Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Факультет інформаційних технологій
(факультет)

Кафедра системного аналізу та управління
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня бакалавра

Студента Говорухи Андрія Юрійовича

академічної групи 124-19-2

спеціальності 124 Системний аналіз

на тему: «Дослідження поведінки споживача за допомогою методів машинного навчання»

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**Інституційною**	
кваліфікаційної роботи	*к.т.н., доц. Желдак Т.А.*			
розділів:				
Інформаційно-аналітичний	*к.т.н., доц. Желдак Т.А.*			
Спеціальний розділ	*к.т.н., доц. Желдак Т.А.*			
Рецензент	*д.т.н.,проф. Алексєєв М.А.*			
Нормоконтролер	*к.ф.-м.н., доц. Хом'як Т.В.*			

Дніпро
2023

ЗАТВЕРДЖЕНО:
завідувач кафедри
Системного аналізу та управління
(повна назва)

_____ *к.т.н., доц. Желдак Т.А.*
(підпис) (прізвище,ініціали)

«____» _____20___року

ЗАВДАННЯ
на кваліфікаційну роботу
ступеня *бакалавра*

студенту *Говорусі А.Ю.* **академічної групи** *124- 19-2*
спеціальності: *124 Системний аналіз*
на тему «*Дослідження поведінки споживача за допомогою методів машинного навчання»*
затверджену наказом ректора НТУ «Дніпровська політехніка»
від *16.05.2023 р. №350-с*

Розділ	Зміст	Терміни виконання
1. Інформаційно-аналітичний розділ	*Визначити предметну область дослідження та проблему, що розв'язується. Обґрунтувати методи для виконання поставлених завдань.*	23.03.2023 – 02.05.2023
2.Спеціальний розділ	*Розв'язати поставлені задачі: описати структуру об'єкта досліджень, створити систему для автоматизації класифікації поведінки споживачів.*	02.05.2023 – 27.05.2023

Завдання видано _____ *доц. Желдак Т.А.*
(підпис) (прізвище, ініціали)

Дата видачі: *05.01.2023 р.*
Дата подання до екзаменаційної комісії: *15.06.2023 р.*

Прийнято до виконання _____ Говоруха А.Ю.
(підпис студента) (прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 141 сторінки, 10 таблиць, 67 рисунків, 7 додатків, 30 джерел.

Виявлення проблем в існуючому підприємстві є одним із ключових факторів, який допомагає йому функціонувати та розвиватися. Але, зазвичай, вони не лежать на поверхні, тому що ланцюг взаємодії з покупцем містить безліч процесів: від вироблення товару до його транспортування. Щоб облегшити процес виявлення проблем, підприємство повинно дати можливість клієнтам робити зворотню реакцію, а після цього опрацьовувати її. Але обробляти сотні тисяч відгуків вручну вимагає створення окремої посади або додаткових навантажень на існуючих менеджерів.

Об'єкт дослідження: результат діяльності бразильського маркетплейс-сайту *«Olist»* за 2017-2018 роки.

Мета роботи: підвищення ефективності роботи сайту маркетплейсу *«Olist»* за рахунок класифікації споживачів на основі їхніх відгуків щодо купленого товару.

Предмет дослідження: створення автоматизованого класифікатора реакцій споживачів на основі їхніх відгуків щодо купленого товару на маркетплейс-сайті *«Olist»* за допомогою методів машинного навчання.

В *інформаційно-аналітичному розділі* розглянуто відомості про теперішнє положення методів обробки природньої мови (NLP), а також методів класифікації за допомогою машинного навчання.

У *спеціальному розділі* описано структуру об'єкта, алгоритми та реалізацію інформаційних систем побудови словників для обробки природньої мови та класифікаторів на їх основі.

Практична цінність роботи полягає в автоматизація процесу обробки зворотної реакції споживача, що зекономить грошові ресурси підприємства і дозволить перенаправити їх на вирішення виявлених проблем.

КЛАСИФІКАЦІЯ, СПОЖИВАЧ, ВІДГУК, ПРИРОДНЯ МОВА, МАШИНЕ НАВЧАННЯ, МОДЕЛЬ, WORD2VEC.

ABSTRACT

Explanatory note: 141 pages, 10 tables, 67 drawings, 7 apps, 30 sources.

Identifying problems in an existing business is one of the key factors that helps it work and develop. But, usually, they do not lie on the surface, because the chain of interaction with the buyer includes many processes: from the item production to its transportation. In order to facilitate the process of identifying problems, the company should allow customers to give feedback and then process it. But processing hundreds of thousands of reviews manually requires the creation of a new worker or increasing work load for existing managers.

Object of study: Result of the Brazilian marketplace Olist's activity in 2017-2018 years.

Objective: Increasing productivity of marketplace Olist's website by classifying consumers based on their review on the purchased product.

Subject of research: Creation of an automated classifier of the customer's reviews on purchased items on the marketplace site "Olist" using machine learning methods.

In the information-analytical section explored information on the current state of natural language processing (NLP) methods and classification methods using machine learning.

Special section the structure of the research object, the algorithms and the implementation of information systems for creating an embedded vocabulary for Natural Language Processing. Creating customer classifiers based on reviews processed with embedded vocabulary.

The practical value of the work consists in the automation of the process of processing the customer's feedback, which will save the company's financial resources and allow it to use in solving the identified problems.

CLASSIFICATION, CONSUMER, FEEDBACK, NATURAL LANGUAGE, MACHINE LEARNING, MODEL, WORD2VEC.

Зміст

Сторінка

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ШІ - Штучний інтелект

ML- Machine Learning

MRFR - Market research future

NLP - Natural language processing

ГМП - Гібридний машинний переклад

FAHQMT - Fully Automatic High Quality Machine Translation

NER- Named Entity Recognition

QA - Питально-відповідна

RNN - Recurrent Neural Network

LSTM - Long short-term memory

ОВМ - Опорно-векторна машина

МНК - Метод найменших квадратів

GBM - Gradient Boosting Machine

kNN - k-Nearest Neighbor

НМ - Нейронна мережа

POS - Part of Speech

BoW - Bag of Words

CBoW - Continious Bag of Words

Word2vec - Word to Vector

ВСТУП

На сьогоднішній день розвиток штучного інтелекту (ШІ) дозволяє використовувати його у все більшій кількості галузей життєдіяльності людини. Якщо на початку своєї історії у 50х роках програми ШІ розвивалися на замовлення державних підприємств, то, приближаючись до кінця 20-го сторіччя, розробку почали фінансувати приватні компанії, які бачили свою вигоду в цій галузі.

Поняття штучного інтелекту вміщає в себе різні гілки, які розвиваються окремо один від одного, і мають наступні назви: штучний інтелект, машинне навчання, глибоке навчання. У цій кваліфікаційній роботі розглядаються методи машинного навчання.

Метою цієї роботи є підвищення ефективності роботи сайту маркетплейсу «Olist» за рахунок класифікації споживачів на основі їхніх відгуків щодо купленого товару.

Актуальність вирішення цієї задачі полягає в автоматизації процесу обробки фідбеку покупців, що практично неможливо зробити вручну з великим об’ємом замовлень, паралельно з цим сортувати на негативні, позитивні відгуки. Автоматизація цього процесу зекономить керівництву гроші і час, а зекономлені гроші можуть бути направлені на вирішення знайдених проблем.

Об’єктом дослідження є результат діяльності бразильського маркетплейс-сайту «Olist» за 2017-2018 роки.

Щоб мати можливість досліджувати і корегувати роботу сайту з такою потужністю, менеджмент сайту повинен структуровано накопичувати інформацію різних складових сайту, таких як: інформація про товари, відгуки, інформація про продавців, покупців, про замовлення в цілому. Можна вважати, що накопичення таких даних є другорядним продуктом бізнесу, який може принести прибуток у майбутньому.

Сайт, роботу якого буде досліджено, окремо накопичував інформацію про свою діяльність. Було сформовано вісім наборів даних, серед яких: дані про замовлення, дані з чого складається замовлення, дані про відгуки, дані про оплату, дані продавців, дані покупців, дані географічних положень, дані про товар. Використовуючи накопичені дані, можна побачити як змінювалась динаміка продажів, рейтинги продавців, популярність товарів тощо.

Предметом дослідження є створення автоматизованого класифікатора реакцій споживачів на основі їхніх відгуків щодо купленого товару на маркетплейс-сайті *«Olist»* за допомогою методів машинного навчання. Наприклад, у відгуку покупець може вказати, що: товар якісний або навпаки - товар низької якості через брак або підробку популярних товарів; доставка відбувається з перевищенням зазначеного терміну тощо. На основі цього ми можемо поділити відгуки на позитивні і негативні – саме ця задача поділу (класифікації) буде вирішуватися у цій кваліфікаційній роботі.

Виконання роботи буде поділено на слідуючі етапи:

- провести аналіз науково-технічних публікацій та даних Інтернет джерел в області сучасних підходів для обробки природномовних текстів;

- провести порівняльний аналіз основних відомих методів для вирішення задачі класифікації;

- здійснити вибір моделі та методів для проведення дослідження;

- здійснити вибір набору даних для експериментальних досліджень;

- підготовка даних для безперешкодної роботи моделі;

- формування корпусу відгуків споживачів для експериментальних досліджень;

- реалізувати модель класифікації;

- провести обґрунтований вибір методів ембедінгу і класифікацій тексту за тональністю;

- здійснити дослідження можливостей реалізованого програмного додатку.

1. ІНФОРМАЦІЙНО-АНАЛІТИЧНИЙ РОЗДІЛ

1.1 Загальні відомості

У якості предметної області було обрано задачу дослідження і класифікація поведінки споживачів за їхніми відгуками. Класифікація відбувається методами машинного навчання на основі обробки природньомовних текстів.

Обробка природньої мови – це здатність комп’ютера вміти обробляти і розуміти людську мову, а саме як вона звучить і як пишуться слова. Це компонент штучного інтелекту (ШІ).

Хоча поняття Natural Language Processing існує більше 50 років і має коріння в галузі лінгвістики, історія машинного перекладу має зачатки ще у сімнадцятому столітті, коли такі філософи, як Декарт і Лейбніц, висунули пропозиції щодо кодів, які б зв’язували слова між мовами. Але усі ці пропозиції залишалися теоретичними, і жодна не приводила до створення справжньої машини аж до середини 20-го сторіччя.

