Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

АТЕСТАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти – другий (магістерський)

Дослідження нейромережевого підходу для прогнозування поведінки користувачів у WEB системах
(тема)

Виконав: студент 2 курсу, групи ІПЗм-18-1
Доротенко О.В.
(прізвище, ініціали)

спеціальності 121– Інженерія програмного забезпечення
(код і повна назва спеціальності)
Освітньо-наукової програми
(тип програми)
Інженерія програмного забезпечення
(тип програми)

Керівник к.т.н., доц. Лановий О.Ф.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Дудар З.В.
(підпис) (прізвище, ініціали)

2020 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
Кафедра Програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)
Тип програми освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА АТЕСТАЦІЙНУ РОБОТУ

Студентові Доротенку Олександру Вадимовичу
(прізвище, ім'я, по батькові)

1. Тема роботи (проекту): Дослідження нейромережевого підходу для прогнозування поведінки користувачів у WEB системах
затверджена наказом по університету від "27" березня 2020 р. №473 Ст
2. Термін подання студентом роботи (проекту)
10 травня 2020 р.
3. Вихідні дані до роботи (проекту) набір даних з вільною ліцензією, скрипти для аналізу набору даних, алгоритми аналізу та класифікації даних. Використовувати ОС Linux, мову програмування Python та Java.

4. Перелік питань, що потрібно опрацювати в роботі: мета роботи, аналіз проблемної галузі і постановка задачі, застосування методів для поставленої задачі, пошук способів підвищення якості та аналіз їх ефективності.

(Наступний аркуш завдання)

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка*
1.	Аналіз предметної галузі	03 квітня 2020 р.	
2.	Огляд існуючих методів	10 квітня 2020 р.	
3.	Дослідження методів класифікації томографічних зображень	17 квітня 2020 р.	
4.	Підготовка пояснювальної записки	24 квітня 2020 р.	
5.	Спецчастина	29 квітня 2020 р.	
6.	Підготовка презентації та доповіді	30 квітня 2020 р.	
7.	Попередній захист	04 травня 2020 р.	
8.	Нормоконтроль, рецензування	06 травня 2020 р.	
9.	Занесення диплома в електронний архів	08 травня 2020 р.	
10.	Допуск до захисту у зав. кафедри	10 травня 2020 р.	
* заповнюється вручну після виконання чергового пункту			

Дата видачі завдання 27 березня 2020 р.

Студент_____
(підпис)

Керівник роботи (проекту) _____ к.т.н., доцент Лановий О.Ф.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Атестаційна робота магістра містить: 59 с., 17 рис., 4 табл., 6 формул, 12 джерел.

АВТОКОДУВАЛЬНИКИ, АНАЛІЗ, КЛАСИФІКАЦІЯ, МЕРЕЖІ ГЛИБИННОГО НАВЧАННЯ, ПЕРЕДБАЧЕННЯ, РЕГУЛЯРІЗАЦІЯ.

Метою роботи є дослідження, аналіз та порівняння алгоритмів аналізу та класифікації у сфері передбачення дій користувача у WEB орієнтованих системах.

Нейромережеві методи розробки базуються на інструментах машинного навчання для мови Python з використанням фреймворку Keras. Також була застосована мова програмування Java та Spring Framework.

У результаті роботи було здійснено вимірювання ефективності використання алгоритмів машинного навчання та аналізу на різних наборах даних. Було проведене порівняння результатів роботи алгоритмів на основі машинного навчання та розробка програмного забезпечення яке використовує розглянутий алгоритм.

AUTOENCODERS, ANALYSIS, CLASSIFICATION, DEEP LEARNING NETWORKS, PREDICTION, REGULARIZATION.

The aim of this work is to research, analysis and compare algorithms for analysis and classification in case of user action prediction in WEB-based systems.

Implementation of neural networks methods are based on machine learning techniques for Python language with using Keras Framework. Also Java language was used with Spring Framework.

As a result, effectiveness analysis of using machine learning algorithms and analysis on different datasets were performed. Comparing results of algorithms execution and implementation of software which used proposed algorithm.

ЗМІСТ

ВСТУП

Всесвітня мережа (WWW) представляє багато комерційних можливостей, даючи користувачам інформацію для придбання товарів та або послуг. Відстеження поведінки або тенденцій користувачів дозволяє прогнозувати майбутні дії користувачів та можуть забезпечити більше можливостей для отримання користувачем актуальної для нього інформації.

Розуміння поведінки користувачів в Інтернеті викликає все більший інтерес для наукових дослідників у різних сферах. Традиційно, в галузі маркетингу, комерційні дослідницькі компанії вивчають поведінку споживачів, щоб зрозуміти, коли і де клієнти вирішують купувати продукцію. З цією метою веб-метрики окремих веб-сайтів служать докладним джерелом інформації про те, коли, як, якими розділами цікавиться користувач.

Прогнозування наміру користувача щодо певного продукту чи категорії продуктів на основі взаємодії в межах веб-сайту має вирішальне значення для сайтів електронної комерції та мереж показу реклами. Слідкуючи за моделями пошуку споживачів, інтернет-продавці можуть краще зрозуміти їх поведінку та наміри. А це, в свою чергу, допомагає підвищувати якість послуг які надає бізнес.

Все це стало ще більш актуальнішим через пандемію COVID-19. Так як багато бізнесу стикнулися з необхідністю починати або поліпшувати свою працю в онлайн режимі. Стало надзвичайно важливо боротися за увагу користувачів у мережі Інтернет. Так як зменшення кількості дій які повинен зробити користувач щоб отримати товар чи послугу це найефективніший шлях щоб привернути аудиторію, то передбачення нам ірів користувача стає доцільним шляхом для досягнення цієї мети.

Існує безліч алгоритмів та методів які використовуються для аналізу та прогнозування дій користувачів. У цій роботі ми використаємо деякі із них, включаючи нейромережеві алгоритми задля того щоб оцінити їх доцільність та ефективність.

Мета в цій роботі полягає у тому щоб розглянути алгоритми класифікації та аналізу даних у сфері аналізу та передбачення дій певних користувачів, класифікації їх з використанням алгоритмів машинного навчання, на основі отриманих результатів провести порівняння та зробити висновок щодо ефективності такого підходу.

Об’єктом дослідження даної роботи є поведінка користувачів у WEB-системах.

Аналіз дій користувачів за допомогою декількох алгоритмів допоможе наглядно оцінити та порівняти їх характеристики ефективності.

Отже, перед даною роботою поставлені наступні задачі:

- визначення найбільш влучних алгоритмів для передбачення дій користувача у WEB-системі;
- застосування обраних алгоритмів до навчальної вибірки;
- порівняння результатів роботи нейромережевих алгоритмів
- програмна реалізація застосунку.

Наукова новизна роботи полягає у порівнянні ефективності роботи нейромережевих алгоритмів для передбачення дій користувача у WEB-системах, обрання найефективніших.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз тенденції розвитку WEB-систем

Кожен день кількість користувачів всесвітньою мережею Інтернет невпинно зростає. На рисунку 1.1 наведено графік зростання користувачів в мережі Інтернет, який показує стрімкий зріст користувачей мережі, за даними ITU – міжнародної спілки електрозв'язку.

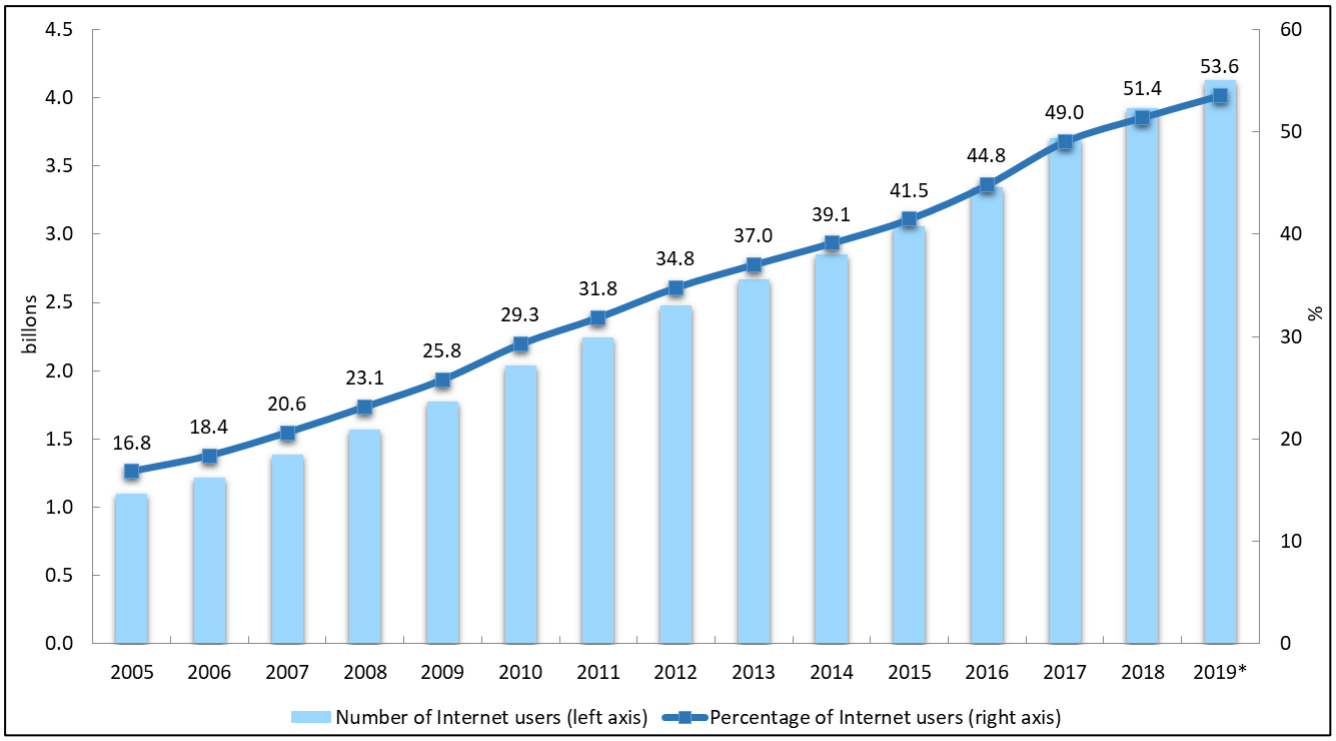

Рисунок 1.1 – Графік зростання користувачів Інтернет

В поточній ситуації вибухового зростання знань в мережі Інтернет користувачі змушені відфільтровувати все більше й більше інформації кожен день в пошуках потрібної. Стає складніше для користувачів знайти релевантну для себе інформацію. Аналіз і моделювання поведінки користувачів у WEB допомагає зрозуміти потреби онлайн користувачів, а розуміючи ці потреби можна передбачити наступні дії користувача та запропонувати йому потрібну для нього інформацію.

Передбачення – це інформація про певні події які повинні здійснитися через якийсь проміжок часу. В залежності від типу передбачень вони можуть бути як раціональними, так і ірраціональними. З давніх часів людство відчувало тягу до передбачень з різних причин. Це могли бути практичні причини, такі як пророкування погоди, війни, урожаю, тощо. Та навіть жага чогось містичного та неможливого.

В наші часи передбачення мають вже зовсім інший характер і вже зовсім не асоціюються з чимось містичним. Багато сучасних галузей бізнесу, промисловості, тощо в певній частині покладаються на передбачення. Господарська діяльність на прогнози погоди, інвестори на передбачення курсу акцій та валют, тощо.

