкрито питання прогнозування попиту;

– досліджено принцип роботи автокодувальника;

– отримано набір прихованих векторів після навчання автокодувальника;

– проведено навчання регресійних моделей;

– отримано точності прогнозування попиту з та без прихованих векторів;

– з’ясовано позитивний вплив на точність прогнозування з використанням прихованих векторів;

– досліджено шляхи вдосконалення системи прогнозування попиту на основі прихованих векторів автокодувальника.

Завдяки отриманим результатам можна підсумувати, що використання прихованих векторів, що були отримані після навчання автокодувальника, позитивно впливають на точність прогнозування попиту. Вектори отримані шляхом стиснення продуктово-канібалізаційної інформації у вигляді історичних продажів. Серед регресійних алгоритмів саме LightGBM зміг найбільш сильно вловити корисні взаємозв’язки з прихованих векторів. Також досліджено шляхи вдосконалення автокодувальника, а значить ще більше покращення точності прогнозування.

Отже, використання даного підходу є практично значимим елементом у системах прогнозування попиту.

ПЕРЕЛІК ПОСИЛАНЬ

1. Electricity Demand and Energy Consumption Management System. URL: https://arxiv.org/abs/0809.2421 (дата звернення: 09.11.2021).
2. An XGBoost-Based Forecasting Framework for Product Cannibalization. URL: https://arxiv.org/abs/2111.12680 (дата звернення: 10.11.2021).
3. A Characterization of Mean Squared Error for Estimator with Bagging URL: https://arxiv.org/abs/1908.02718 (дата звернення: 10.11.2021).
4. Performance Metrics (Error Measures) in Machine Learning Regression, Forecasting and Prognostics: Properties and Typology. URL: https://arxiv.org/ftp/arxiv/papers/1809/1809.03006.pdf (дата звернення: 10.11.2021).
5. Mean Absolute Percentage Error for regression models. URL: https://arxiv.org/abs/1605.02541 (дата звернення: 10.11.2021).
6. Small Is Beautiful: The Use and Interpretation of R2 in Social Research. URL: https://statisticsbyjim.com/regression/interpret-r-squared-regression/ (дата звернення: 10.11.2021).
7. A Beginner's Guide to Neural Networks and Deep Learning. URL: https://www.researchgate.net/publication/242329609_Small_Is_Beautiful_The_Use_and_Interpretation_of_R2_in_Social_Research (дата звернення: 10.11.2021).
8. Goodfellow, Ian; Bengio, Yoshua; Courville, Aaron (2016). "6.5 Back-Propagation and Other Differentiation Algorithms". Deep Learning. MIT Press. с. 200–220. (дата звернення: 10.11.2021).
9. An Introduction to Variational Autoencoders. URL: https://arxiv.org/abs/1906.02691 (дата звернення: 10.11.2021).

10. II МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ «TELECOMMUNICATION: PROBLEMS AND INNOVATION. Pages 11 - 12. URL: http://www.dut.edu.ua/uploads/l_2187_52947573.pdf (дата звернення: 10.11.2021).
11. M5 Forecasting – Accuracy. URL: https://www.kaggle.com/c/m5-forecasting-accuracy/overview (дата звернення: 10.11.2021).
12. Auto-Encoding Variational Bayes. URL: https://arxiv.org/abs/1312.6114 (дата звернення: 11.11.2021).

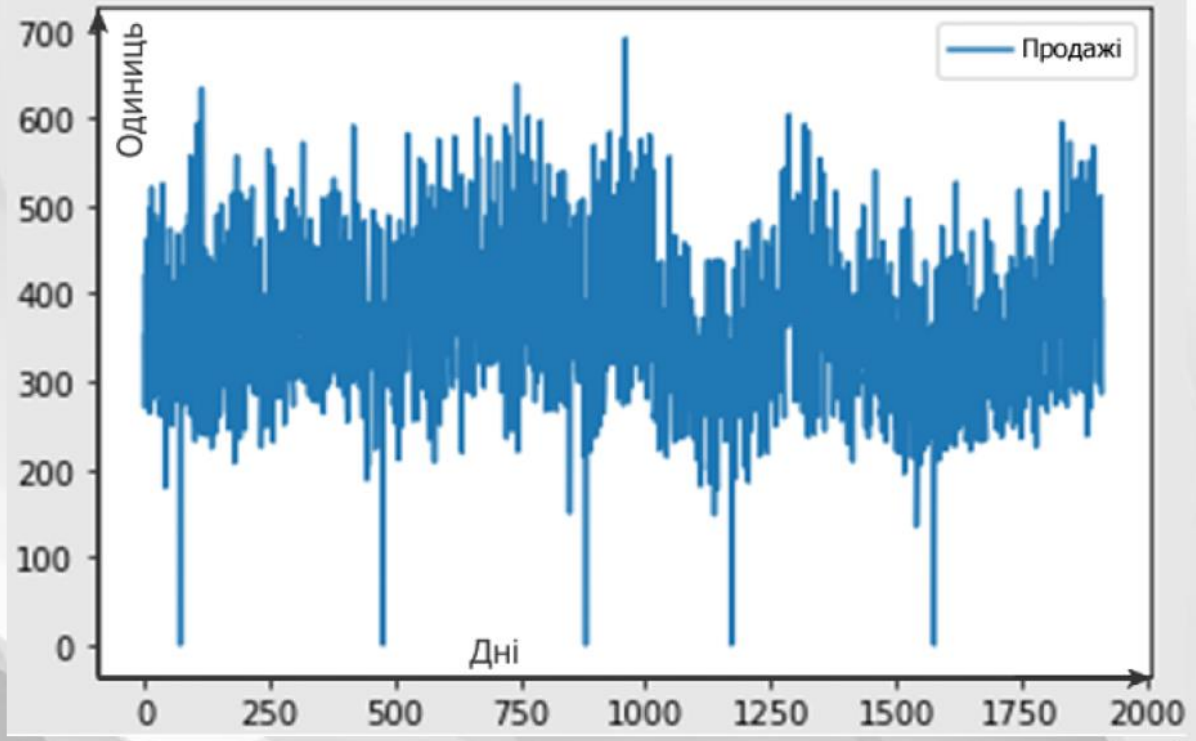
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