Перші патенти на машини обробки природньої мови почали з’являтися у 30-х роках 20 сторіччя. Але постійне використання приладів машинного перекладу почалося наприкінці 60х років, коли компанія Latsec розробила систему Systran для ВПС США, робота якої базувалася на правилах – переклад кожного слова або конструкцій слів. На той час лідерами технічного розвитку можна вважати дві держави - США і СРСР, тому трендом того часу були системи перекладу технічної документації з російської на англійську, чим і займалась машина Systran. [1]

З часом список мов для перекладу розширився, наприклад, в Канаді використовувалася система перекладу з англійської на французьку, а разом із цим збільшилась кількість системи, які працювали на тих самих електронно-обчислювальних машинах Systran. Поширення мікрокомп’ютерів у 80-х роках з підвищенням обчислювальних потужностей знаменувало появлення дешевого ринку систем машинного обробки/перекладу на основі

статистичних моделей перекладу. Статистичні моделі базуються на порівняні великого обсягу мовних пар, які записані мовою яку перекладають і на яку перекладають. [1]

У тепершній час широко використовується гібридний машинний переклад/обробка (ГМП). ГМП використовує сильні сторони обох моделей обробки мови, в результаті якого користувач отримує переклад високою якості, завдяки перекладу на основі правил, та високу швидкість, яку надає статистичний метод. [2]

Якщо раніше товчком розвитку були державні замовлення США у 60-х роках, то у 80-х роках цьому сприяло поширення комп'ютерів. На тепершній час ситуація схожа, але тепер поширюються інтелектуальні пристрої, які можна помістити у кишеню.

Market Research Future (MRFR) очікує, що світовий ринок обробки природної мови зростатиме в середньому на 24% у період з 2017 по 2023 рік (прогнозований період). Такий тренд зумовлено тим, що за останні 10 років був виконаний масовий стрибок у цифровізації даних, збільшення використання інтелектуальних пристроїв та попит на поліпшення якості обслуговування клієнтів. Окрім вкладень у маркетингову галузь, очікується підвищення інвестицій у розробку моделей в галузі охорони здоров'я, що у подальшому приведе до розвитку ринку обробки природньої мови.

1.2 Способи використання природньої мови

1. Перше і найважливіше завдання для якого і створили цю галузь – це машинний переклад. Їй займаються дуже давно і є величезний прогрес, але завдання отримання повністю автоматичного перекладу високої якості (FAHQMT - Fully Automatic High Quality Machine Translation) так і залишається невирішеним. Це в якомусь сенсі двигун NLP, одне з найбільших завдань, яким можна займатися. Яскравим прикладом буде спроба перекласти одну з приказок англійською мовою (див. рисунок 1.1).[3]

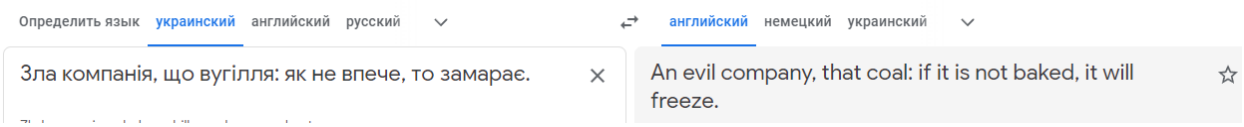

Рисунок 1.1. Приклад перекладу приказки.

Як видно, зміст приказки не перекладається відповидним чином, а це значить, що системі є чому вчитися.

2. Друге завдання – класифікація цілого тексту. Наприклад, дано набір текстів і задача класифікувати ці тексти за категоріями. За якими категоріями? Це питання до корпусу і до умов задачі.[3]

Корпус – це підібрана і опрацьована сукупність текстів, які використовуються як база для дослідження мови. Корпусом може бути набір відгуків або всі статті з сайту Wikipedia.

Тобто суть в тому, щоб було багато тексту, на основі якого можна було визначити взаємозв’язок між словами. Відповідно до того які категорії записано у корпус, на те і можна класифікувати дані.

Перший спосіб з практичної точки зору способів застосування класифікації - це сортування листів на пошті на спам і хам (не спам).

Другий варіант — класифікація за провідною темою тексту, наприклад, теми новин за категоріями (рубрикація) — політика, фінанси тощо.

Третій варіант застосування завдання текстової класифікації – сентиментний аналіз. Наприклад, класифікація відгуків на позитивні, негативні та нейтральні (див. рисунок 1.2)[3]

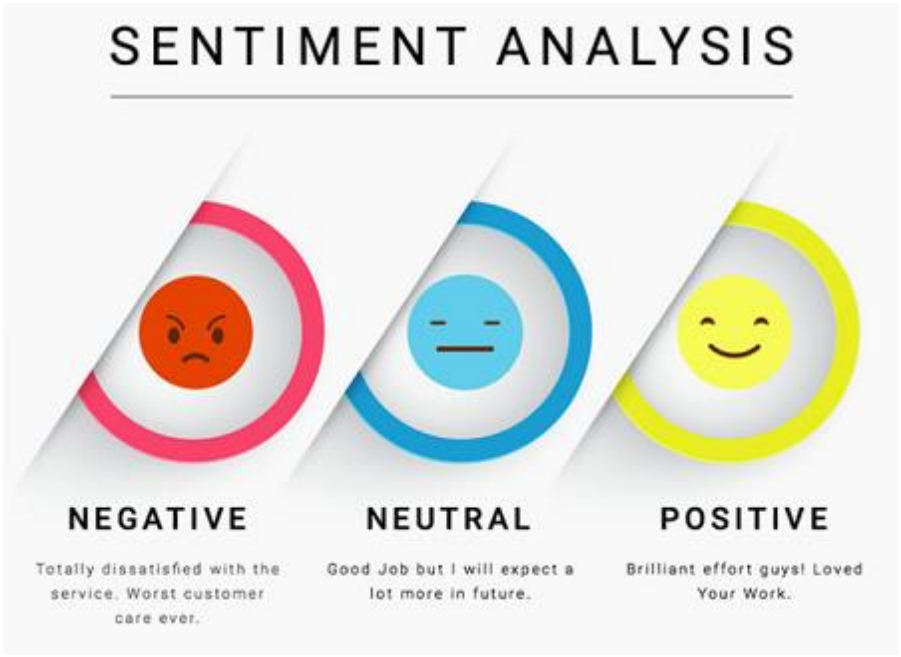

Рисунок 1.2. Класифікація клієнтів за тональністю їхніх відгуків.

Оскільки можливих категорій, на які можна ділити тексти, можна вигадати дуже багато, текстова класифікація є однією з найпопулярніших практичних завдань NLP.

3. Третє завдання – вилучення іменованих сутностей (NER). Ми виділяємо в тексті ділянки, які відповідають заздалегідь обраному набору сутностей, наприклад, треба знайти в тексті всі локації, особи та організації.[3] У тексті «Назар Петренко працює у кав'ярні «Magic Coffee» на вулиці Шевченка». З тексту отримуємо: особа «Назар Петренко», організація «Magic Coffee», локація «вулиця Шевченка». Одна із прикладних задач це структуризація неструктурованих даних або побудова запитально-відповідних систем (QA – система).

4. Два наступні завдання, тренд останніх часів. Це питання-відповіді та діалогові системи (чат-боти). Amazon Alexa, Сірі і набравший популярність ChatGPT – це традиційні приклади діалогових систем. Щоб вони нормально працювали, система повинна вміти вирішити багато завдань NLP. Наприклад, текстова класифікація допомагає визначити, чи ми

потрапляємо в один зі сценаріїв goal-oriented чат-бота. Наприклад, питання про курс валюти.

Relation extraction потрібно для визначення учасників шаблону сценарію, а завдання ведення діалогу загальні теми (“болталки”) допоможе нам у ситуації, коли ми не потрапили до жодного з сценаріїв.

Питально-відповідні системи - теж зрозуміла і корисна річ. Ви ставите запитання - машина шукає відповідь на нього у базі даних або корпусі текстів. Прикладами таких систем можуть бути IBM Watson або Wolfram Alpha.[3]

Перерахований список не обмежується лише цими задачами, але вони є найпопулярнішими при розв’язку задач NLP.

1.3 Методи обробки природньої мови

З розвитком мобільних телефонів, інтеграції все більшої частини життєдіяльності в телефон і створення автоматизованих чатів, так звані чат-боти, NLP стала однією з найважливіших технологій штучного інтелекту. Але повне розуміння мови надзвичайно складне завдання, оскільки людська мова має свої особливості: емоційність, тональність від чого може залежати семантика слів. В залежності від цього мову поділяють на деякі рівні обробки природньої мови, тому перерахуємо їх.

1. Істотна частина технологій NLP працює завдяки deep learning (глибокому навчанню) — галузь машинного навчання, яка почала набирати обертів лише на початку цього десятиліття з таких причин:

- Нагромаджено великі обсяги тренувальних даних;
- Створення нових моделей та алгоритмів з розширеними можливостями та покращеною продуктивністю;
- Розроблено обчислювальні потужності: багатоядерні CPU та GPU;

- З'явилися навчальні методи з використанням контексту, нові методи регуляризації та оптимізації.

Більшість методів машинного навчання добре працюють із-за розроблених людиною представлень (representations) даних та вхідних ознак, а також оптимізації коефіцієнтів, щоб зробити фінальне передбачення кращим.[4]

У глибокому навчанні алгоритм намагається автоматично отримати найкращі ознаки чи представлення із сирих вхідних даних.[4]

2. У традиційному NLP слова сприймаються як дискретні символи, які далі представляються у вигляді one-hot векторів. Проблема зі словами – дискретними символами – це відсутність визначення схожості для one-hot векторів. Тому альтернативою є кодування схожості у вектори.

Векторне моделювання - метод уявлення рядків, як векторів зі значеннями. Будується щільний вектор (dense vector) для кожного слова так, щоб слова, які зустрічаються в схожих контекстах, мали схожі вектори.

Одна із реалізацій такого принципу є метод Word2Vec, де використовується великий масив речень (корпус) для навчання моделі. Алгоритм є контекстно-залежним, тому що навчання моделі відбувається за допомогою вирішення задачі вибору правильного контексту для заданого слова. Модель вивчає локальний контекст навколо слова у вікні заданої розмірності, і потім намагається спрогнозувати його.[4]

Існує також покращена версія, а саме метод GloVe (global vectors for word representation) означає «глобальні вектори подання слів». Це алгоритм навчання без вчителя, який був розроблений у Стенфорді. Його основна ідея полягає у тому, щоб отримати семантичні відносини між словами, використовуючи матрицю спільного використання. Ідея дуже схожа на word2vec, але є невеликі відмінності. GloVe використовує для навчання контексту весь набір речень і витягує контекст навколо слова, а Word2Vec

лише локальний, тобто в межах ковзного вікна. GloVe має перевагу в тому, що може знаходити більше слів синонімів, тому що аналізує більшу кількість варіантів контексту. Як було зазначено, схожі слова мають схожі вектори, тому дистанція між словами відповідно дорівнює семантичній різниці між ними. А так як використовується весь набір речень, то слова, які використовуються в однаковому контексті, можуть бути визначені як синоніми.[4]

1. Одновимірні згорткові нейронні мережі (рисунок 1.3) також застосовуються для аналізу текстів. Однак для цієї мети використовуються лише одновимірні.