Однією з найпоказових сфер, де передбачення дій користувача грає дуже важливу роль є сфера онлайн комерції. Онлайн-покупки або роздрібна торгівля через Інтернет – це форма електронної торгівлі, що дозволяє споживачам безпосередньо купувати товари або послуги у продавця через Інтернет за допомогою веб-браузера. Інтернет-магазин являє собою фізичну аналогію покупки товарів або послуг в торговому центрі. Цей процес називається інтернет-шопінгом за принципом "бізнес-споживач" (Business-to-P-consumer, B2C). У разі, коли підприємство купує товари в іншого підприємства, цей процес називається покупкою через Інтернет (B2B). Найбільшими з цих корпорацій, що займаються роздрібною торгівлею через Інтернет, є eBay і Amazon.com, розташовані в США. Успіх роздрібної торгівлі вже не обмежується тільки фізичними магазинами, це стає очевидним завдяки збільшенню числа рітейлерів, що пропонують споживачам інтерфейси для інтернет-магазинів. З ростом інтернет-магазинів з'являється безліч нових можливостей для охоплення ринку для магазинів, які можуть належним чином задовольнити потреби офшорного ринку і вимоги до обслуговування. Інтернет-магазини – це зростаюча область технологій. Створення магазину в Інтернеті дозволяє роздрібним торговцям розширити свій ринок і охопити споживачів, які в іншому випадку можуть не відвідати фізичний магазин. Головною визначною ознакою для споживачів є зручність здійснення покупок через Інтернет. Унікальні системи онлайн-платежів пропонують легку і безпечну

покупку у інших осіб. З перевагами покупок в Інтернеті також приходять й деякі потенційні ризики і небезпеки, про які повинні знати споживачі. В майбутньому ми можемо очікувати, що інтернет-магазини значно поліпшать свої технології, що дозволить їм здійснювати покупки легше й безпечніше. Зараз люди все частіше віддають перевагу купівлі товарів у онлайн магазині. На рисунку 1.2 ви можете бачити графік прибутку за рік онлайн-магазину Amazon.

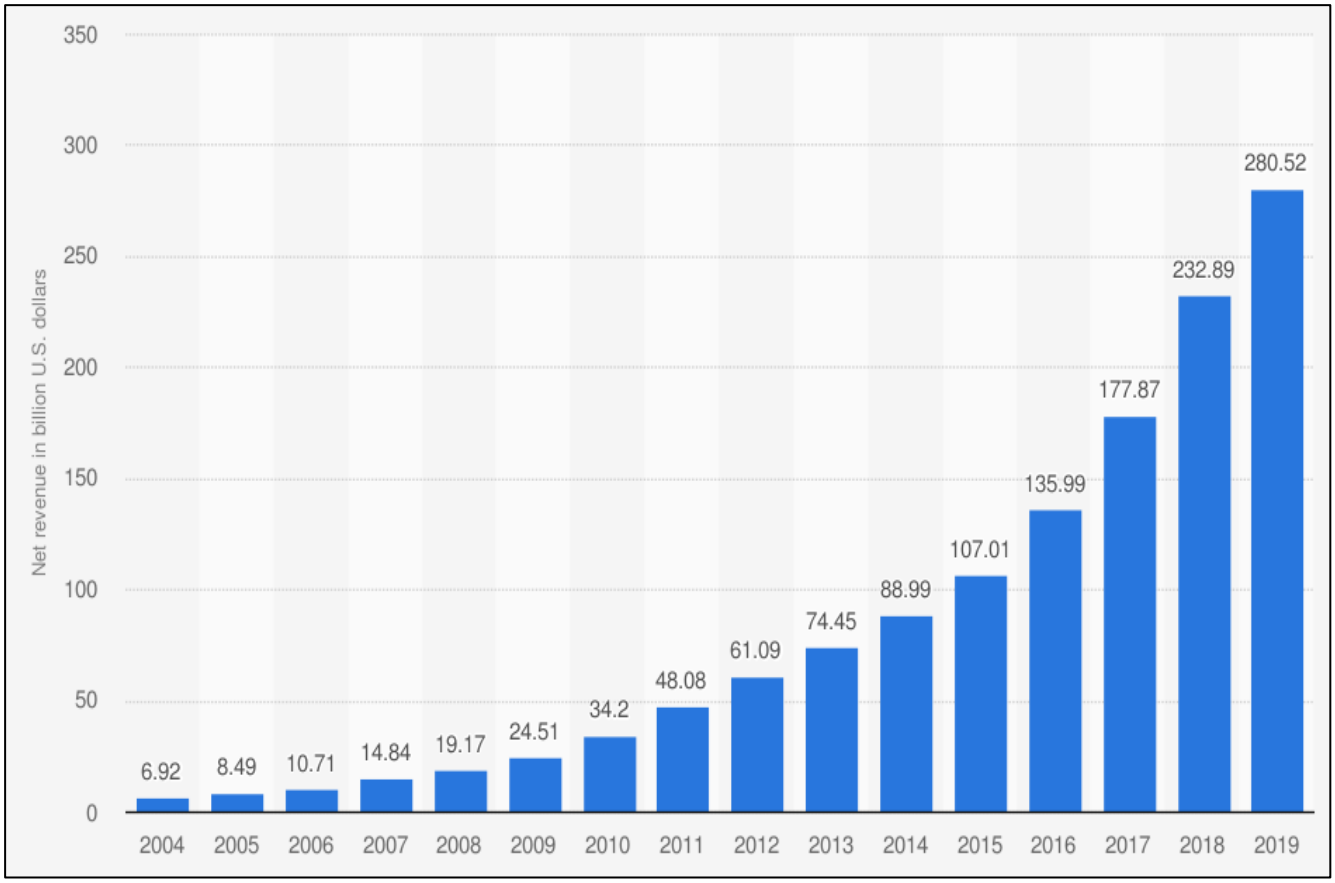

Рисунок 1.2 – Графік зростання прибутку Amazon

Ця тенденція в деякій частині полягає у тому, що покупці хочуть все менше й менше часу витрачати на такі речі як похід по магазинам. Тому для галузі онлайн комерції дуже важливо мати змогу передбачати які товари зацікавлять користувача у майбутньому. На рисунку 1.3 ви можете бачити один із прикладів де було б доцільно використати таку інформацію.

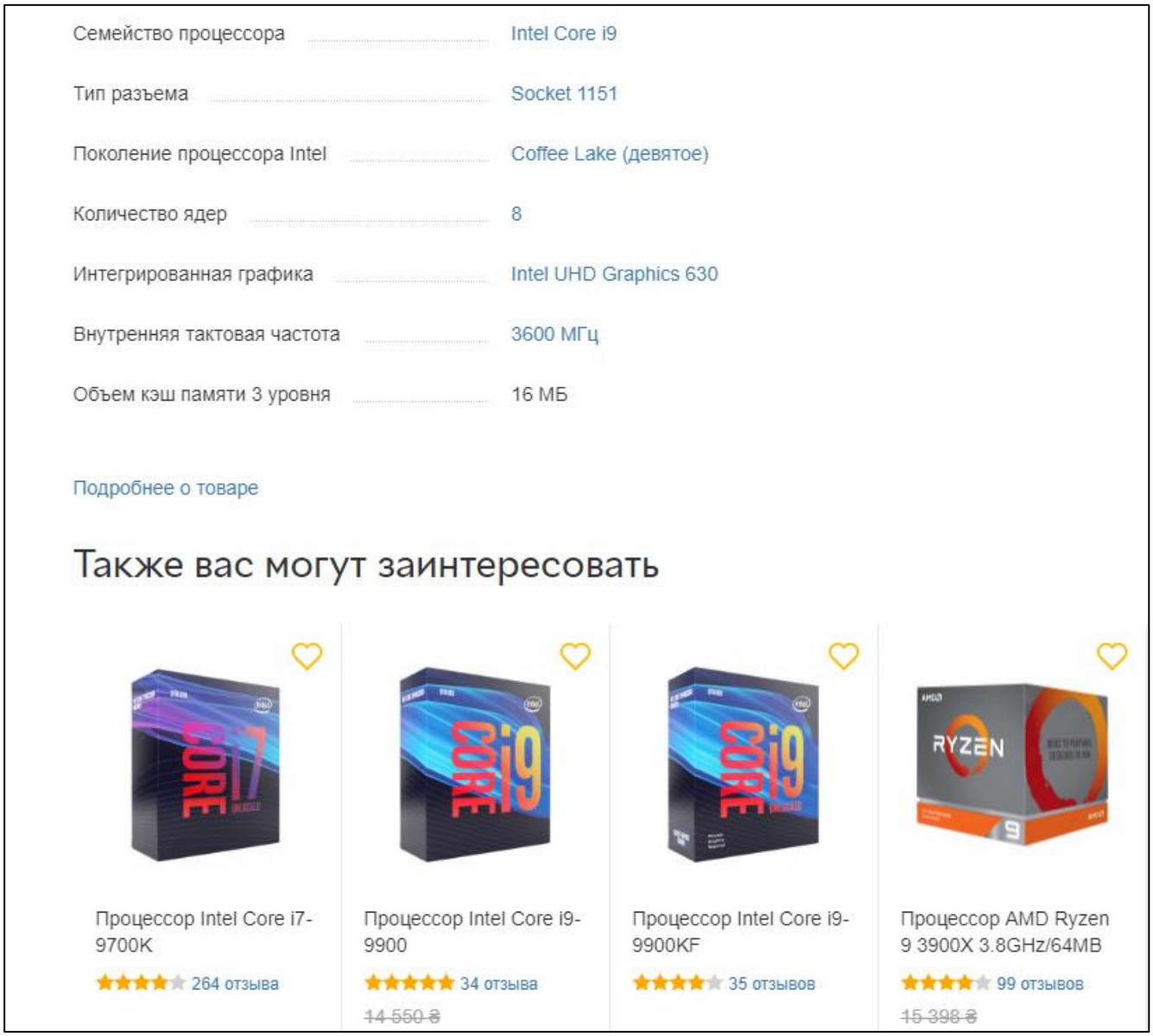

Рисунок 1.3 – Персональна вибірка товарів на сайті rozetka.com.ua

У сфері електронної комерції доступний величезний набір даних, і потенційні споживачі вивчають інформацію про товар, перш ніж приймати рішення про покупку, тим самим висловлюючи свої наміри придбати товар. Шукаючи товар, користувачі використовують різні схеми пошуку, таким чином мають різний час, витрачений на кожний предмет, частоту пошуку та кількість відвідувань. Сфера онлайн комерції також дуже показова тому що має безліч механізмів для відстеження дій користувача та отримання інформації про користувача.

1.2 Аналіз механізмів доступу до інформації про дії користувача

Почнемо з того, що всі взаємодії з веб-сайтами побудовані на базі HTTP-протоколу. Структура HTTP протоколу зображена на рисунку 1.4.