Об’єкт, предмет, мета роботи

- ***Об’єкт дослідження*** – застосування автокодувальників у прогнозуванні попиту
- ***Предмет дослідження*** – стискання інформації автокодувальниками
- ***Мета роботи*** – отримання більш високої точності прогнозування попиту з використанням автокодувальників, аніж стандартними підходами

2

Прогнозування попиту

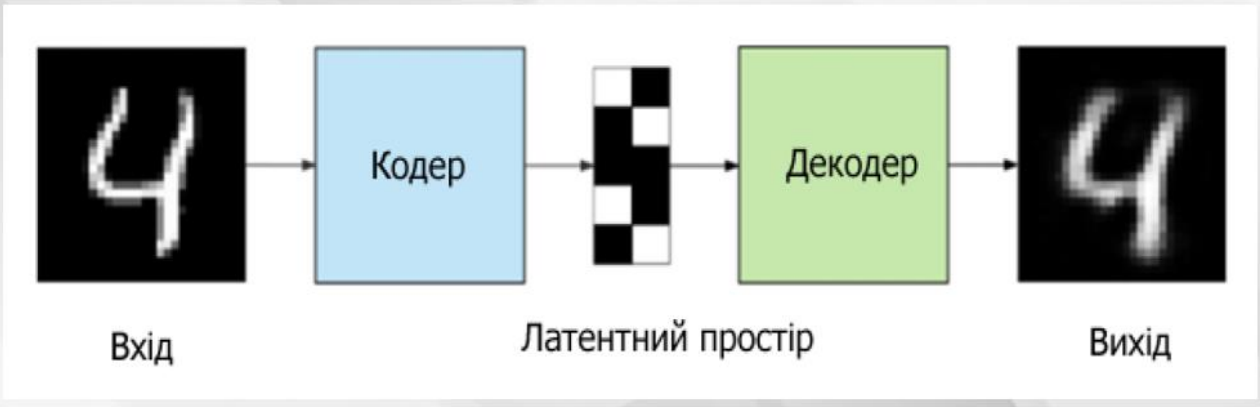


3

Продуктова канібалізація

4

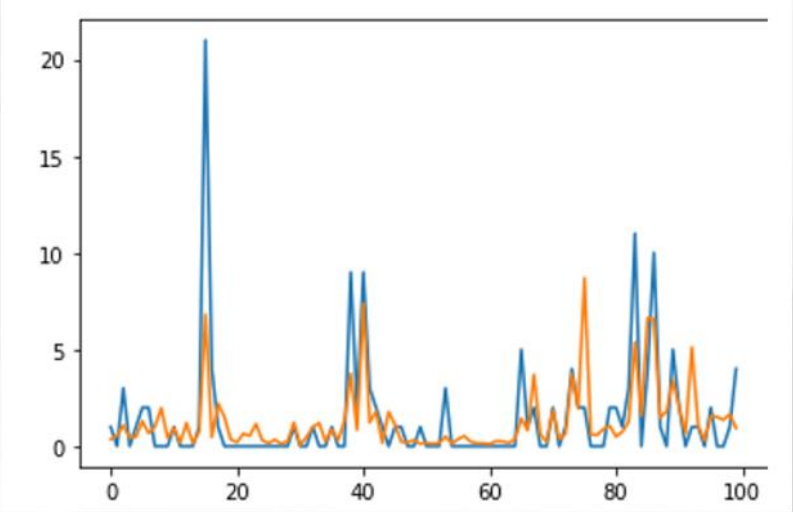
Автокодувальники



5

Регресійні алгоритми

- Linear regression
- LightGBM



6

Результати

Таблиця 3.6

Зміни в похибках для Лінійної регресії (%)