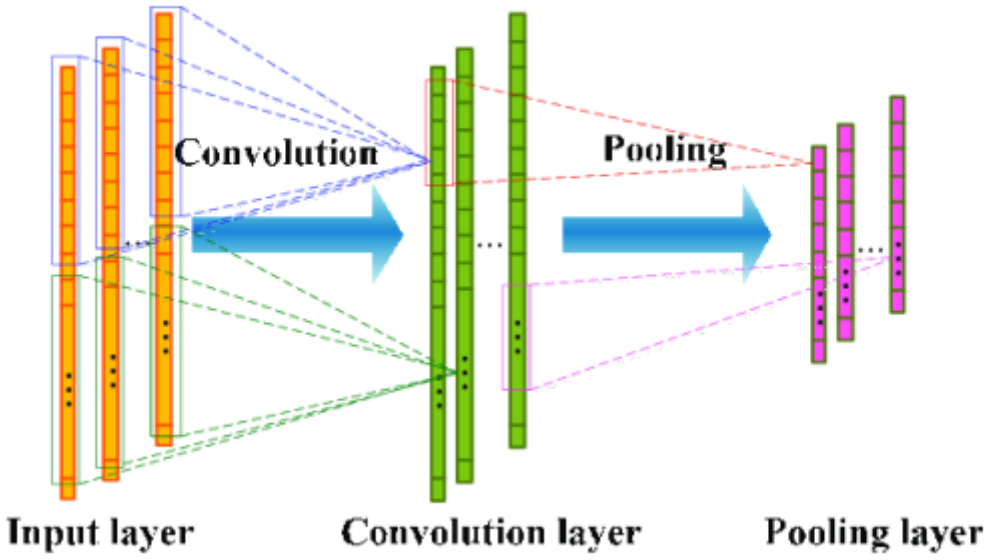

Рисунок 1.3 Архітектура одновимірної згорткової нейронної мережі

2. Як початковий рівень машинного перекладу, виконується співставлення двох текстів перекладених у різних мовах, для того, щоб визначити, як перекладається кожне слово. Але недоліком такого порівняння двох заздалегідь перекладених текстів є те, що вона не зберігає контекст, який присутній у тексті, який треба перекласти. Наприклад, у першому реченні абзацу вказується, що історія, яка описана в цьому абзаці, про вас. Але звичайне співставлення текстів не має пам’ять, щоб зберігати якісь

деталі з перекладеного тексту у минулому. На допомогу приходять рекурентні нейронні мережі .

RNN (Recurrent Neural Network) — нейромережа з залежністю від попередніх станів, яка має зв'язки між проходами. Нейрони одержують інформацію з попередніх шарів, а також із самих себе на попередньому кроці. Це означає, що порядок, в якому подається на вхід дані та тренується мережа, важливий: результат подачі "Дональд" - "Трамп" не збігається з результатом подачі "Трамп" - "Дональд". На рисунку 1.4 концептуально наведено алгоритм роботи нейронного мережного перекладу. Мережа зберігає інформацію про минуле слово і подає його на вхід для перекладу наступного слова. [5]

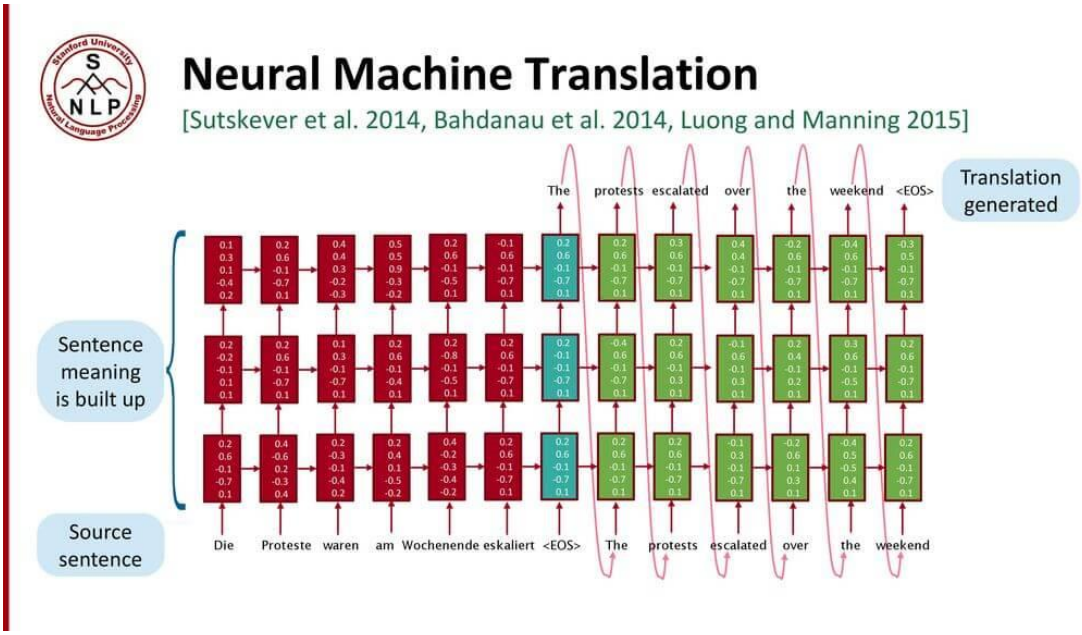

Рисунок 1.4. Концептуальна модель роботи алгоритму Нейронного машинного перекладу.

Головна проблема RNN є проблема затухання градієнта, коли з часом втрачається інформація про минуле. Інтуїтивно здається, що це не є серйозною проблемою, оскільки це лише коефіцієнти, а не стан нейронів. Але з часом нові коефіцієнти стають на місця, де зберігається інформація з минулого. Як наслідок, RNN будуть відчувати складності у запам'ятовуванні

слів, що стоять далі в послідовності, а прогнозування буде виконуватися на основі крайніх слів, що створює проблеми у смисловому навантажені перекладу. [5]

Для вирішення проблеми з запам’ятовуванням було створено модель мережі короткостроково-довгострокової пам'яті (Long short-term memory, далі LSTM). Рішення полягає у введені гейтів (gates) і комірок пам'яті. Кожен нейрон являє собою клітинку пам'яті з трьома гейтами: на вхід, на вихід та забування (forget). Ці елементи виконують функцію охоронців для інформації, дозволяючи або забороняючи її використання.

Вхідний гейт визначає, яка кількість інформації з попереднього шару зберігатиметься на теперішньому кроці. Вихідний гейт виконує роботу на іншому кінці та визначає, яка частина наступного шару дізнається про стан поточної комірки. Гейт забування контролює міру збереження значення у пам'яті: якщо при вивчені книги починається новий розділ, іноді для нейромережі стає необхідним забути деякі слова з попереднього розділу. На рисунку 1.5 зображено концептуально потік інформації у нейроні нейронної мережі.[5]

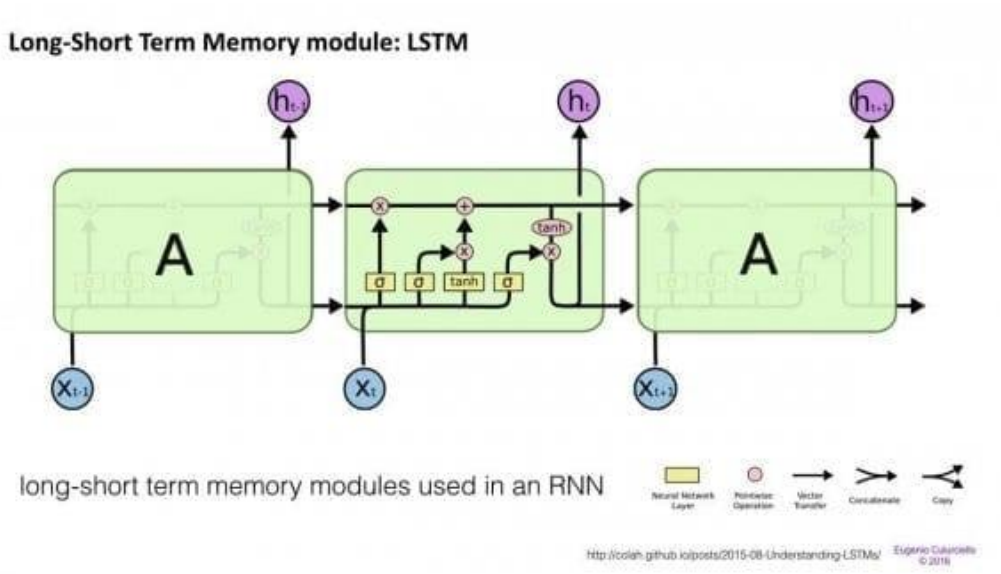

Рисунок 1.5. Концептуальна модель будови нейрона у LSTM.

Було показано, що LSTM здатні навчатися на складних послідовностях і, наприклад, писати в стилі Гоголя або складати примітивну музику.

На основі рекурентних нейронних мереж також можуть створюватися голосові помічники та системи питання-відповідь, які зараз широко використовуються у вигляді чат-ботів у месенджерах. Яскравим прикладом голосового помічника можна вважати систему «Окей, Гугл».

Отже, вище було наведено методи обробки природньої мови, а також приклади, де ми можемо зустрітися з автоматичною обробкою мови. Підбиваючи підсумок можна сказати, що нейронні мережі дедалі ширше використовуються у нашому повсякденному житті.

1.4 Класифікація інформації

1.4.1 Загальні відомості

Усі ми перевіряємо електронну пошту щодня, можливо, кілька разів. Корисною функцією більшості постачальників послуг електронної пошти є можливість автоматично відокремлювати спам від звичайних листів. Це приклад використання популярного завдання NLP (Natural Language Processing), відомого як класифікація тексту, якому присвячено цей розділ.

Класифікація тексту — це завдання присвоєння однієї чи кількох категорій певному фрагменту тексту з набору можливих категорій. У прикладі спам–ідентифікатора пошти ми маємо дві категорії — спам і не спам. Кожен вхідний електронний лист віднесено до однієї з цих категорій. Це завдання класифікації текстів на основі певних властивостей має широкий спектр застосувань у різних областях, таких як соціальні мережі, електронна комерція, охорона здоров’я, право та маркетинг тощо.

Незважаючи на те, що мета та застосування класифікації тексту можуть відрізнятися від домену(галузь знань, наприклад, комерція, реклама) до домену, основна абстрактна проблема залишається незмінною. Ця інваріантність основної проблеми та її застосування у безлічі областей

робить класифікацію тексту найбільш широко використовуваним завданням NLP у промисловості та найбільш дослідженим в академічних колах. У цьому розділі ми обговоримо корисність класифікації тексту, методи підготовки даних для класифікації і методи самої класифікації.[6]

У машинному навчанні класифікація — це проблема класифікації екземпляра даних в один або кілька відомих класів. Дані можуть мати різний формат, наприклад текст, мову, зображення або число. Класифікація тексту — це особливий випадок проблеми класифікації, де точка вхідних даних є текстом, а мета полягає в тому, щоб класифікувати фрагмент тексту в одне або більше сегментів (званих класами) із набору попередньо визначених сегментів (класи).