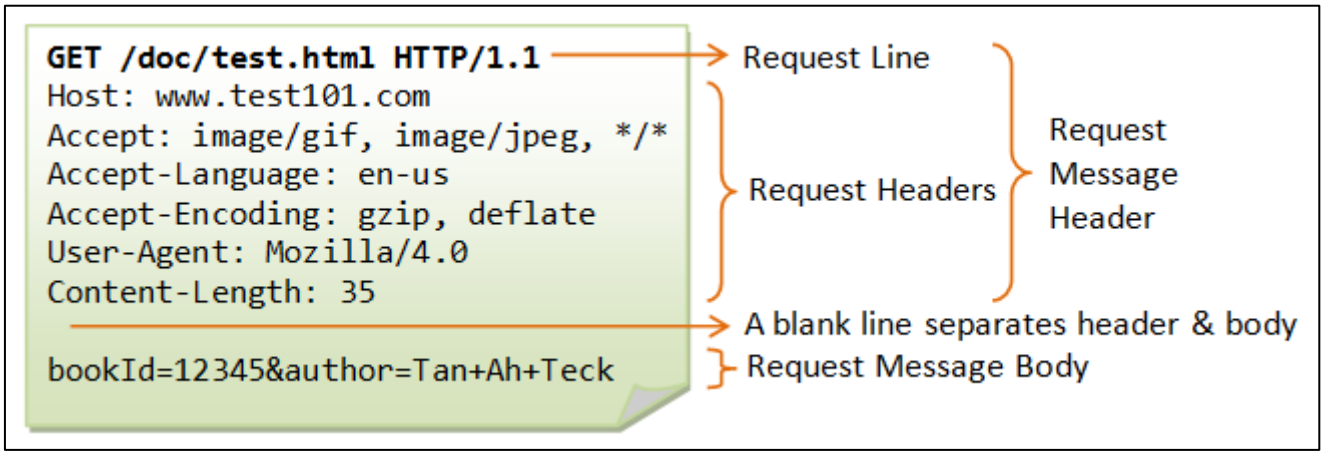

Рисунок 1.4 – Структура HTTP протоколу

Однією з частин HTTP запитів є так звані заголовки. Вони можуть містити інформацію про місцезнаходження користувача, його мовні преференції тощо.

Крім того всі дії користувача на веб-сайті можуть бути записані. Стандартними метриками записаними в логах як правило є наступні:

– HTTP помилки;

– тривалість сесії;

– геолокація;

– час найбільшої активності.

Є декілька відомих сервісів для автоматичного збору такої інформації:

– Google Analytics;

– IO Technologies;

– Segment.

Типові логи показані на рисунку 1.5.

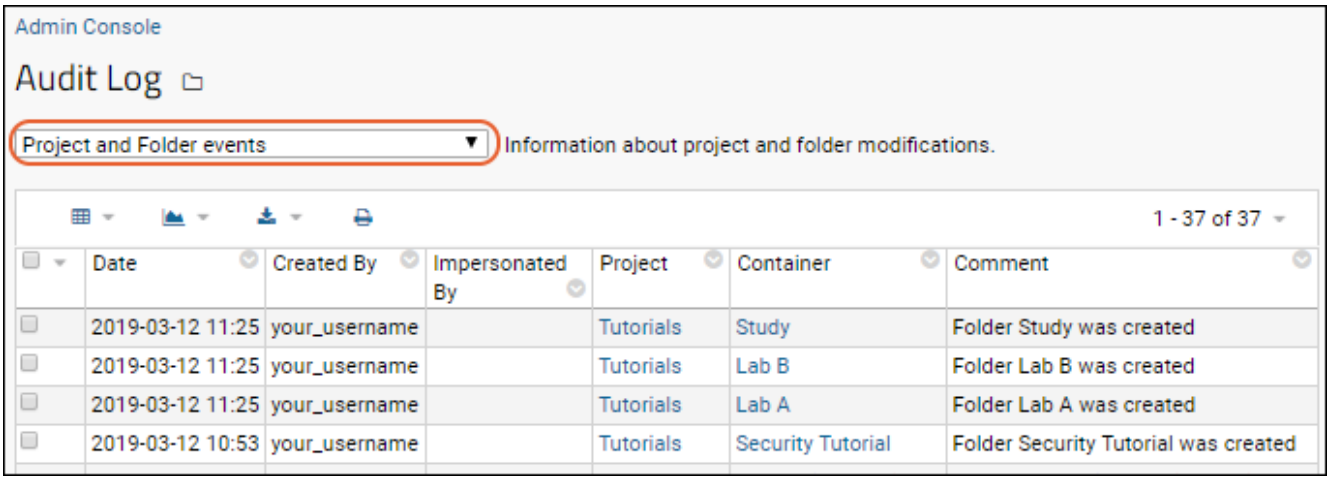

Рисунок 1.5 – Приклад веб-логів

Також одним із видів традиційно зберігаємої інформації є шлях кліків. Шлях кліку або потік кліків – це послідовність гіперпосилань, які один або більше відвідувачів веб-сайту відвідали на певному веб-сайті, подані в порядку перегляду. Шлях натискання відвідувача може починатися на веб-сайті або на окремому веб-сайті третьої сторони, часто на сторінці результатів пошуку, і він продовжується як множина послідовних веб-сторінок, які відвідує користувач. Шлях натискання приймає дані та може порівнювати їх із джерелами оголошень, ключовими словами та / або посилаються доменами, щоб фіксувати дані. Ці метрики також містять в собі інформацію про те як багато часу користувачі витрачають на вашому веб-сайті і як часто вони повертаються. Вони також покажуть які сторінки відвідуються найчастіше на веб-сайті. Найбільш очевидною причиною вивчення потоків кліків є отримання конкретної інформації про те, що люди роблять на вашому сайті. Вивчаючи окремі потоки кліків, ви отримаєте інформацію, необхідну для прийняття рішень, пов’язаних із вмістом, не роблячи хибних прогнозів. Знаючи, що роблять ваші відвідувачі на вашому веб-сайті, допоможе вам змінити зміст, макет і розміщення контенту на веб-сайті таким чином щоб якнайбільше спростити користувачеві доступ до потрібної інформації. Це не тільки покращить користувацький досвід, але й призведе до росту популярності. На рисунку 1.6 зображена типова візуалізація даних шляху кліків у виді графу.

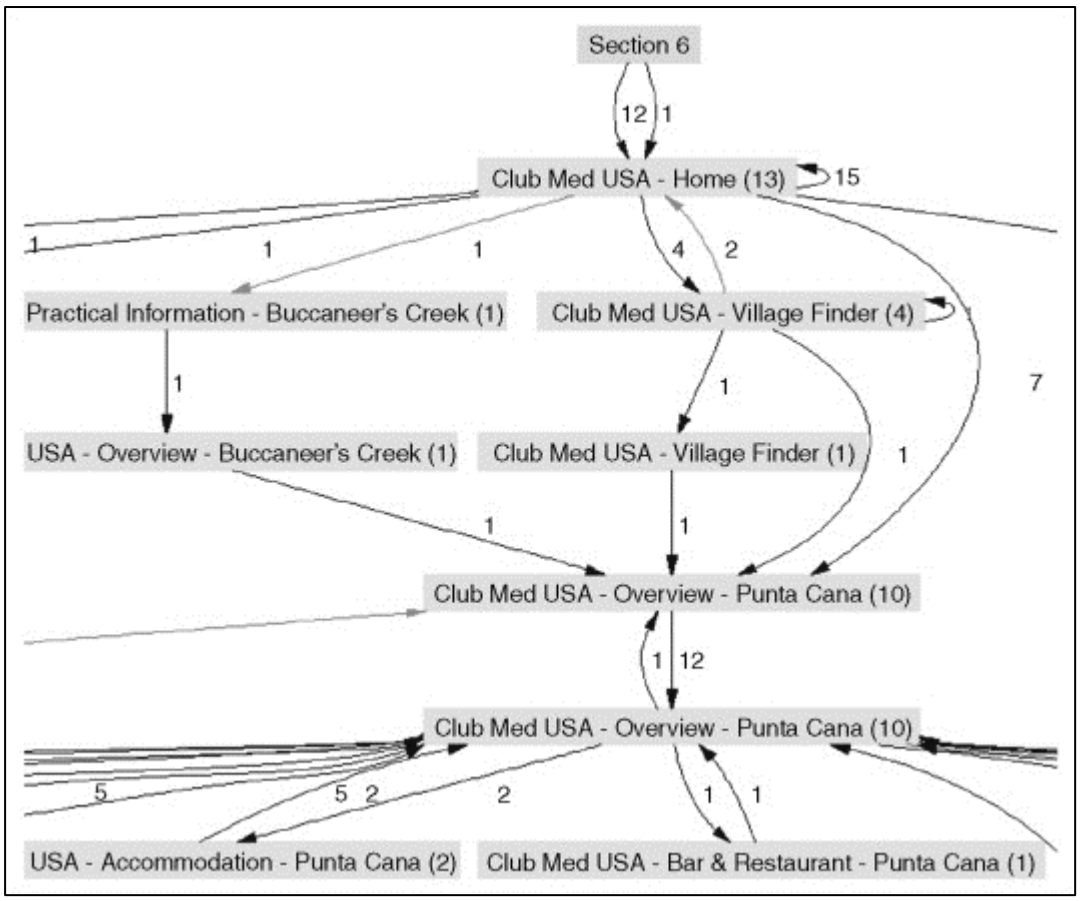

Рисунок 1.6 – Візуалізація шляху кліків у виді графу

Дані потоку кліків можуть використовуватися для кількісної оцінки поведінки пошуку за допомогою машинного навчання, в основному, зосередженого на записах про покупки.

В той час як покупка явно вказує на те що користувач віддає перевагу конкретній групі товарі, простий пошук також є важливим компонентом для вимірювання інтересу щодо конкретної категорії.

Ми будемо використовувати ймовірнісний генеративний процес для моделювання історії дослідження та покупок користувачів, в якому, для фіксації одночасного впливу як часу, так і місця, вводиться латентна контекстна змінна. Розпізнаючи шаблони пошуку споживачів, ми можемо оцінити їхні дії у конкретному контексті та рекомендувати правильні продукти. На рисунку 1.7 зображена типова послідовність дій для аналізу поведінки користувачів.