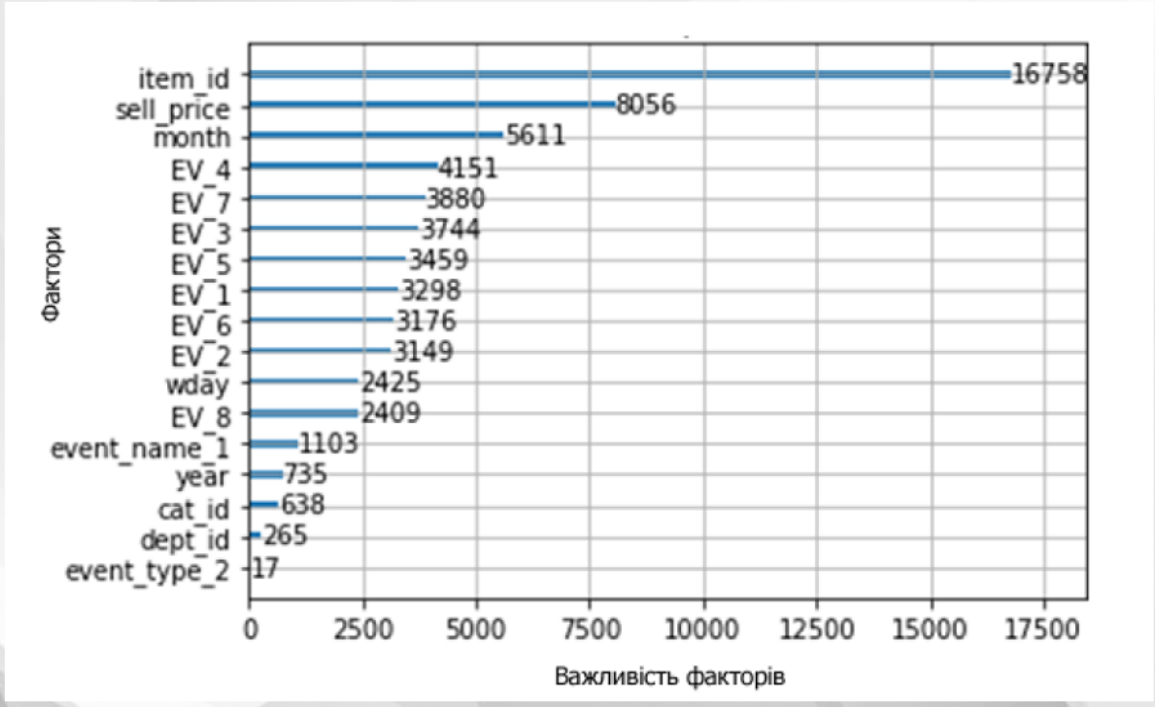
Dataset	MAE	MSE	MAPE	R-Squared
Train	-1.42	-12.4	-	13.8
Validation	0.695	6.2	-	-5.3
Test	-0.34	-4.2	-	2.8

Таблиця 3.7

Зміни в похибках для LightGBM (%)

Dataset	MAE	MSE	MAPE	R-Squared
Train	-6.4	-5.9	-	2.4
Validation	-6.2	-6.97	-	8.35
Test	-3.9	-1.69	-	1.5

Важливість факторів



Висновки

- Завдяки отриманим результатам можна підсумувати, що використання прихованих векторів, що були отримані після навчання автокодувальника, позитивно впливають на точність прогнозування попиту. Вектори отримані шляхом стиснення продуктово-канібалізаційної інформації у вигляді історичних продажів. Серед регресійних алгоритмів саме LightGBM зміг найбільш сильно вловити корисні взаємозв'язки з прихованих векторів. Також досліджено шляхи вдосконалення автокодувальника, а значить ще більше покращення точності прогнозування

Список публікацій

- «Особливості використання автокодувальників для зниження розмірності даних».
  Назва конференції: II МІЖНАРОДНА НАУКОВО-ПРАКТИЧНА КОНФЕРЕНЦІЯ «TELECOMMUNICATION: PROBLEMS AND INNOVATION»
  Цапро І.В. – ДУТ, 2021 – укр– 11-12с.

- «Згорткові нейромережі та механізм уваги в задачах прогнозування багатовимірних часових рядів».
  Назва конференції: XIII МІЖНАРОДНА НАУКОВО-ТЕХНІЧНА КОНФЕРЕНЦІЯ СТУДЕНТСТВА ТА МОЛОДІ «СВІТ ІНФОРМАЦІЇ ТА ТЕЛЕКОМУНІКАЦІЙ»
  Цапро І.В. – ДУТ, 2021

- «Особливості отримання прихованих просторових уявлень автокодувальника у задачах прогнозування попиту».
  Назва конференції: СИСТЕМНИЙ АНАЛІЗ В БІЗНЕСІ ТА УПРАВЛІННІ
  Матеріали І Всеукраїнської науково-практичної конференції
  Цапро І.В. – ДУТ, 2021