Текст може мати довільну довжину: символ, слово, речення, абзац або повний документ. Розглянемо сценарій, коли ми хочемо класифікувати всі відгуки клієнтів про продукт на дві категорії: позитивні і негативні. Завдання класифікації текстів полягає в тому, щоб «вивчити» цю категоризацію з колекції прикладів для кожної з цих категорій і передбачити категорії для нових, небачених продуктів і нових відгуків клієнтів. Ця класифікація не завжди призводить до однієї категорії, і може бути будь-яка кількість доступних категорій. Але, наприклад, якщо нам потрібна виконати бінарну класифікацію, то в результаті клас буде той, у якого більше значення вірогідності бути відповідним класом.[6]

1.4.2 Основні задачі класифікації

Класифікація тексту викликала інтерес у багатьох прикладних сценаріях, починаючи від ідентифікації автора невідомого тексту в 1800-х роках і закінчуючи зусиллями USPS у 1960-х роках виконати оптичне розпізнавання символів за адресами та поштовими індексами [6]. У 1990-х роках дослідники почали успішно застосовувати алгоритми ML для класифікації тексту для великих наборів даних. Фільтрування електронної

пошти, широко відоме як «класифікація спаму», є одним із найперших прикладів автоматичної класифікації тексту, який впливає на наше життя донині. Від ручного аналізу текстових документів до суто статистичних, комп'ютерних підходів і найсучасніших глибоких нейронних мереж, ми пройшли довгий шлях у класифікації тексту. Давайте коротко обговоримо деякі з популярних програм, перш ніж заглибитися в різні підходи до класифікації тексту. Ці приклади також будуть корисні для виявлення проблем, які можна вирішити за допомогою методів класифікації тексту у вашій організації.

- Класифікація та організація змісту

Це стосується завдання класифікації/позначення тегами великих обсягів текстових даних. Це, у свою чергу, використовується для забезпечення таких випадків використання, як організація вмісту, пошукові системи та системи рекомендацій тощо. Приклади таких даних включають веб-сайти новин, блоги, онлайн-книжкові полиці, огляди продуктів, твіти тощо; позначення тегами опису продукту на веб-сайті електронної комерції; направлення запитів на обслуговування клієнтів у компанії до відповідної групи підтримки; і систематизація електронних листів на особисті, соціальні та рекламні листи в Gmail – усе це приклади використання класифікації тексту для класифікації та організації вмісту.

- Електронна комерція

Клієнти залишають відгуки про низку продуктів на веб-сайтах електронної комерції, таких як Amazon, eBay тощо. Прикладом використання класифікації тексту в такому сценарії є розуміння й аналіз сприйняття продукту чи послуги клієнтами на основі їхніх коментарів. Це широко відомо як «аналіз настроїв». Його широко використовують бренди по всьому світу, щоб краще зрозуміти, наближаються вони до своїх клієнтів чи віддаляються від них. Замість того, щоб класифікувати відгуки клієнтів як просто позитивні, негативні чи нейтральні, з часом аналіз настроїв перетворився на

більш складну парадигму: аналіз настроїв на основі «аспектів». Щоб зрозуміти це, розглянемо відгук клієнтів про ресторан, показаний на рисунку 1.6.

Рисунок 1.6. Приклад відгуку

Чи можна назвати відгук на рисунку 1.6 негативним чи позитивним? Важко на це відповісти — їжа була чудова, але обслуговування було погане. Практики та бренди, які працюють з аналізом настроїв, зрозуміли, що багато продуктів або послуг мають багато аспектів. Щоб зрозуміти загальні настрої, важливе розуміння кожного аспекту. Класифікація тексту відіграє важливу роль у виконанні такого детального аналізу відгуків клієнтів.

1.4.3 Методи класифікації

Задача класифікації виникла ще за довго до появлення електронно-обчислювальних машин і програм до них. Люди поділяли речі на деякі категорії, тому що так легше обробляти інформацію, яка пов’язана з цими речами. Наприклад, складали схожі речі в одну коробку або зобов’язувати членів команди виконувати роботу в сфері, в якій вони вже працюють.

Переходячи від прикладів з життя до математики, можна сказати, що класифікація виконується на визначеній множині. А першим і простішим способом класифікації можна вважати метод заснований на теоремі Байєса.

1. Метод Наївного Баєса

Теорема Баєса і баєсівської ймовірності була представлена в публікації Томаса Баєса у 1763 році, але до розвинуте і сучасне формулювання було представлене у книзі П’єра-Симона Лапласа у 1812 році.

В основі байєсіанського підходу лежать концепції апріорної (безумовної) та апостеріорної (умовної) ймовірностей. Апріорна ймовірність теорії - це початковий ступінь впевненості суб'єкта в її істинності, апостеріорна ймовірність - ступінь впевненості суб'єкта після отримання нових дослідних даних. Зміна ймовірності гіпотези може бути формалізована за допомогою так званого простого принципу обумовлення. Його можна сформулювати наступним чином: при апріорній ймовірності P_{r_i} після отримання нових досвідчених даних, представлених твердженням e (за умови, що початкова ймовірність e була вищою за нуль), принципи раціональності вимагають переоцінки апріорної ймовірності P_{r_i} і введення апостеріорної ймовірності P_{r_f} так, що $P_{r_f}(h) = P_{r_i}(h|e)$, де h - будь-яка гіпотеза. Простий принцип обумовлення близький до теореми Байєса; він показує, що різниця між апостериорною і апріорною ймовірністю гіпотези h може бути зафіксована як кількісна оцінка того, якою мірою досвідчені дані e підтримують h.[7]

Завдяки формулюванню самої теореми Баєса, стало можливим використати формулу як класифікатор. Запишемо формулою теорему Баєса і позначимо складові його формули:

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}, \qquad (1.1)$$

де $P(A) -$ апріорна вірогідність гіпотези А,

$P(A|B)$ −вірогідність гіпотези А після виконання події В (апостеріорна вірогідність або правдоподібність події А при виконані В),

$P(\mathrm{B}|\mathrm{A}) -$ вірогідність події В, якщо гіпотеза А істинна (правдоподібність події В при виконані А),

$P(\mathrm{B})$ – повна вірогідність виконання дії B.

Також словами можна описати теорему наступним чином: вірогідність гіпотези A, після виконання події B буде дорівнювати відношенню апріорних ймовірностей A і B помножених на правдоподібність події B за умови виконання A. Як приклад приведемо наступну ситуацію: Нехай подія $\boldsymbol{B}$ – машина не заводиться, а гіпотеза $\boldsymbol{A}$ – у баку немає палива. Очевидно, що ймовірність $P(B|A)$ того, що машина не заведеться, якщо в баку немає палива, дорівнює одиниці. Як наслідок, апостеріорна ймовірність, що в баку немає палива, якщо машина не заводиться, тобто $P(A|B)$, дорівнює $\frac{P(A)}{P(B)}$, тобто відношенню апріорної ймовірності, що в баку немає палива, до ймовірності, що машина не заводиться. Наприклад, якщо апріорна ймовірність, що в баку немає палива, дорівнює 0,01, а ймовірність, що машина не заводиться, дорівнює 0,02, і випадково обрана машина не завелася, то ймовірність, що в баку немає палива, дорівнює 0, 5.[8]

Сформулювавши теорему можемо перейти до методу класифікації Наївним Баєсом. Наївний Баєс — це проста техніка для побудови класифікаторів: моделей, які призначають мітки класів екземплярам проблеми, представлених у вигляді векторів значень ознак, де мітки класів беруться з деякого кінцевого набору. Існує не єдиний алгоритм для навчання таких класифікаторів, а сімейство алгоритмів, заснованих на загальному принципі: усі наївні класифікатори Баєса припускають, що значення окремої ознаки не залежить від значення будь-якої іншої ознаки, заданої змінною класу. Наприклад, фрукт можна вважати яблуком, якщо він червоний, круглий і діаметром близько 9 см. Наївний класифікатор Баєса розглядає кожну з цих ознак як таку, що незалежно впливає на ймовірність того, що цей фрукт є яблуком, незалежно від будь-яких можливих кореляцій між ознаками кольору, округлості та діаметра. [9]

Байєсовський класифікатор належить до розряду методів машинного навчання. Наведемо приклад: система, перед якою стоїть завдання визначити, чи є наступний лист спамом, заздалегідь навчена якоюсь кількістю листів точно відомих де спам, а де не спам. Вже стало зрозуміло, що це навчання з учителем, де у ролі вчителя виступаємо ми. Байєсовський класифікатор представляє документ (у нашому випадку лист) у вигляді набору слів, які нібито не залежать один від одного (тут витікає та сама наївність).[10]

Необхідно розрахувати оцінку для кожного класу (спам/не спам) та вибрати ту, яка вийшла максимальною. Для цього використовуємо таку формулу:

$$\arg\max\left[P(Q_k)\prod_{i=1}^{n}P(x_i|Q_k)\right] \qquad (1.2)$$

$P(Q_k) - \dfrac{\text{число документів } Q_k}{\text{загальна кількість документів}}$

$P(x_i|Q_k) = \frac{\alpha+N_{ik}}{\alpha M+N_k}$ – входження слова x_i у документ класу Q_k (зі згладжуванням)*.

N_k – кількість слів, що входять до документа класу.

M – кількість слів з навчальної вибірки.

N_{ik} – кількість входжень слова x_i до документа класу Q_k.

α – параметр для згладжування.

Параметр згладжування потрібений на той випадок, коли значення з документу неможливо знайти в навчальній вибірці. Це може призвести до того, що оцінка дорівнюватиме нулю і документ не можна буде віднести до жодної з категорій (спам/не спам). Як би ви не хотіли, ви не зможете навчити свою модель всім можливим словам. Для цього необхідно застосувати згладжування, а точніше – зробити невеликі поправки у всі ймовірності входження слів у документ. Вибирається параметр $0 < \alpha \leq 1$ (якщо $\alpha = 1$, це згладжування Лапласа).

Коли обсяг тексту дуже великий, доводиться працювати з дуже малими числами. Щоб цього уникнути, можна перетворити формулу за властивістю логарифму: [10]

$$\log ab = \log a + \log b \tag{1.3}$$

Підставляємо значення з формули 1.2 у 1.3 і отримаємо:

$$\arg\max \left[\log P(Q_k) + \sum_{i=1}^{n} \log P(x_i|Q_k) \right] \tag{1.4}$$

Перейдемо від теорії до практики розв’язавши просту задачу класифікації на спам/не спам.

Навчальна вибірка:

Спам: «Путівки за низькою ціною»; «Знижка! Купи шоколадку та отримай телефон у подарунок».

Не спам: «Завтра відбудуться збори»; «Купи кілограм яблук та шоколадку».

Завдання: визначити, до якої категорії віднести такий лист: «У крамниці гора яблук. Купи сім кілограм та шоколадку»

Розв’язання: Складаємо таблицю (див. Таблиця 1.1). Видаляємо усі "стоп-слова", розраховуємо ймовірності, параметр для згладжування $\alpha = 1$.

Стоп-слова - це слова, знаки, символи, які самостійно не несуть жодного смислового навантаження та просто ігноруються пошуковими системами.