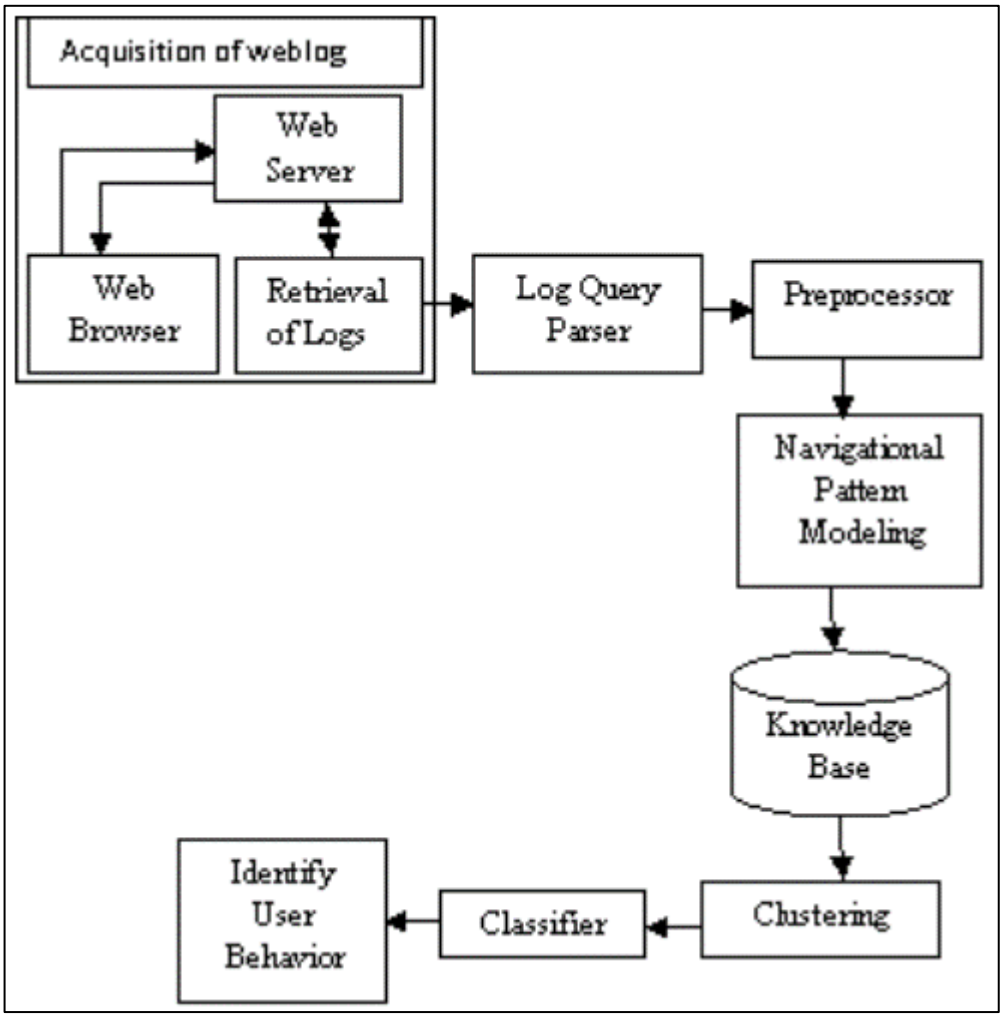

Рисунок 1.7 – Типова послідовність дій для аналізу поведінки користувачів

Сучасні пошукові системи використовують підходи машинного навчання для прогнозування активності користувачів у веб-контенті. Популярні моделі включають логістичну регресію (LR) [1] та посилені дерева рішень. Нейронні мережі мають перевагу перед LR, тому що вони здатні вловлювати нелінійні відносини між вхідними об'єктами і їх «більш глибока» архітектура володіє значно більшою силою моделювання. З іншого боку, дерева рішень – хоч і популярні в цій галузі, стикаються з додатковими проблемами, що стосуються великовимірних та розріджених даних.

Перевага ймовірнісних генеративних моделей, натхнених глибокими нейронними мережами, полягає в тому, що вони можуть імітувати процес купівельної поведінки споживача та вловлювати приховані змінні для пояснення даних.

1.3 Постановка задачі дослідження

Аналіз стану розвитку мережі інтернет, механізмів доступу до інформації про дії користувачів, проведений в попередньому розділі, показав, що задача передбачення дій користувача дуже актуальна та здійсненна в наші часи. Існує безліч шляхів для реалізації цієї ідеї. У дипломній роботі ставиться задача дослідження нейромережевого підходу для аналізу поведінки користувачів з метою передбачення наступних дій у WEB-системі. Для наочності дослідження звузимо предметну галузь до передбачення дій користувача на сайті інтернет-магазину. Таким чином ми повинні вирішити дві задачі: задачу передбачення того, чи закінчиться сеанс користувача покупкою (вдалий сеанс), та задачу передбачення того які сторінки на сайті інтернет-магазину користувач буде відвідувати у майбутньому, тобто які товари можуть його зацікавити. В якості нейронної мережі було прийнято рішення використати повно зв’язну, динамічну, багатошарову нейронну мережу.

Для навчання нейронної мережі будемо використовувати наступні методи:

- логістична регресія;
- випадковий ліс;
- глибинна мережа переконань (DBN);
- автокодувальники (stacked denoising autoencoders).

Буде виміряна їх ефективність за допомогою обчислення AUC показнику. На основі цього буде проведений порівняльний аналіз використаних методів та обраний найкращий. Для навчання нейронних мереж будуть використані навчальні набори даних шляху кліків на сайті інтернет-магазину. Тобто перед дипломною роботою стоять наступні задачі:

- аналіз та підготовка даних шляху кліків: на цьому етапі буде розглянута вибірка даних, для навчання нейронних мереж, буде проведена її

фільтрація та агрегація, введенні зміні для подальшого опрацювання, вибірка буде поділена на 6 частин для більш наочного опрацювання;

- реалізація обраних алгоритмів для прогнозування дій користувача та їх тестування на підготовлених наборах даних: для передбачення того чи закінчиться сеанс придбанням товару будуть використані алгоритми класифікації, а саме алгоритм логістичної регресії та алгоритм випадкового лісу, для передбачення того які товари зацікавлять користувача будуть побудовані дві нейронні мережі, а саме DBN та SdA;
- дослідження результатів роботи та підведення підсумків щодо їх оптимальності: на цьому етапі будуть вимірянні показники AUC (показник який дозволяє якість класифікації) та проведене їх порівняння, у задачі передбачення «вдалості сеансу» AUC логістичної регресії та випадкового лісу, у задачі передбачення товарів AUC нейронних мереж навчених за допомогою SdA та DBN;
- реалізація програмного забезпечення для ілюстрації результатів дослідження.

2 ПІДГОТОВКА ДАНИХ

Дані складаються з півроку записів взаємодії користувачів із веб-сайтом електронної комерції, взяті з Інтернету на сайті Kaggle. Цей веб-сайт надає безкоштовні датасети для науковох діяльності. Події мають ідентифікатор користувача, часову мітку та тип події. Існує 5 категорій подій: перегляд сторінки, перегляд кошика, покупка, перегляд реклами та показ реклами. На рисунку 2.1 зображений приклад даних у датасеті.

	A	B	C	D	E	F
1	user_id	timestamp	event_type	page_id	page_descr	additional_info_1
2	317	1589299963	BUY	14	Graphic Chip..	-
3	317	1588297413	PAGE_VIEW	268	Japanese Knife...	-
4	317	1588316413	CART_VIEW	-	-	-

Рисунок 2.1 – Приклад даних у датасеті

Існує близько 25 000 різних видів продукції. У разі купівлі чи перегляду у кошику ми маємо інформацію про ціну та додаткові деталі. Ми ігноруємо події показу реклами та її перегляду, оскільки вони не є актуальними для наших цілей. Дані дуже розріджені та великі. Є два очевидних способи зменшити розмірність даних: або згрупувати перегляди сторінок на користувача за період часу (сукупні перегляди сторінок на користувача за період), або згрупувати перегляди сторінок продукту (сукупні продукти, що переглядаються за часові рамки). У цій роботі ми дотримуємось першого підходу, оскільки більшість покупок (~ 87%) відбувається протягом 5 днів із першого відвідування. Навчальна вибірка складаються з набору сеансів $s \in S$ і кожен сеанс містить набір елементів $i \in I$, які відображалися користувачеві. Елементи, придбані в сесії s, позначаються B_s. Існує два типи сеансів S_b – сеанси, які закінчуються покупкою, та S_{nb} – сеанси, які не закінчуються покупкою.

Враховуючи набір сеансів S_t, завдання – знайти всі сеанси S_b, які мають принаймні одну подію покупки. Якщо сеанс буде містити подію покупки, ми хочемо передбачити придбані товари. Тому у нас є дві широкі цілі: 1) класифікація

та 2) передбачення порядку. У цій роботі ми зупинимося лише на першому завданні.

Дані є вкрай незбалансованими для двох розглянутих класів (купівля та не купівля), тому ми стикаємося з серйозною проблемою дисбалансу класів. Крім того, лише близько 1% продукції (близько 250) мають повну ідентифікацію категорії. Однак ця частка відповідає приблизно 85% переглядів сторінок і 92% покупок, тому у нас дуже перекошений розподіл. Спочатку ми розглядаємо лише взаємодію з цією підмножиною продуктів. Дані є приблизно 10 гігабайт і не можуть бути завантажені в пам'ять, тому ми спершу взяли вибірку перших 100 000 подій просто для того, щоб зробити знімок взаємодій. Ми знайшли:

– 78 360 подій перегляду сторінок (78.4% всіх подій) від 13 342 унікальних користувачів;
– 16 409 подій перегляду кошику (16.4% всіх подій) від 3091 унікальних користувачів;
– 2430 подій покупки (2.5%) від 2014 унікальних користувачів, це близько 1.2 покупок на користувача.

Якщо ми обмежимося 257 типами продуктів, ми знайдемо 39 561 переглянутих сторінок з 7469 унікальних користувачів, які складають близько половини вибірки. Введенні зміні на основі вхідних даних наведені у таблиці 2.1.

Таблиця 2.1 – Введенні зміні на основі вхідних даних

Позначення	Опис
D_s	Тривалість сеансу перед покупкою
C/B	Співвідношення кліків до покупок
S_B	Середня кількість сеансів перед покупкою
N_c	Число кліків за сеанс

У цій роботі ми не розглядали час, так як дані дуже роздроблені, і ми групуємо їх за часовими періодами. Так як наша вибірка досить багатомірна й розріджена, то вона вимагає подальшої обробки, фільтрації та групування. Таким чином ми розглянули тренувальні дані, охарактезували його, ввели деякі змінні для подальшої обробки підготували дані для подальшої обробки.

2.1 Попередня обробка даних

Кожна дія користувача протягом сеансу має унікальний ідентифікатор і мітку часу, тому ми маємо інформацію щодо кліків користувача по елементам веб-сайту в рамках сеансу. Тривалість кліка можна легко визначити, просто віднявши час цього кліка з часу наступного кліка. Тепер, для кожного окремого елемента в сеансі, якщо ми підсумовуємо тривалість кліків, в яких елемент з'являється, ми визначаємо тривалість елемента в цьому сеансі. Після сортування по часу коли клік був зроблений ми додаємо тривалість цього кліку до інформації про нього, далі ми виділяємо інші властивості наведені в таблиці 3.1, які є специфічними для елемента і додаємо їх до кожного кліку. Ми будуємо співвідношення кількості кліків і купувань користувачів шляхом усереднення співвідношення кліків і купівлі всіх елементів сеансу.

Ми також використовували опис придбаного товару у вигляді невеликого тексту. Для обробки текстових даних ми перетворили слова описів у вектори за допомогою word2vec [2] та використали середнє арифметичне для векторів. Робота word2vec здійснюється наступним чином: word2vec приймає великий текстовий масив в якості вхідних даних і зіставляє кожному слову вектор, видаючи координати слів на виході. Спочатку він генерує словник, а потім обчислює векторне подання слів, «навчаючись» на вхідних текстах. Векторне подання ґрунтується на контекстної близькості: слова, що зустрічаються в тексті поруч з однаковими словами (а отже, мають схожий зміст), будуть мати близькі (по

косинусній відстані) вектори. Отримані векторні уявлення слів можуть бути використані для обробки природної мови та машинного навчання [3].

Для створення набору даних спочатку обмежимось набором 257 категорій товарів. Дані агрегували на рівні тижня за категорією товару та півтижня (два періоди часу). У цій першій ітерації ми не будемо додавати події "перегляду кошика", оскільки більшість з них проводиться в один і той же сеанс/день продажних подій. Ми розглянемо це в наступній ітерації. Користувачів, на яких менше 10 кліків на веб-сайті, було видалено. Усі набори даних були збалансовані: однакова кількість купівель та відвідувань без покупок.

Завдяки великому розміру даних, ми, по суті, вивчаємо важливість розміру вибірки та ефективність алгоритмів, що стосуються розмірності даних. Оскільки ми хочемо прогнозувати покупки протягом 24 годин, ми виключали події в цей період. В таблиці 2.2 наведені шість підготовлених наборів даних з якими і будуть проведені дослідження.