Таблиця 1.1

Таблиця розрахунків

		Кількість входження в "спам"	Кількість входження в "не спам"	$P(X_i\|\text{Спам})$	$P(X_i\|\text{Не спам})$
Слова наявні в навчальній вибірці	Путівки	1	0		
	низькою	1	0		
	ціною	1	0		
	Знижка	1	0		
	Купи	1	1	$\frac{1+1}{14+9}$	$\frac{1+1}{14+7}$
	шоколадку	1	1	$\frac{1+1}{14+9}$	$\frac{1+1}{14+7}$
	отримай	1	0		
	телефон	1	0		
	подарунок	1	0		
	Завтра	0	1		
	відбудуться	0	1		
	збори	0	1		
	кілограм	0	1	$\frac{1+0}{14+9}$	$\frac{1+1}{14+7}$
	яблук	0	1	$\frac{1+0}{14+9}$	$\frac{1+1}{14+7}$
	крамниці	0	0	$\frac{1+0}{14+9}$	$\frac{1+0}{14+7}$
	гора	0	0	$\frac{1+0}{14+9}$	$\frac{1+0}{14+7}$
	сім	0	0	$\frac{1+0}{14+9}$	$\frac{1+0}{14+7}$

Запишемо розрахунок ймовірності потрапляння в кожну групу повідомлень.

Ймовірність потрапити в групу «Спам»:

$$\frac{2}{4}\cdot\frac{2}{23}\cdot\frac{2}{23}\cdot\frac{1}{23}\cdot\frac{1}{23}\cdot\frac{1}{23}\cdot\frac{1}{23}\cdot\frac{1}{23} \approx 0{,}000000000587 \text{ (або } 5{,}87E-10)$$

Ймовірність потрапити в групу «Не спам»:

$$\frac{2}{4}\cdot\frac{2}{21}\cdot\frac{2}{21}\cdot\frac{2}{21}\cdot\frac{2}{21}\cdot\frac{1}{21}\cdot\frac{1}{21}\cdot\frac{1}{21} \approx 0{,}00000000444 \text{ (або } 4{,}44E-9)$$

Отже, тепер ми можемо порівняти дві ймовірності і вирішити до якої групи віднести наше повідомлення. Так як $4{,}44E-9$ більше $5{,}87E-10$, можна вважати, що повідомлення не є спамом.

Те саме розрахуємо і за допомогою функції, перетвореної за властивістю логарифму:

Оцінка для категорії «Спам»:

$$\log\frac{2}{4}+\log\frac{2}{23}+\log\frac{2}{23}+\log\frac{1}{23}+\log\frac{1}{23}+\log\frac{1}{23}+\log\frac{1}{23}+\log\frac{1}{23}\approx -9{,}23$$

Оцінка для категорії «Не спам»:

$$\log\frac{2}{4}+\log\frac{2}{23}+\log\frac{2}{23}+\log\frac{1}{23}+\log\frac{1}{23}+\log\frac{1}{23}+\log\frac{1}{23}+\log\frac{1}{23}\approx -8{,}35$$

Відповідь: аналогічно розрахункам попереднім методом, ймовірність заданих повідомлень бути в категорії «Не спам» вище, ніж «Спам.

2. Метод опорних векторів

У машинному навчанні метод опорних векторів — це метод аналізу даних для регресійного аналізу або їх класифікації за допомогою моделей з керованим навчанням з пов'язаними алгоритмами навчання, які називаються опорно-векторними машинами (ОВМ). Для заданого набору тренувальних зразків, для кожного з яких вказано приналежність до однієї з двох категорій. Далі алгоритм тренування ОВМ будує модель, яка відносить нові зразки до однієї чи іншої категорії, роблячи це неймовірнісним бінарним лінійним класифікатором. Модель ОВМ є представленням зразків як точок у просторі, відображених таким чином, що зразки з окремих категорій розділено чистою гіперплощиною (прогалиною), яка є щонайширшою (див. рисунок 1.7). Нові зразки тоді відображуються до цього ж простору, й робиться передбачення про їхню належність до категорії на основі того, на який бік прогалини вони потрапляють. [11]

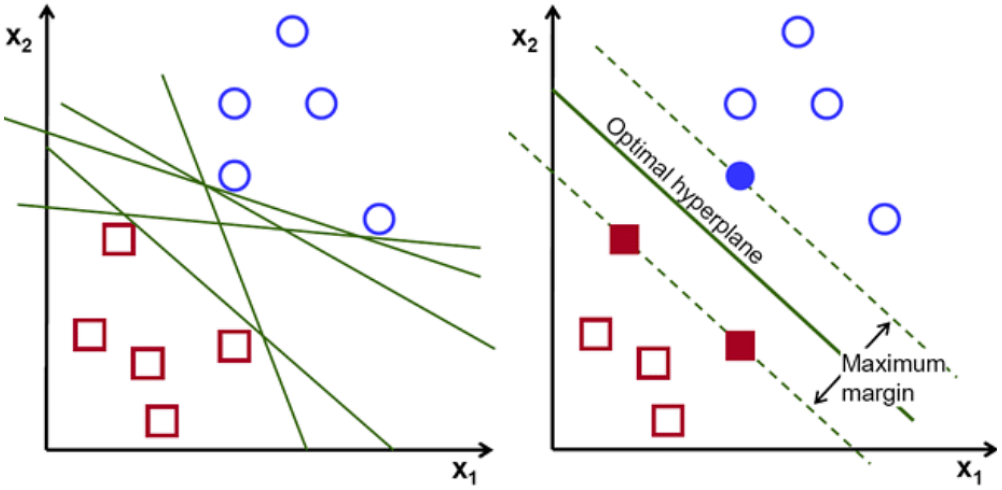

Рисунок 1.7 Приклад роботи алгоритму опорних-векторів.

Математично метод записується наступним чином:

У нас є тренувальний набір даних з n точок вигляду:

$$(\overrightarrow{x_1}, y_1), \ldots, (\overrightarrow{x_n}, y_n) \tag{1.5}$$

Де y_i дорівнює або 1, або -1, і кожен з них вказує на клас, до якого належить точка $\overrightarrow{x_i}$. Кожен $\overrightarrow{x_i}$ є $p-$вимірним дійсним вектором. Нам треба знайти «максимально розділову гіперплощину», яка відділяє групу точок $\overrightarrow{x_i}$, для яких $y_i = 1$, від групи точок, для яких $y_i = 1$, і визначається таким чином, що відстань між цією гіперплощиною та найближчою точкою $\overrightarrow{x_i}$ з кожної з груп є максимальною.

Будь-яку гіперплощину може бути записано як множину точок $\overrightarrow{x_i}$, які задовільняють

$$\vec{w} \cdot \vec{x} - b = 0, \tag{1.6}$$

де $\vec{w}$ є (необов'язково нормалізованим) вектором нормалі до цієї гіперплощини. Параметр $\frac{b}{\|w\|}$ визначає зсув гіперплощини від початку координат вздовж вектора нормалі $\vec{w}$. [11]

Коли ми маємо два класи між якими треба розподілити точки, то класи будуть мати окремі гіперплощини, де для першої площини $\boldsymbol{y}$ буде дорівнювати 1, а в іншій – 1.

Між цими двома гіперплощинами існує розділова гіперплощина, яку потрібно максимізувати. Саме в цьому і полягає сенс методів опорних векторів.

Гіперплощини різних класів будуть записуватися наступним чином:

$$\vec{w} \cdot \vec{x} - b = 1 \text{ (будь – що на або вище цієї межі належить до класу з міткою 1)} \quad (1.7)$$

Та

$$\vec{w} \cdot \vec{x} - b = -1 \text{ (будь – що на або нижче цієї межі належить до класу з міткою – 1)} \quad (2.8)$$

З геометричної точки зору, відстанню між цими двома гіперплощинами є $\frac{b}{\|\vec{w}\|}$ тож для максимізації відстані між ними нам треба мінімізувати $\|\vec{w}\|$. Оскільки ми також маємо завадити потраплянню точок даних до розділення, ми додаємо таке обмеження: для кожного $\boldsymbol{i}$, або [11]

$$\vec{w} \cdot \vec{x} - b \geq 1, \text{ якщо } y_i = 1 \quad (1.9)$$

або

$$\vec{w} \cdot \vec{x} - b \leq -1, \text{ якщо } y_i = -1 \quad (1.10)$$

Вказані обмеження стверджують, що кожна точка мусить лежати з правильного боку розділення. Ці дві нерівності можна записати як одну:

$$y_i \cdot (\vec{w} \cdot \vec{x_i} - b) \geq 1, \text{для всіх } 1 \geq i \geq n. \quad (1.11)$$

Ми можемо зібрати це докупи, щоб отримати задачу оптимізації:

«Мінімізувати $\|\vec{w}\|$ за умови $y_i \cdot (\vec{w} \cdot \vec{x_i} - b) \geq 1$ для $i = 1, \dots, n$»

$\vec{w}$ і b, які розв'язують цю задачу, визначають класифікатор $\vec{x} \to sgn(\vec{w} \cdot \vec{x} - b)$

Важливим наслідком цього геометричного опису є те, що максимально розділова гіперплощина повністю визначається тими $\vec{x_i}$, які лежать найближче до неї. Ці $\vec{x_i}$ називають опорними векторами. [11]

Для розширення ОВМ на випадки, в яких дані не є лінійно роздільними, ми вводимо завісну функцію втрат.

$$\max\left(0, 1 - y_i \cdot (\vec{w} \cdot \vec{x_i} - b)\right) \tag{1.12}$$

Зауважимо, що y_i є i-ю цілю (тобто, в нашому випадку, 1 або -1) та $(\vec{w} \cdot \vec{x} - b)$ - поточний результат.

Ця функція є нульовою, якщо обмеження в (2.11) задовольняється, іншими словами, якщо $\vec{x_i}$ лежить в правильній площині розділення. Для даних із неправильної площини розділення значення цієї функції є пропорційним до відстані від розділення.

Тоді нам треба мінімізувати

$$\left[\frac{1}{n} \sum_{i=1}^{n} \max\left(0, 1 - y_i \cdot (\vec{w} \cdot \vec{x_i} - b)\right)\right] + \lambda \|\vec{w}\|^2 \tag{1.13}$$

де параметр λ визначає компроміс між збільшенням розміру розділення та забезпеченням того, що $\vec{x_i}$ лежить у правильній площині розділення.

Таким чином, для достатньо малих значень λ ОВМ із м'яким розділенням поводитиметься однаково з ОВМ із жорстким розділенням, якщо вхідні дані є лінійно класифіковними, але, тим не менше, вчитиметься життєздатного правила класифікації, якщо ні. [11]

Первинний алгоритм максимально розділової гіперплощини, запропонований В. Вапником у 1963 році, будував лінійний класифікатор. Проте 1992 року Б. Босер, І. Гійон та В. Вапник запропонували спосіб створювати нелінійні класифікатори шляхом застосування до максимально розділових гіперплощин ядрового трюку. Отриманий алгоритм є формально аналогічним, за винятком того, що кожен скалярний добуток замінено нелінійною ядровою функцією.

Ядрова функція — це вагова функція, що використовується в непараметричних методах оцінки. Вони можуть діяти в неявному просторі ознак високої вимірності навіть без обчислення координат даних у цьому просторі.

Це дозволяє алгоритмові узгоджувати максимальну гіперплощину, яка розділяє каси у перетвореному просторі ознак. Перетворення може бути нелінійним, а перетворений простір — великої вимірності; хоч класифікатор і є гіперплощиною в перетвореному просторі ознак, у первинному вхідному просторі він може бути нелінійним. [11]

До деяких найпоширеніших ядер належать:

- Поліноміальне (однорідне) : $k(\overrightarrow{x_i}, \overrightarrow{x_j}) = (\overrightarrow{x_i}, \overrightarrow{x_j})^d$,
- Поліноміальне (неоднорідне) : $k(\overrightarrow{x_i}, \overrightarrow{x_j}) = (\overrightarrow{x_i} \cdot \overrightarrow{x_j} + 1)^d$,
- Сигмоїдне (гіперболічний тангенс) : $k(\overrightarrow{x_i}, \overrightarrow{x_j}) = tanh(k\,\overrightarrow{x_i} \cdot \overrightarrow{x_j} + c)$, для деяких (не всіх) $k > 0$ та $c < 0$.

До потенційних недоліків методу ОВМ включають наступні аспекти:

1. Вимагають повністю розмічених вхідних даних;
2. Ненормовані ймовірності приналежності точок до класів;
3. ОВМ застосовні напряму лише до двокласових задач. Отже, мають застосовуватися алгоритми, які зводять багатокласову задачу до кількох бінарних задач;
4. Інтерпретувати параметри розв'язаної моделі важко.

3. Дерево прийняття рішень

Дерево прийняття рішень (або дерево класифікації) — засіб для прийняття рішень, який застосовується у машинному навчанні. Структура дерева складається з «листя» і «гілок». На ребрах («гілках») дерева рішення записані ознаки, від яких залежить цільова функція, а в «листі» записано значення цільової функції. Замість листя на гілці може буде ще одна гілка зі своїм значення ознаки, за якою відрізняються випадки. Щоб класифікувати новий випадок, треба спуститись по дереву до листя і отримати значення, яке

в ньому записано. [12] На рисунку 1.8 показано принцип роботи такого алгоритму.

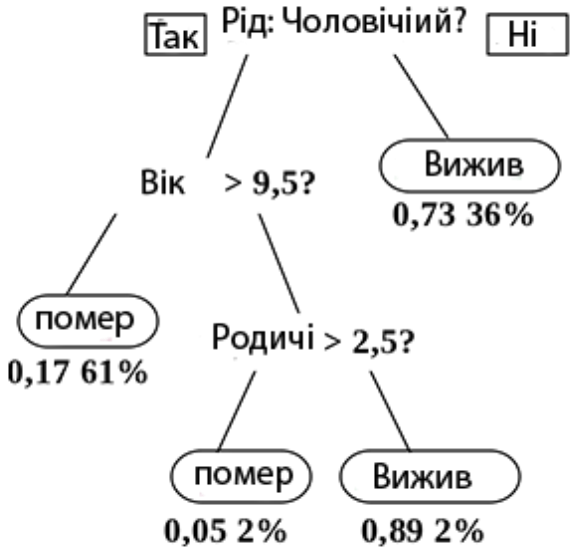

Рисунок 1.8. Приклад дерева рішень.

Такі і подібні до цього дерева рішень широко використовують в інтелектуальному аналізі даних. Мета класифікаторів полягає в тому, щоб створити модель, яка передбачає значення цільової змінної на основі декількох змінних на вході.

У кожному листі записано значення цільової функції, зміненої в ході руху від кореня по ребрах дерева до листа. Кожен внутрішній вузол зіставляється з одним із вхідних змінних.[12]

Відповідно до задач (класифікація або регресія), дерева поділяються на типи:

- Дерева для класифікацій, коли прогнозований результат є клас, якому належать дані;
- Дерева для регресії, коли прогнозований результат є дійсним числом.

Опишемо побудову моделі дерева рішень, спочатку опишемо можливий вигляд дерев. Як ми бачили на рисунку 1.8, кожен вузол має дві

гілки, якщо він не має гілок – він перетворюється на листя. Відповідно, якщо ми маємо датасет, який потрібно класифікувати, і в ньому знаходиться 3 признаки і 1 атрибут, який відповідає за відповідність класу, дерево для такого датасету може мати 3 вузли. Кожен вузол буде мати стільки гілок, скільки унікальних значень признаку існує. Розбиття закінчується коли закінчується список признаків.

Щоб модель мала можливість класифікувати дані, ми повинні навчити її цьому. Насьогодні існує велика кількість алгоритмів навчання: ID3, CART, C4.5., C5.0, NewId, ITrule, CN2 тощо. Охарактеризуємо коротко

Опишемо початок вибору признаків, які потрібно додати в модель. Для цього потрібно для кожного атрибуту розрахувати значення інформаційної ентропії.

$$H = -\sum_{i=1}^{n} \frac{N_i}{N} \log(\frac{N_i}{N}) \quad (1.14)$$

де n – кількість класів у вхідній підмножині, $N_i -$ кількість прикладів $i -$ го класу, $N -$ загальна кількість прикладів у підмножині.

Ентропія сприймається як міра неоднорідності класів у представленій підмножини. І якщо класи представлені у рівних частках, а невизначеність класифікації найбільша, то ентропія теж максимальна. Логарифм від одиниці перетворюватиме ентропію в нуль, якщо всі приклади вузла відносяться до одного класу. [13]

Якщо обраний атрибут розбиття забезпечує максимальне зниження ентропії результуючого підмножини щодо батьківського, його можна вважати найкращим.

Але насправді про ентропію говорять рідко. Фахівці приділяють увагу зворотній величині – інформації. У такому разі найкращим атрибутом буде той, який забезпечить максимальний приріст інформації результуючого вузла щодо вихідного:[13]

$$Gain(A) = Info(S) - Info(S_a) \quad (1.15)$$

де $Info(S) -$ інформація, пов'язана з підмножиною S до розбиття, $Info(S_a) -$ інформація, пов'язана з підмножиною, яку отримано у результаті розбиття атрибуту A.

Завдання вибору атрибута в такій ситуації полягає у максимізації величини Gain(A), яку називають приростом інформації. Тому теоретико-інформаційний підхід також відомий за назвою «критерій приросту інформації.[13]

Ми навели формули вибору атрибуту, який потрібно додати в модель. Ці формули використовуються в алгоритмах навчання ID3, C4.5. Далі буде наведено принципи, за якими виконується навчання дерев.

Алгоритм навчання може працювати до такого стану, що для кожного елемента буде окреме листя, тобто число класів буде дорівнювати числу елемнтів. На практиці такі дерева не вдасться застосовувати через перенавченість, тому що кожному елементу відповідатиме свій листок і при потраплянні в систему невідомого елементу, результат буде непередбачений. В результаті такого навчання вийде набір правил, актуальний лише для цього прикладу. Перенавчання у разі дерева рішень має схожі з нейронними мережами наслідки. Воно точно розпізнаватиме приклади з навчання, але не зможе працювати з новими даними. Ще один мінус - структура перенавченого дерева складна і погано інтерпретується. [13]

Як у більшості методах пошуку рішень, існує значення точності розв'язку, яке дозволяє економити ресурс, не витрачаючи його на деталізоване рішення, коли вже досягнуто заданий рівень точності. Так і в цій ситуації – дерева рішень мають правила зупинки навчання, для цього використовують кілька підходів:

- Рання зупинка. Алгоритм зупиняється після досягнення заданого значення критерію (наприклад, відсоткової частки розпізнаних прикладів або

кількість ітерацій). Перевага методу – скорочення витрат на навчання. Головний недолік - рання зупинка негативно впливає на точність дерева. Через це багато фахівців радять віддавати перевагу відсіканню гілок.

- Обмеження глибини дерева. Зупинка алгоритму навчання відбувається після досягнення встановленої кількості розбиття у гілках. Цей підхід негативно позначається на точності дерева.
- Завдання мінімально допустимої кількості прикладів у вузлі. Встановлюється обмеження створення вузлів з числом прикладом менше заданого (наприклад, 5). У такому разі не створюватимуться тривіальні розбиття та малозначущі правила.

Цими підходами користуються рідко, тому що вони не гарантують кращого результату. Найчастіше вони працюють тільки в якихось певних випадках.

Але без обмеження «росту» дерево рішень буде становитися занадто великим і складним, що унеможливить подальшу інтерпретацію. А якщо робити вирішуючі правила для створення вузлів, у які потраплятимуть по 2-3 приклади, вони не втратять практичну цінність. [13]

У цій ситуації багато фахівців віддають перевагу альтернативному варіанту — побудувати всі можливі варіанти дерева, а потім вибрати ті, які забезпечують прийнятний рівень помилки розпізнавання, спираючись на адекватній інших параметрів, наприклад, глибини дерева. Основне завдання в такій ситуації – пошук найбільш вигідного балансу між складністю та точністю дерева. [13]

Але й тут є проблема: таке завдання відноситься до класу NP-повних задач, а вони, як відомо, ефективних рішень не мають. Тому вдаються до методу відсікання гілок, який реалізується наступним чином:

- Будівництво повного дерева, в якому листя містить приклади одного класу

- Визначення двох показників: відношення числа правильно розпізнаних прикладів до загального числа прикладів (відносну точність моделі) та число неправильно класифікованих прикладів (абсолютну помилку).
- Видалення листів та вузлів, які мало впливають на точність моделі.

Відсікання гілок проводять протилежно зростанню дерева, тобто знизу вгору, шляхом послідовного перетворення вузлів на листя.

Головна відмінність методу відсікання гілок від передчасної зупинки — це можливість знайти оптимальне співвідношення між точністю і зрозумілістю. При цьому потрібно більше часу на навчання, тому що в перед відсіканням гілок спочатку треба побудувати повне дерево. [13]

Модель машинного навчання дерев рішення CART (Classification and Regression Tree) використовує для вибору атрибуту формулу Джині.

$$Gini(Q) = 1 - \sum_{i=1}^{n} {p_i}^2 \quad (1.16)$$

де Q – результуюча множина, n – число класів у ньому, p_i – ймовірність $i-$го класу (виражена як відносна частота прикладів відповідного класу).

Значення показника Джині може змінюватися від 0 до 1. Якщо індекс дорівнює 0, отже, всі приклади вихідної множини відносяться до одного класу. Якщо дорівнює 1, значить класи представлені в рівних пропорціях і рівноймовірні. Оптимальним розбиттям вважається таке, для якого значення індексу Джині мінімальне. [13]

Моделі CART формуються шляхом вибору вхідних змінних і оцінки точок розділення цих змінних на дві гілки, доки не буде створено відповідне дерево. Вибір гілок виконується за допомогою жадібного методу, коли частина елементів множини рівна між собою, а декілька точок ні. Ці точки порівнюються між собою за функцією вартості, тобто оцінюється

ефективність вибору цих точок як розділяючих. Ця модель використовує те саме обрізання гілок і використовує кількість екземплярів класу як критерій для зупинки навчання – якщо кількість менше за мінімальне, то розділення за цим класом відхиляється.