Таблиця 2.2 – Набори даних використанні в дослідженні

Набір	Розмір	Опис
Набір 1	3 000	Покупки згруповані за тиждень
Набір 2	10 000	Набір 1 з більшою кількістю даних
Набір 3	30 000	Набір 1 з більшою кількістю даних
Набір 4	10 000	Набір 2, але згрупований за пів тижня
Набір 5	10 000	Набір 1 за 2000 категорій продуктів
Набір 6	30 000	Набір 3 за 2000 категорій продуктів

Розмір відноситься до кількості сеансів покупки. Усі набори даних були врівноважені підгрупуванням даних про сеанси, що не закінчилися покупкою. Дані були надані у форматі JSON [4], і ми сортуємо всі сеанси та покупки за sessionId. Кількість сеансів у наших власних тестових даних склала 1 506 453. Дані про кліки сеансу покупки містять набір придбаних елементів (Bs). Для кожного елемента і ∈ Bs ми отримуємо характеристики двох типів, засновані на сеансі, і на основі переглянутих товарів.

2.2 Розклад невід'ємних матриць

Для того щоб перевірити вплив виключення деяких категорій товарів, ми розглянемо набір 5 з першими 2000 найбільш відвідуваних категорій товарів. Оскільки це простір пошуку дуже великої розмірності, для її зменшення ми використовували розклад невід'ємних матриць (NMF). Розклад невід'ємних матриць – це група алгоритмів багатовимірного аналізу та лінійної алгебри, де матриця V розкладається в, зазвичай, дві матриці W, H, враховуючи, що жодна з трьох матриць немає від'ємних елементів. Завдяки невід'ємності результуючі матриці легко перевіряються. Так як проблема немає точних розв'язків, в загальному випадку, зазвичай, знаходять числове наближення. NMF – це клас непідконтрольних алгоритмів навчання, таких як метод основних компонент (PCA) або квантування векторного навчання (LVQ) [5], який факторизує матрицю даних, що зазнає обмежень. Хоча PCA є широко використовуваним алгоритмом, він має деякі недоліки, такі як його лінійність. Крім того, він застосовує слабке обмеження ортогональності [6].

Кожен вектор даних V може бути наближений лінійною комбінацією стовпців W, зваженою за матрицею шаблонів H. Тому W можна вважати таким, що містить основу для лінійного наближення даних у V. Оскільки відносно мало базових векторів використовується для представлення багатьох векторів даних,

хорошого наближення можна досягти лише в тому випадку, якщо базові вектори виявлять структуру, яка є латентною в даних. NMF був успішно застосований до проблем із великими розмірами із розрідженими даними, як-от розпізнавання зображень та аналіз тексту. У нашому випадку ми використовували NMF для стиснення даних у підмножину функцій. Основна проблема NMF – відсутність оптимального методу для обчислення факторних матриць та зупиняючих критеріїв для визначення ідеальної кількості функцій, що підлягають вибору.

2.3 Аналіз та класифікація даних

Наше завдання розділено на дві підзадачі:

– прогнозування результату сеансу;
– передбачення набору елементів, які слід придбати на цьому сеансі.

Задіяно два набори класифікаторів: двійкове та ранжувальне прогнозування. Побудова одного класифікатора недоцільна через велику розмірність проблеми. На основі наборів даних ми перевіряємо продуктивність двох класифікаторів: логістична регресія та випадковий ліс. Перший є стандартом у промисловості і слугує базовою лінією, другий – більш надійним і дає загалом кращі результати. Недоліком їх прогнозів є важкість для розуміння (чорний ящик) [7]. Ми використовували алгоритми без оптимізації параметрів (кількість дерев, кількість змінних, які слід враховувати в кожному розрізі, рівень розбиття тощо). В якості KPI для вимірювання продуктивності ми використовуємо стандартну криву Area Under Roc (AUC). AUC = 0,5, що означає випадковий (марний) клас і 1 – досконалий. Для всіх ітерацій ми використовували 10-кратну перехресну перевірку.

Дерева рішень мають ряд переваг порівняно з іншими методами класифікації, такими як векторні алгоритми, нейронні мережі, лінійна регресія та логістична регресія [8]. Серед переваг дерев рішень:

- надзвичайно легко візуалізувати та інтерпретувати: дерево рішення може бути представлене графічно, що дозволяє користувачеві реально бачити структуру класифікатора;
- моделі «білого ящику»: спостерігаючи дерево рішень, можна чітко зрозуміти всі проміжні етапи процесу класифікації, наприклад, які змінні використовуються, яким порядком тощо. Це не стосується інших методів, таких як нейронні мережі, чиї параметри не можуть бути безпосередньо інтерпретовані;
- надзвичайно швидко: дерева рішень навчаються за порівняно короткий час і особливо швидко класифікують нові дані.

Однак дерева рішень мають ряд недоліків. Процес побудови оптимального дерева рішень може бути доведений як важкий NP, а тому створити глобально оптимальне дерево неможливо. Дерева рішень часто перевищуватимуть дані, якщо не будуть використані деякі методи регуляризації, такі як обрізка або накладення мінімальної кількості навчальних зразків на лист. Крім того, через характеристики функції витрат, що використовуються для визначення найкращого розбиття на вузлі, дерева будуть віддавати перевагу категоричним змінним з більшою кількістю категорій перед іншими змінними. Це може призвести до того, що класифікатор неправильно вважає ці змінні важливішими, ніж ті, що мають менше категорій.

Алгоритм Random Forest (RF) створює ансамбль дерев рішень за допомогою рандомізації [9]. Коли вхідні дані повині бути класифіковані, кожне дерево класифікує вхідні дані окремо. Потім визначається остаточна класифікація, обираючи більшість голосів за всі дерева. Ймовірність того, що певні вхідні дані належать до кожного класу, обчислюється шляхом усереднення ймовірностей на листках кожного дерева.

Кожне дерево будується незалежним, випадковим чином. Набір, який використовується для тренування даного дерева, є підмножиною вихідних даних про навчання; кожен приклад тренінгу вибирається навмання (із заміною) з вихідного набору даних. На кожний вузол дерева замість тестування найкращого розбиття серед усіх атрибутів використовується лише випадковим чином обрана

підмножина атрибутів (яка, як правило, значно менша, ніж повний набір атрибутів) для визначення найкращого розбиття. Кожне дерево вирощується повною мірою, це означає, що обрізка не відбувається.

Фінальний класифікатор є ефективним і здатний працювати з великими наборами даних (тобто з даними, що містять велику кількість змінних), відсутніми даними та викидами. У цій проблемі для кожного клієнта доступна велика кількість інформації. Щоб уникнути видалення можливо значущих змінних з метою зменшення даних до керованого розміру: те, що було б обов'язковим, якби використовували нейронні мережі. В нашому випадку ми обрали випадковий ліс.

Випадковий ліс зберігає сили дерев рішень, протидіючи деяким їх недолікам. Навіть якщо дерева в лісі будуються без обрізки, той факт, що результат виконання класифікатора залежить від усього набору дерев, а не від одного дерева, значно знижує ризик перенасичення. Випадковість, яка вводиться при створенні кожного дерева, також не дозволяє класифікатору запам’ятати всі приклади навчального набору. Прийоми регуляризації, згадані в попередньому пункті, також можна застосовувати до дерев у лісі, ще більше знижуючи ризик перенасичення. Однак випадкові ліси мають однаковий ухил до змінних з багатьма категоріями, як дерева рішень.

3 МЕТОДИ ГЛИБИННОГО НАВЧАННЯ

Глибинне навчання відноситься до широкого класу машинних методів навчання та архітектури, що є ознакою використання багатьох шарів нелінійної обробки, які мають ієрархічний характер. Концепція глибокого навчання виникла з досліджень штучних нейронних мереж, що рухаються вперед або MLP, з багатьма прихованими шарами, що відносяться до глибоких нейронних мереж (DNN). Ці мережі, як правило, навчаються алгоритмом градієнтного спуску, а саме алгоритмом зворотного розповсюдження (BP). Однак для глибоких мереж лише у BP є кілька проблем: локальні оптимістичні пастки в невипуклій цільовій функції та зникнення градієнтів (навчальний сигнал зникає експоненціально як інформація, що передається по шарах). У цьому розділі ми запровадимо два підходи до глибокого навчання для вирішення високої розмірності пошукового простору та порівняємо їх продуктивність з логістичною регресією та алгоритму випадкових лісів.

У 2006 році Гінтон запропонував непідконтрольний алгоритм, названий глибинною мережею переконань (DBN). DBN можна розглядати як композицію обмежених машин Больцмана (RBM) [11]. Основним компонентом DBN є жадібний, пошаровий алгоритм навчання, який оптимізує ваги DBN. DBN може бути розглянутий як композиція простих мереж, в якій прихований шар минулої підмережі виступає в якості видимого шару для наступної. DBN належить до класу моделей на основі енергії. У цьому випадку алгоритм працює наступним чином. Для заданої RBM ми співвідносимо одиниці з енергетичною функцією,

$$Energy(v,h) = -b'h - c'h - h'Wv \tag{3.1}$$

де b, c – зміщення,

w – вагові з’єднувальні вузли.

Спільна ймовірність видимого (v) та прихованого (h) єдностей, (v, h) дорівнює:

$$P(v,h) = \frac{1}{Z} e^{-Energy(v,h)} \tag{3.2}$$

де Z – член нормалізації.

Вільну форму енергії ми отримуємо, маргіналізуючи h:

$$P(v) = \frac{\sum_h e^{-Energy(v,h)}}{Z} = \frac{e^{-FreeEnergy(v)}}{Z} \tag{3.3}$$

Користуючись перевагами вільної енергії, це полегшує обчислення градієнтів лише з видимими одиницями. Ми переписуємо енергетичну функцію у форму,

$$Energy(v,h) = -\beta(v) - \sum_i \gamma_i(v,h_i) \tag{3.4}$$

Тоді ми розкладемо P (v):

$$P(v) = \frac{\sum_h e^{-Energy(v,h)}}{Z} = \frac{e^{-FreeEnergy(v)}}{Z} = \frac{1}{Z}\sum_{h_1}\sum_{h_2}\dots\sum_{h_k} e^{\beta(v)-\sum_i \gamma_i(v,h_i)} =$$
$$\frac{1}{Z}\sum_{h_1}\sum_{h_2}\dots\sum_{h_k} e^{\beta(v)}\prod_i e^{-\gamma_i(v,h_i)} = \frac{e^{\beta(v)}}{Z}\sum_{h_1} e^{-\gamma_1(v,h_1)} \dots \frac{e^{\beta(v)}}{Z}\sum_{h_2} e^{-\gamma_2(v,h_2)} =$$
$$\frac{e^{\beta(v)}}{Z}\prod_i \sum_{h_i} e^{-\gamma_i(v,h_i)} \tag{3.5}$$

DBN використовувались для найрізноманітніших проблем, починаючи від розпізнавання зображень, алгоритмів рекомендацій та моделювання тем. Окрім постачання прийнятних точок ініціалізації для багатошарової мережі (див. рис. 3.1), DBN має й інші привабливі властивості:

– алгоритм навчання дозволяє ефективно використовувати незазначені дані;

– він може бути інтерпретований як імовірнісна генеративна модель;

– проблема переоцінки, яка часто спостерігається в моделях з мільйонами параметрів, таких як DBN, може бути ефективно усунена на етапі генеральної попередньої підготовки.

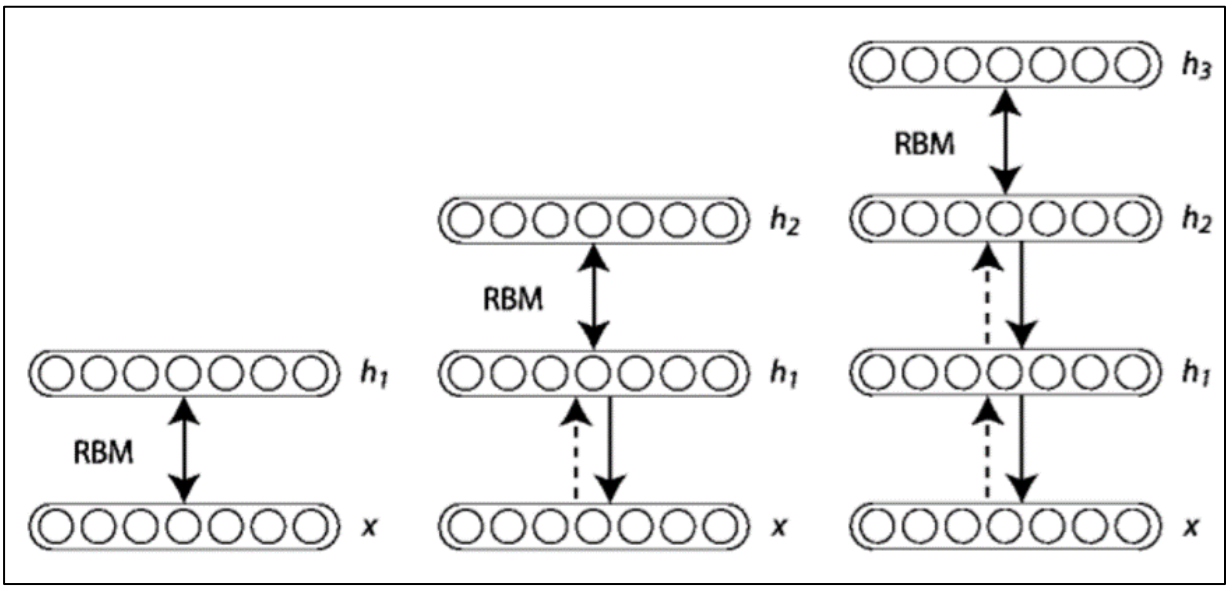

Рисунок 3.1 – Структура глибинної мережі переконань

Мінусом DBN є те, що вони важко навчаються і вони дуже чутливі до таких параметрів навчання, як ініціалізація ваг.

3.1 Автокодувальники

Autoencoder (див. рис. 3.2) – це непідтримувана штучна нейронна мережа, яка вчиться ефективно стискати та кодувати дані, потім навчається реконструювати дані назад із зменшеного кодованого подання до представлення, яке є максимально наближеним до вихідного вводу. Autoencoder, за задумом, зменшує розміри даних, навчившись ігнорувати шум у даних [10].

Автокодувальник – це нейронна мережа з одним прихованим шаром, де вихідний і вхідний шар мають однаковий розмір. Припустимо, що вхід $x \in Rm$ і припустимо, що в прихованому шарі є n вузлів. Тоді маємо вагову матрицю $W \in Rm \times n$ та вектори зміщення b і b0 в Rm і Rn відповідно.

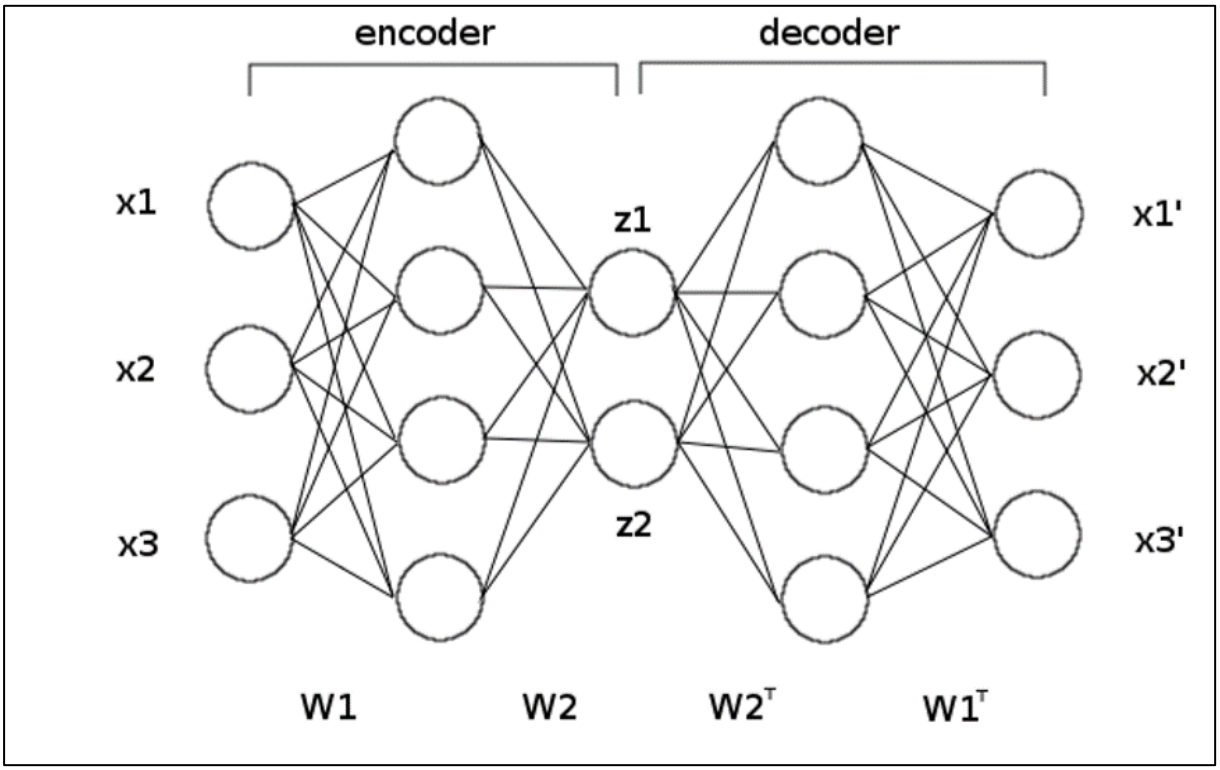

Рисунок 3.2 – Структура автокодувальника

Нехай s (x) = 1 / (1 + e $^{-x}$) – сигмоїдна (логістична) функція передачі. Тоді ми маємо нейронну мережу, як показано на рисунку 4.2. Використовуючи автокодер для кодування даних, обчислюємо вектор y = s (W_x + b); відповідно, коли ми використовуємо автокодер для декодування та реконструкції назад вхідних даних, обчислюємо z = s (W^T_x + b`). Вагова матриця стадії декодування – це транспонування вагової матриці стадії кодування з метою зменшення кількості параметрів для вивчення. Ми хочемо оптимізувати W, b і b` так, щоб реконструкція була максимально схожою на початкові вхідні дані стосовно деякої функції втрат. Використана функція втрат – це найменша втрата квадратів:

$$E(t,z) = \frac{1}{2}(t-z)^2 \tag{3.6}$$

де t – оригінальні вхідні дані.

Після тренування автокодера, його етап декодування відміняється, а етап кодування використовується для перетворення прикладів введення тренінгу як етап

попередньої обробки. Після того, як пройшов тренінг шару автокодування, другий автокодер може тренуватися, використовуючи вихід першого шару автокодера. Цю процедуру можна повторити нескінченно та створити складені шари автоенкодера довільної глибини. Показано, що кожен наступний натренований шар вчиться кращому розпізнаванню ознак ніж попередній. Використання глибоких нейронних мереж, таких як складені автокодери для розпізнавання ознак також називається глибокими автокодерами – окрема область машинного навчання. Для звичайних автокодерів ми зазвичай хочемо, щоб n < m, вивчене існувало в просторі менших розмірів, ніж вхідне. Це робиться для того, щоб автокодер не засвоїв тривіальну трансформацію ідентичності. Варіантом є позначаючі автокодери, які використовують різний критерій реконструкції для вивчення ознак. Це досягається пошкодженням вхідних даних та навчанням автокодера для реконструкції вихідних непошкоджених даних. Навчившись позначати, автоекодер змушений зрозуміти справжню структуру вхідних даних і навчитися добре їх уявляти. Під час навчання з критерієм, що позначає, глибокий автокодер є також генеративною моделлю. Хоча функція втрат E (t, z) для нейронних мереж взагалі не випукла, стохастичний градієнтний спуск (SGD) достатній для більшості проблем, і ми використовуємо його в цій роботі.

3.2 Регулярізація

У статистиці та машинному навчанні одним із найпоширеніших завдань є пристосовування «моделі» до набору тренувальних даних таким чином, щоби уможливити здійснення надійних передбачень на загальних даних, на яких не здійснювалося тренування. При перенавчанні (англ. overfitting) статистична модель описує випадкову похибку або шум, замість взаємозв'язку, що лежить в основі даних [13]. Перенавчання виникає тоді, коли модель є занадто складною, такою, що має занадто багато параметрів відносно числа спостережень. Перенавчена модель

має погану передбачальну продуктивність, оскільки вона занадто сильно реагує на другорядні відхилення в тренувальних даних.

Уникнення перенавчання особливо важливе для глибоких нейронних мереж, які зазвичай мають мільйони параметрів. DBN може генерувати великі та виразні моделі, здатні представляти складні залежності між вхідними та вихідними даними. Генеральна неконтролюєма попередня підготовка є потужним регуляризатором, залежним від даних, в той час як метод виключення є найбільш поширеним.

Регуляризація L2 зміщує ваги до нуля, що може бути небажаним. Виключення карає великі ваги, що призводить до непевних прогнозів. Іншим способом перегляду виключення є модель усереднення за експоненціально численними різними нейронними мережами, що виробляють обрізку випадкових підмножин прихованих одиниці і вхідних даних. У цій роботі ми використовували регуляризацію виключення.

Зазвичай доля відсіву визначається емпіричним шляхом, в якості початкового значення було обране 0.3. В майбутньому, в залежності від показників ефективності це значення може бути змінено.

4 РЕАЛІЗАЦІЯ ТА РЕЗУЛЬТАТИ ДОСЛІДЖЕНЬ

ROC-крива – графік, що дозволяє оцінити якість бінарної класифікації, відображає співвідношення між часткою об'єктів від загальної кількості носіїв ознаки, вірно класифікованих до загальної кількості об'єктів, що не несуть ознаки, помилково класифікованих, як такі, що мають ознаку. Також відома як крива похибок. Аналіз класифікацій із застосуванням ROC-кривих називається ROC-аналізом.

Кількісну інтерпретацію ROC дає показник AUC – площа, обмежена ROC-кривою і віссю частки помилкових позитивних класифікацій. Чим вище показник AUC, тим якісніше діє класифікатор, при цьому значення 0,5 демонструє непридатність обраного методу класифікації (відповідає звичайному вгадуванню).

Спочатку ми запускаємо алгоритми, використовуючи лише агреговані змінні для прогнозування подій покупки. Оскільки ці дані маломірні, ми розглядаємо лише LR та RF алгоритми. Однак зауважте, що більшість подій з купівлі відбувається протягом 24 годин. Ми знайшли і AUC для LR 0,58 і для RF 0,61.

Тоді ми використовували всі дані, поєднуючи таблицю 2.1 з таблицею 2.2 та опис продуктів, попередньо оброблений за допомогою word2vec алгоритму (виключаючи набір стоп-слів). Результати представленні в таблиці 4.1.