Ця модель є нелінійним методом розв'язання, на рисунку 1.9 показано вигляд дерева, створеного за моделлю CAST. Коротко про задачу: треба спрогнозувати зарплатню бейсболіста, керуючись його віком (Years) і кількістю хомранів (HmRun).

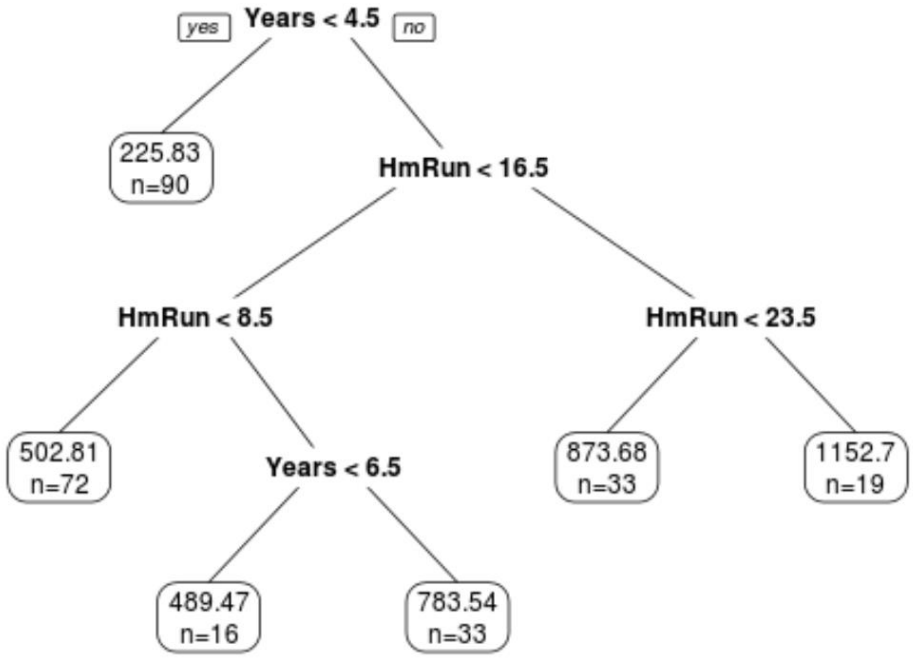

Рисунок 1.9. Дерево рішень для задачі зарплати бейсболіста.

Як ми бачимо дерево розрахувало гілки і навіть поверталося до атрибуту, який вираховувався раніше.

Однак недоліком моделей CART є те, що вони, як правило, мають високу дисперсію. Тобто, якщо ми розділимо набір даних на дві половини та застосуємо дерево рішень до обох половин, результати можуть бути різними.

Один з методів, який ми можемо використовувати для зменшення дисперсії моделей CART, відомий як бегінг, який іноді називається агрегацією початкового завантаження.[14]

Ідея бегінгу дуже проста і була запропонована Лео Брейманом у 1994 році. Наприклад, ми маємо навчальну вибірку S, на її основі ми створюємо декілька різних вибірок, які сформовані незалежно. Обучити за цими вибірками модель, а після цього результат усереднити. [15]

Для того щоб сформувати незалежні вибірки, нам допоможе метод бутстреп (bootstrap). Суть бутстрепа полягає у формуванні T нових навчальних вибірок на основі однієї вихідної. Для цього випадково вибирається $k-$ й елемент вибірки і копіюється в нову вибірку (у колишній він залишається, а не видаляється). Потім, ця операція повторюється m разів - за заданим розміром нової вибірки. Таким чином формується нова навчальна вибірка, що складається з елементів вихідної з деякими повтореннями, тому що цілком можна кілька разів випадково відібрати той самий елемент. Після формування однієї вибірки, переходять до формування наступної. Так T разів для T вибірок, які, очевидно, дещо відрізнятимуться один від одного. [15] На рисунку 1.10 показано, як може виглядати вибірка, яку сформовано методом бутстреп.

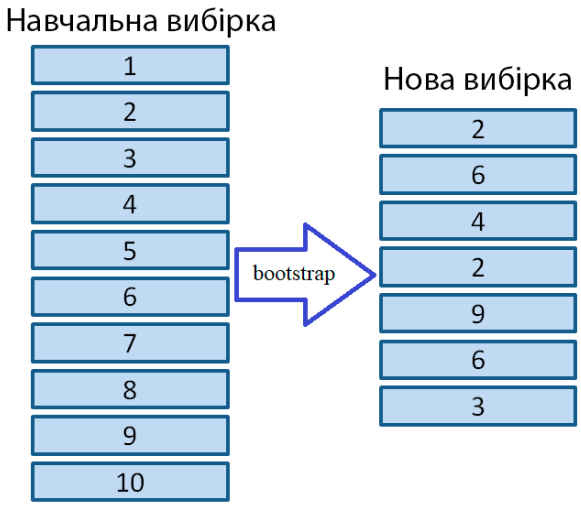

Рисунок 1.10. Вибірка сформована методом бутстреп.

Таким чином отримаємо T різних моделей, які сформують ансамбль.

Ансамблеве навчання — техніка машинного навчання, що використовує кілька навчених алгоритмів з метою отримання кращої ефективності у прогнозуванні, ніж можна було б отримати від кожного алгоритму окремо.

Ансамбль – поєднання одразу кількох алгоритмів, які навчаються одночасно та виправляють помилки один одного.

Як не дивно, схильність дерев до перенавчання відіграє позитивну роль. Вони, по-перше, виходять досить різноманітними і, по-друге, описують різні результати для вхідних векторів. Потім, при усередненні результатів, ефект перенавчання природно нівелюється (зменшується) і підсумкове вихідне значення виявляється досить точним і стійким до окремих викидів. У ряді завдань точність виявляється вищою за всі інші підходи машинного навчання. Саме тому бутстреп швидко завоював свою популярність. [15]

Після навчання T дерев на наших вибірках, ми можемо побудувати вихідний класифікатор методом голосування:

$$a(x) = sign\left(\frac{1}{T}\sum_{i=1}^{T} a_i(x)\right) \quad (1.17)$$

Щоб вирішальні дерева виходили ще різноманітнішими і формували менш залежні відповіді, пропонується під час навчання у кожній проміжній вершині випадковим чином відбирати певну кількість ознак ($m < n$, де n загальна кількість ознак) і серед них відбирати кращі для розгалуження. Набори таких дерев називають випадковим лісом (random forest). Зазвичай значення m розраховують за формулою:

$$m = \sqrt{n} \quad (1.18)$$

Алгоритм побудови випадкового лісу з N дерев наступний:[25]

Для кожного i з $n = 1, .., n$

1. Згенерувати вибірку X_i за допомогою бутстрепу;

2. Побудувати вирішуюче дерево b_i за вибіркою X_i:

- за обраним критерієм ми вибираємо кращу ознаку, робимо розбиття у дереві по ньому і так до вичерпання вибірки;
- дерево будується до тієї пори, поки у кожному листі не більше n_{min} об'єктів або поки не досягнемо певної висоти дерева;
- при кожному розбитті спочатку вибирається p випадкових ознак з n вихідних, і оптимальний поділ вибірки шукається тільки серед них.

Формування вирішального випадкового лісу відбувається за формулою 1.17, тобто за голосуванням.[25] Наприклад, повернемося до нашого прикладу з бейсболістами. На рисунку 1.11 зображемо початковий вигляд дерева для задачі бейсболіста.

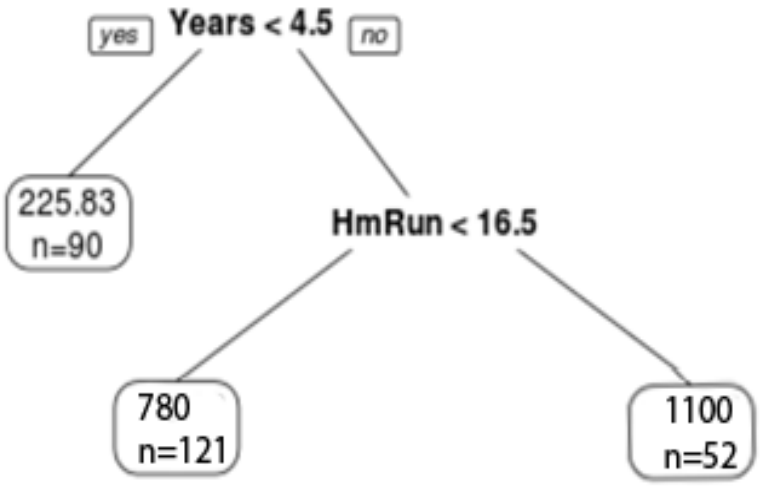

Рисунок 1.11. Початкове дерево прогнозування для задачі бейсболіста

Розраховуючи показник суми квадратів залишків (RSS) для нашої, ми отримали, наприклад, 250. Проте після виконання бутстрепу, ми отримали інформацію, що показник HmRun можна поділити ще на дві гілки і зменшити показник RSS для моделі на 10 одиниць. Таким чином ми отримали нове дерево, яке зображено на рисунку 1.12.

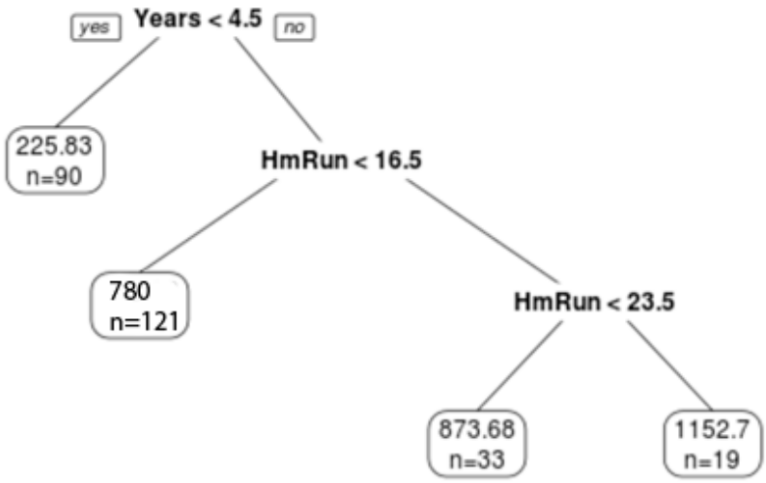

Рисунок 1.12. Модель прогнозування після процедури бегінгу.

Як можна побачити, утворився поділ на дві гілки, таким чином голосують по всім деревам, які були розраховані. Якщо значення помилки зменшується, тоді можна додавати вузол у модель.