Таблиця 4.1 – Показник AUC при аналізу з використанням LR та RF методів

Набір	Показник LR	Показник RF
Набір 1	0.67	0.71
Набір 2	0.69	0.76
Набір 3	0.70	0.80

Продовження таблиці 4.1

Набір	Показник LR	Показник RF
Набір 4	0.68	0.82
Набір 5-100	0.62	0.67
Набір 5-200	0.64	0.69
Набір 5-300	0.64	0.72

Ми робимо висновок, що розмір вибірки є важливим фактором продуктивності класифікатора, хоча логістична регресія не має таких самих переваг, як алгоритм Random Forest (RF).

З набору даних 4 ми також робимо висновок, що час подій є важливим фактором, який слід враховувати: хоча ми збільшуємо розмірність простору пошуку, ми все ще отримуємо чистий прибуток, навіть використовуючи менше прикладів навчання.

З набору даних 5 ми дійшли висновку, що алгоритм NFM робить деяке стиснення даних, але не дуже ефективним способом (лише дані з 300 признаками покращили точність порівняно з початковою підмножиною продуктів). У наступному розділі ми пропонуємо використовувати автокодувальники для зменшення розмірності даних для всіх 25 000 категорій.

Досить дивно, ми виявили, що використання детальної інформації про те, які продукти відвідував користувач, не приносить великої точності логістичній регресії (в деяких випадках вона навіть зменшується – ймовірно, за рахунок збільшення розмірності), тоді як алгоритм Random Forest може захоплювати більш високі точності.

4.1 Результати з використанням глибинного навчання

Однією з головних переваг DBN або SdA є те, що ми можемо використовувати всі наявні дані (навіть якщо вони не є маркованими) для попередньої підготовки моделі без нагляду або використати шлях генерації. У DBN це призначене для збору кореляції високого порядку спостережуваних або видимих даних з ціллю аналізу паттерну або синтезу, коли немає інформації про мітки цільового класу. Тоді ми можемо спільно охарактеризувати статистичні розподіли видимих даних та пов’язані з ними класи, коли вони є. Нарешті, використання правила Байєса може перетворити цей тип генеративних мереж на розрізнювальний автомат. Ми використовували мільйон даних сеансів (перегляди сторінок продуктів, агрегованих на користувача та за тиждень) разом із складеними параметрами, описаними в таблиці 2.1.

Для кожного аналізу ми використали випадково 25% даних для використання в якості тестового набору, залишивши решту 75% як навчальний набір. Ми розділили навчальний набір на чотири склади і тренували кожну модель чотири рази з різною складкою, в якості даних валідації. Ми оцінюємо AUC для тестового набору чотирьох моделей, коли отримуємо результати тестових наборів. Ми використали показники моделей на даних валідації, щоб вибрати найкращу конкретну модель у кожному сімействі моделей.

Нейронні мережі мають багато метапараметрів: архітектурні, такі як розміри шарів та приховані функції передачі одиниць; оптимізаційні, наприклад, темпи навчання та значення імпульсу; і регуляризаційні, така як ймовірність виключення для кожного шару. Глибокі нейронні мережі можуть мати дуже велику кількість параметрів, в нашому випадку – від одного до 4 мільйонів ваг. Всі метапараметри нейронної сітки були встановлені з використанням байєсівської оптимізації для максимізації AUC валідації. Байєсівська оптимізація ідеально підходить для глобальної оптимізації чорного ящику, галасливих функцій, будучи жадібною в кількості оцінок функцій.

Метапараметри, що розглядаються для тренування, були:

– доля відсіву ∈ [0: 0,3];
– кількість навчальних ітерацій ∈ [10: 100] для мереж з одним прихованим шаром і ∈ [10: 150] для мереж з двома або більше прихованими шарів;
– початкова швидкість навчання, застосована до середнього градієнта ∈ [0,001: 0:25];
– вагова вартість L2 ∈ [0: 0,01];
– імпульс ∈ [0: 0,95];
– рівень шуму, застосований до вхідного шару (тільки для SdA) ∈ [0: 0,2].

На рисунку 4.1 наведена схема нейронної мережі.

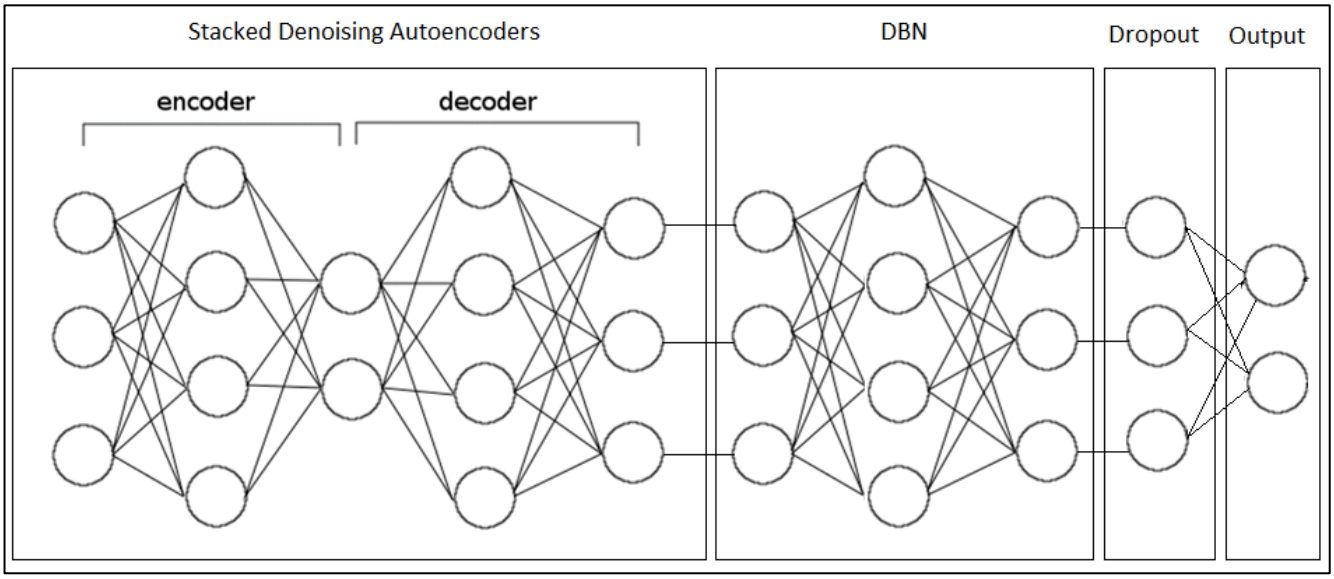

Рисунок 4.1 – Схема нейронної мережі

Для запуску мережевих моделей ми використовували реалізацію Keras на основі бібліотек theano (див. http://keras.io/). Theano – бібліотека Python для обчислення та маніпуляції математичними виразами. Ми оптимізували метапараметри мереж на наборі даних 3 (30 000 операцій з купівлі) та використовували ті ж параметри для інших наборів даних. Розподіл сеансів що закінчилися покупками серед наборів даних ви можете бачити на рисунку 4.2.

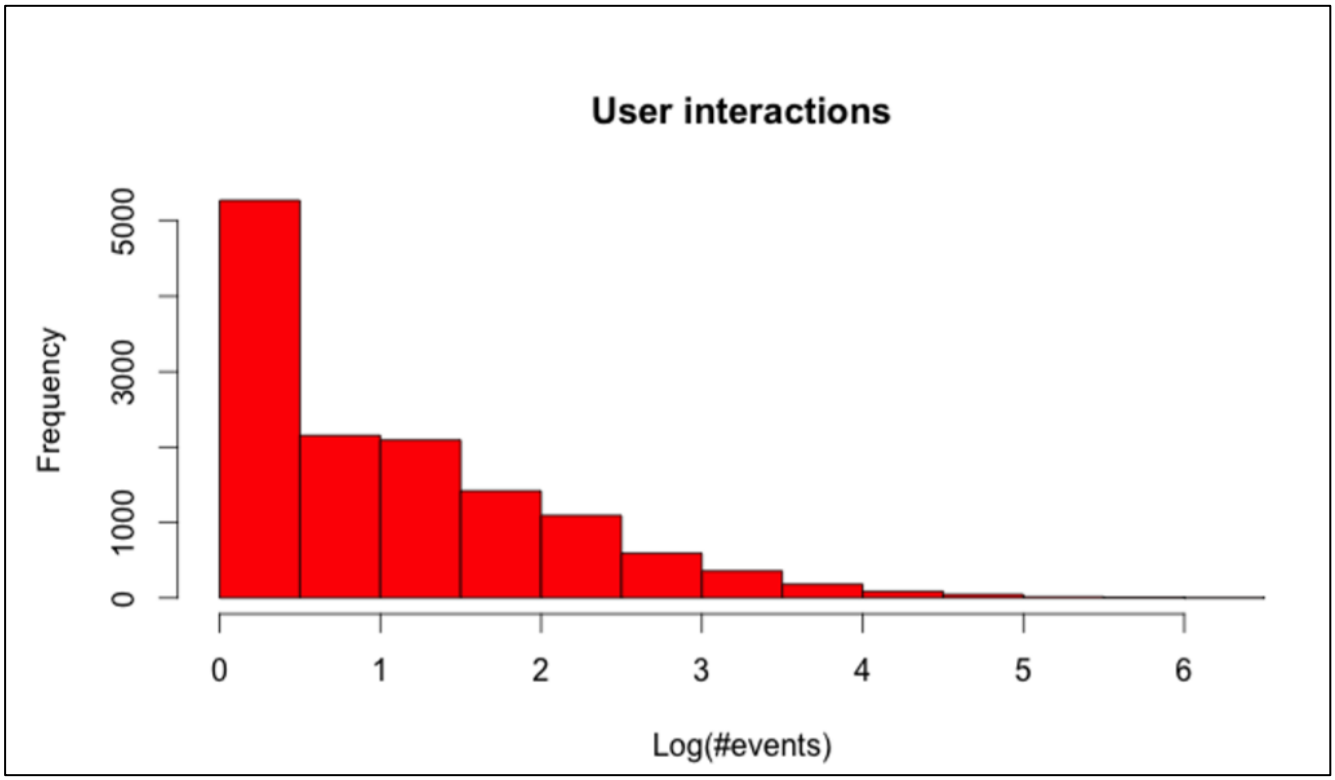

Рисунок 4.2 – Розподіл сеансів що закінчилися покупками серед наборів даних

Результати представлені в таблиці 4.2.

Таблиця 4.2 – Показник AUC для класифікації використанням DBN та SdA методів

Набір	DBN	SdA
Набір 3	0.82	0.83
Набір 6	0.84	0.86

Ми можемо бачити, що мережі, попередньо підготовлені за допомогою Stacked Denoising Autoencoders, досягають найвищої точності, що може бути пов’язано з тим, що у нас дуже розріджений набір даних. Так як цей метод показує найбільшу ефективність при роботі з розрідженими даними. Покращення, порівняно з іншими традиційними методами, мало відомі.

4.2 Програмна реалізація

Для відображення результатів дослідження було прийнято розробити прототип програмної системи. Програмна система виконує функції рекомендації товару в інтернет-магазині базуючись на минулих інтересах користувача отриманих через дані шляху кліків та проаналізованих шляхом описаним в минулих розділах.

Програмна система складається з двох модулів: модулю для аналізу дій користувача та передбачення товарів, які можуть його зацікавити та сервіс-модулю який служить для збереження інформації, виводі її на екран, тощо. Для розробки першого модулю була обрана мова програмування Python. Python – високорівнева мова програмування з мінімалістичним синтаксисом. Її зручність зумовлена тим, що вона підтримує не тільки об’єктно-орієнтований підхід але також функціональний й імперативний. Також Python підтримує динамічну типізацію та багатопоточні розрахунки. Але перш за все ця мова була обрана тому що має безліч вже реалізованих фреймворків для роботи з техніками машинного навчання. Серед них найвідоміші:

- Tensor Flow;
- Keras Python;
- Theano Python;
- PyTorch.

Для реалізації програмної системи був обраний Keras Python.

Для розробки сервіс-модулю була обрана мова Java та середа розробки – Intelij IDEA 2020. Мова Java була обрана через зручність її фреймворків та наявність великого досвіду при роботі з нею. При розробці використовувалися наступні фреймворки для організації залежностей та коду:

- Spring Core;
- Spring Boot;

– Spring Data.

Для збереження проміжного стану виконання та даних була обрана база даних MongoDB. MongoDB – це документо-орієнтована СКБД. Такий підхід був обраний саме через його великі переваги у швидкості над реляційними базами даних. Також був використаний модуль Spring Data для спрощення конфігурації збереження даних та спрощення доступу до них. Spring Framework дає можливість зручного контролю над залежностями завдяки модулю Spring IoC. Inversion of Control – це паттерн проектування програмного забезпечення при якому частини програмного забезпечення не керують залежностями, а отримують їх з зовні, з IoC контейнеру, який саме й керує життєвим циклом залежностей. Структура Spring Framework представлена на рисунку 4.3.

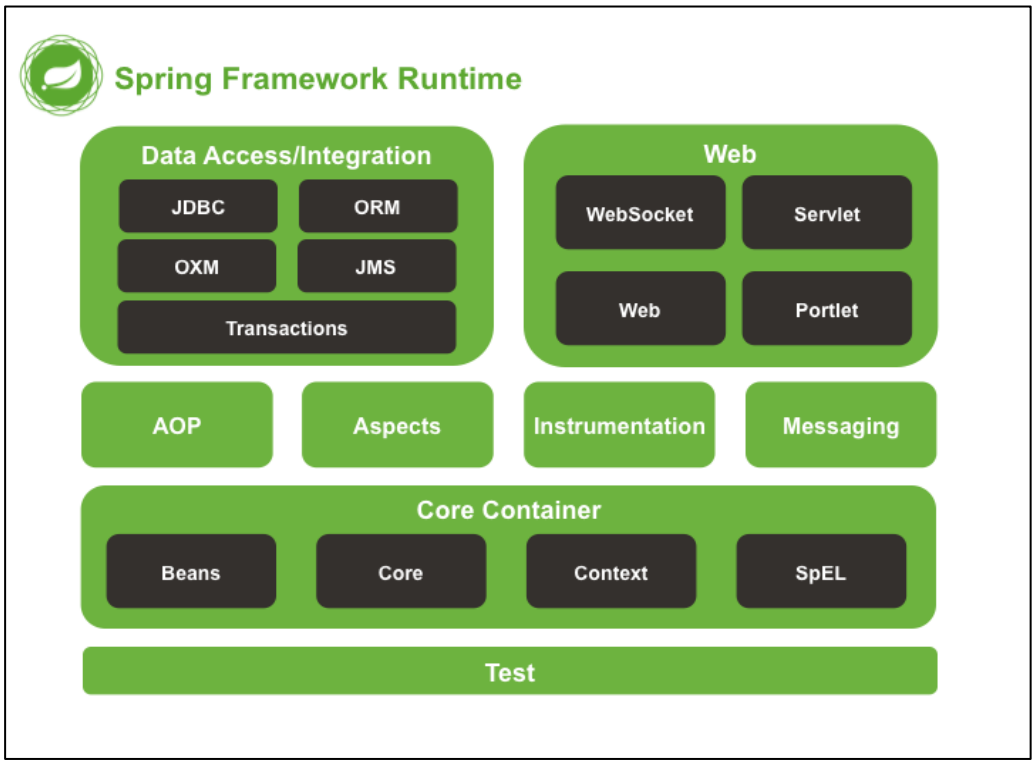

Рисунок 4.3 – Структура Spring Framework

Цей механізм називається Dependency Injection. Це дозволяє позбавитися від залишкової зв’язаності коду. Конфігурацію DI в реалізованій програмній системі ви можете бачити на рисунку 4.4.

```
@Bean
public ObjectMapper objectMapper() {
    return new ObjectMapper()
            .setSerializationInclusion(JsonInclude.Include.NON_EMPTY)
            .disable(DeserializationFeature.FAIL_ON_UNKNOWN_PROPERTIES);
}

@Bean
public ProductService productService(ProductRepository productRepository,
                                     ProductAnalyzer productAnalyzer,
                                     ProductFormatter productFormatter) {
    return new ProductService(productRepository, productAnalyzer, productFormatter);
}
```

Рисунок 4.4 – Приклад конфігурації залежностей в реалізованій програмній системі

Перед тим щоб використовувати механізм передбачення треба спершу натренувати нейронну мережу. На рисунку 4.5 зображений програмний код ініціалізації SDA функції для навчання нейронної мережі.

```
train_fn = theano.function(
    inputs=[index],
    outputs=self.finetune_cost,
    updates=updates,
    givens={
        self.x: train_set_x[
            index * batch_size: (index + 1) * batch_size
        ],
        self.y: train_set_y[
            index * batch_size: (index + 1) * batch_size
        ]
    },
    name='train'
)
```

Рисунок 4.5 – Програмний код ініціалізації SDA функції

Візуалізацію процесу тренування та проміжні значення точності та втрат ви можете бачити на рисунку 4.6.

```
Train on 60000 samples
Epoch 1/10
60000/60000 [==============================] - 4s 72us/sample - loss: 0.4963 - accuracy: 0.8261
Epoch 2/10
60000/60000 [==============================] - 4s 60us/sample - loss: 0.3736 - accuracy: 0.8647
Epoch 3/10
60000/60000 [==============================] - 4s 60us/sample - loss: 0.3347 - accuracy: 0.8789
Epoch 4/10
60000/60000 [==============================] - 4s 60us/sample - loss: 0.3144 - accuracy: 0.8843
Epoch 5/10
60000/60000 [==============================] - 4s 62us/sample - loss: 0.2962 - accuracy: 0.8903
Epoch 6/10
60000/60000 [==============================] - 4s 61us/sample - loss: 0.2829 - accuracy: 0.8957
```

Рисунок 4.6 – Візуалізація процесу тренування нейронної мережі

Після тренування нейронної мережі ми можемо завантажити в програму шлях кліків який нас цікавить та отримати ідентифікаційні номери рекомендованих товарів, які згенеровані нейронною мережею. Також разом з цим відображається час виконання та задані параметри. Скріншот вікна результату зображений на рисунку 4.7.

```
Recommended Product IDs: 317, 214, 567, 18
==========================================
Execution Time: 2s 61ms
Count of desirable products: 4
```

Рисунок 4.7 – Вікно результату виконання програми

В результаті розробки програмного забезпечення ми маємо додаток, головною функцією якого є передбачення товарів які можуть зацікавити користувача в інтернет-магазині на основі посилань до товарів, які він переглядав перед цим. В наступних ітераціях розробки буде доцільно додати графічний інтерфейс та впровадити даний механізм в реальний інтернет-магазин.

ВИСНОВКИ

У цій роботі ми застосовуємо різні алгоритми машинного навчання, а саме глибинні мережі переконань (DBN) та автоенкодери (SdA) для навчання повно зв'язної, динамічної, багатошарової нейронної мережі. Дослідження показало що використання stacked denoising autoencoders для тренування перед використання deep belief network покращило показник AUC в порівнянні з нейронною мережею коли було використано тільки алгоритм DBN для навчання.

Для більш наочної ілюстрації результату досліджень було розроблене програмне забезпечення, яке виконує функції рекомендації товарів для користувача.

Подальші дослідження можуть включати тестування даних у режимі реального часу та спостереження ефективності у режимі реального часу. Однак потрібно зробити більше роботи над покращенням ефективності в реальному часі. З точки зору масштабованості, дані є надзвичайно розрідженими, і клас алгоритмів, який ми використовували, показує поганий паралелізм з декількома ядрами. Оскільки результати наочно показують розрив у покращенні продуктивності при великому розмірі даних, було б цікаво побачити ефект використання на значно більших навчальних даних. Більше того, оскільки багато функцій на основі ідентифікації є у формах слів, може бути корисним ініціалізувати нейронну мережу як RBM, що навчається з непідконтрольним контрастним розбіжжям на великому обсязі незазначених прикладів. А потім визначити це як дискримінаційну модель із зворотнім розповсюдженням. Також може виявитись корисним паралельно тренувати декілька мереж і подавати всі їх виходи окремо в MatrixNet, як функції векторів, а не лише на один середній показник.

Завдяки механізмам передбачення дій користувача у майбутньому людство буде витрачати значно менше часу на такі рутинні справи в мережі Інтернет як купівля товарів чи пошук необхідної інформації.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Hanwu Luo, Xiubao Pan, Qingshun Wang. “Logistic Regression and Random Forest for Effective Imbalanced Classification”. 2019. URL: https://ieeexplore.ieee.org/document/8754250 (дата звернення: 24.03.2020)

2. Al-Amin, Saiful Islam, Shapan Das Uzzal. “Sentiment analysis of Bengali comments with Word2Vec and sentiment information of words”. 2017. URL: https://ieeexplore.ieee.org/document/7912903 (дата звернення: 24.03.2020)

3. Валенда Н.А. «Метод формализации семантического словаря на основе функций». 2013. URL: http://www.hups.mil.gov.ua/periodic-app/article/11166 (дата звернення: 24.03.2020)

4. Осуолт М., Эделман Дж., Лоу С. С.. Автоматизация программируемых сетей. – Москва: ДМК-Пресс, 2019. – 616 с.

5. Evans Miriti, Andrew Mwaura. “Dynamic Vector Quantization for Reinforcement Learning (DVQRL)”. 2018. URL: https://ieeexplore.ieee.org/document/8703233 (дата звернення: 24.03.2020)

6. Rui Zhang, Feiping Nie, Xuelong Li. “Auto-weighted two-dimensional principal component analysis with robust outliers”. 2017. URL: https://ieeexplore.ieee.org/document/7953321 (дата звернення: 24.03.2020)

7. Афанасьева И.В. «Использование свёрточных нейронных сетей для классификации данных». 2018. URL: https://er.nau.edu.ua/bitstream/NAU/36749/1/%d1%87%d0%b0%d1%81%d1%82%d0%b8%d0%bd%d0%b07.pdf (дата звернення: 24.03.2020)

8. Sachin S. Gavankar, Sudhirkumar D. Sawarkar. “Eager decision tree”. 2017. URL: https://ieeexplore.ieee.org/document/8226246 (дата звернення: 24.03.2020)

9. Dandan Feng, Zhanfeng Deng, Tongxun Wang. “Identification of disturbance sources based on random forest model”. 2018. URL: https://ieeexplore.ieee.org/document/8602245 (дата звернення: 24.03.2020)

10. Yidong Liu, Yanzhi Wang, Fabrizio Lombardi. “An energy-efficient stochastic computational deep belief network”. 2018. URL: https://ieeexplore.ieee.org/document/8342191 (дата звернення: 24.03.2020)

11. Qingyang Xu, Zhe Wu, Yiqin Yang. “The difference learning of hidden layer between autoencoder and variational autoencoder”. 2017. URL: https://ieeexplore.ieee.org/document/7979344 (дата звернення: 24.03.2020)

12. Ivan Selesnick. “Sparse Regularization via Convex Analysis”. URL: https://ieeexplore.ieee.org/document/7938377 (дата звернення: 24.03.2020)