Тепер перейдемо до іншого алгоритму машинного навчання дерев рішень – це бустинг. Бустинговий алгоритм також використовує декілька дерев, але навідміну від бегінгу, де кожному дереву одночасно дається своя вибірка і їх результати усереднюються, кожне дерево повинно послідовно виконувати свою роботу на одній вибірці, і виправляти помилки минулого - таким чином бустити результат. Існує два найпопулярніших алгоритмів бустингу, які буде описано нижче [16]

Першим методом є адаптивний бустинг (AdaBoost). AdaBoost означає "Adaptive Boosting" або адаптивний бустинг. Він перетворює слабкі навчальні алгоритми на сильні рішення задачі класифікації.

Остаточне рівняння для класифікації може виглядати так:

$$F(x) = sign\left(\sum_{m=1}^{M} \theta_m f_m(x)\right), \quad (1.19)$$

Тут $f_m -$ це m слабкий класифікатор, а $\theta_m -$ відповідна вага цього класифікатора.

AdaBoost можна використовувати для підвищення продуктивності алгоритмів машинного навчання. Він найкраще працює зі слабкими навчальними алгоритмами, тому такі моделі можуть досягти точності набагато вище за випадкову при вирішенні задачі класифікації. Найбільш поширеними алгоритмами, які використовуються з AdaBoost, є однорівневі дерева рішень. Слабкий навчальний алгоритм – це класифікатор чи алгоритм передбачення, який погано працює в плані точності. Крім цього, можна сказати, що слабкі класифікатори легко обчислюються, тому можна поєднувати багато сутностей алгоритму, щоб створити сильніший класифікатор за допомогою бустингу. [26]

Якщо у нас є набір даних, у якому n – кількість точок та

$$x_j \in R^d, y_i \in \{-1; 1\},$$

Де -1 буде від'ємним класом, а 1 додатнім. Тоді вага кожної точки буде ініціалізовано наступним чином:

$$w(x_i, y_i) = \frac{1}{n}, i = 1, \dots, n. \quad (1.20)$$

Кожне m у наступному виразі ми змінюватимемося від 1 до M.

Для початку потрібно вибрати слабкий класифікатор із найменшою зваженою помилкою класифікації, застосувавши класифікатор до набору даних.

$$\epsilon_m = E_{w_m}[1_{y \neq f(x)}] \quad (1.21)$$

Потім розрахуємо вагу $m -$ го слабкого класифікатора, як показано нижче:

$$\theta_m = \frac{1}{2} \ln\left(\frac{1-\epsilon_m}{\epsilon_m}\right) \quad (1.22)$$

Вага позитивна для будь-якого класифікатора з точністю вище 50%. Чим більша вага, тим точніше класифікатор. Вага стає негативною, коли точність падає нижче 50%. Передбачення можна поєднувати, інвертуючи знак. Таким чином класифікатор з точністю 40% можна перетворити на класифікатор з точністю 60%. Так класифікатор вноситиме внесок у підсумкове передбачення, навіть якщо він працював гірше, ніж випадкове прогнозування. Однак остаточний результат не зміниться під впливом класифікатора, точність якого дорівнює 50%.

Експонента в чисельнику завжди буде більше 1 у разі неправильної класифікації із класифікатора з позитивною вагою. Після ітерації вага неправильно класифікованих об'єктів зросте. Класифікатори з негативною вагою поведуться аналогічним чином. Тут різниця в інверсії знака: правильна класифікація стане неправильною. Остаточний прогноз можна розрахувати шляхом урахування вкладу кожного класифікатора та обчислення суми їх виважених прогнозів. [26]

Вага для кожної точки оновлюватиметься таким чином:

$$w_{m+1}(x_i, y_i) = \frac{w_m(x_i,y_i)\cdot\exp[-\theta_m\cdot y_i\cdot f_m(x_i)]}{Z_m} \qquad (1.23)$$

Тут Z_m – нормалізуючий параметр, він потрібен, щоб впевнитися, що сума всіх коефіцієнтів ваги дорівнює 1.

Поєднати усі результати класифікаторів в один дозволяє формула 1.19.

На рисунку 1.13 можна подивитися як AdaBoost навчається класифікації. Алгоритм поділяє площину на дві частини, він збільшує розмір точок, які були неправильно класифіковані (не потрапили в область виділення). [17]

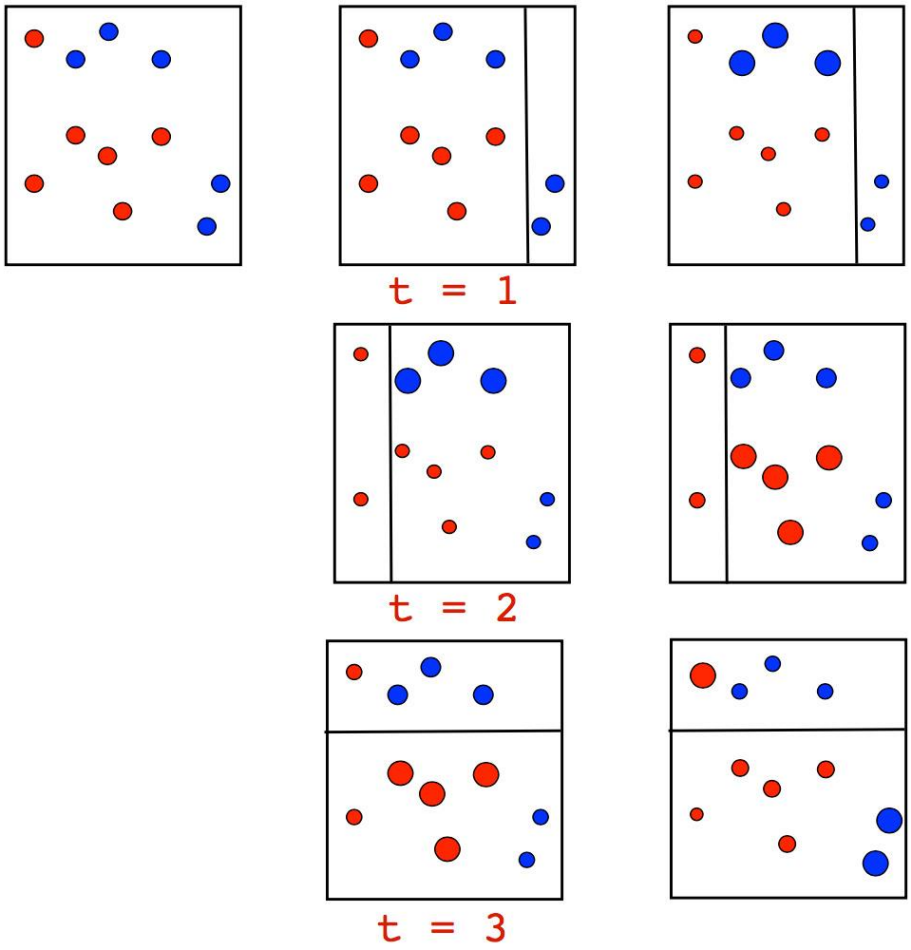

Рисунок 1.13. Поітераційне ділення площини на дві частини, з виділенням точок, які не потрапили в свою область.

У кінці ці всі ітерації об’єдналися з урахуванням коефіцієнтів ваг, які були розраховані відповідно до помилок моделі.

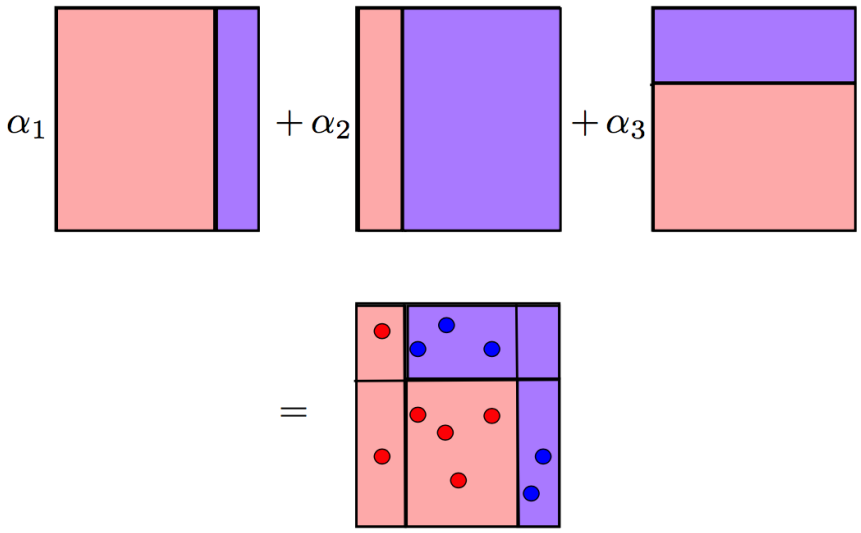

Рисунок 1.14. Результат об’єднання площин.

Алгоритм можливо застосовувати в завданнях навчання складніше, ніж двійкова класифікація, і є досить універсальним, оскільки його можна використовувати з числовими або текстовими даними.

Також AdaBoost вразливий до рівномірно розподіленого шуму. Слабкі класифікатори можуть призвести до поганих результатів та перенавчання.[26]

Градієнтний бустінг є в деякому сенсі продовженням Адаптивного бустінгу. Математична складова, на перший погляд, змінилась не так сильно: ми все також додаємо (бустимо) слабкі алгоритми (дерева), нарощуючи нашу цільову функцію поступовими поліпшеннями тих ділянок даних, де попередні моделі "не допрацювали". Але при побудові наступної простої моделі, вона будується не просто на переважених спостереженнях, а так, щоб якнайкраще наближати загальний градієнт цільової функції. [17]

Ми будемо вирішувати завдання відновлення функції у загальному контексті навчання з учителем. Ми будемо мати набір пар ознак x_i і цільових змінних y_i, $\{x_i, y_i\}_{i=1,\dots,n}$. На цьому ми відновлюватимемо залежність виду $y = f(x)$. Відновлюватимемо наближення $\hat{f}(x)$, а для розуміння, яке наближення краще, у нас також буде функція втрат $L(y, f)$, яку ми будемо мінімізувати. [17]

$$y = \hat{f}(x) \tag{1.24}$$

$$\hat{f}(x) = \arg\min_{f(x)} E_{x,y}[L(y, f(x))] \tag{1.25}$$

Кількість функцій $f(x)$ у світі не просто багато — саме їхній функціональний простір нескінченномірний. Тому, щоб розв'язати задачу в машинному навчанні, зазвичай обмежують простір пошуку яким-небудь конкретним параметризованим сімейством функцій $f(x, \theta), \theta \in R^d$. Це спрощує завдання, тому що вона зводиться до оптимізації значень параметрів: [17]

$$\hat{f}(x) = f\left(x, \hat{\theta}\right), \tag{1.26}$$

$$\hat{\theta} = \arg\min_{\theta} E_{x,y}[L(y, f(x, \theta))] \tag{1.27}$$

Отримання оптимальних $\hat{\theta}$ в один рядок відбуваються досить рідко, тому параметри зазвичай наближають ітеративно. Спочатку нам треба виписати емпіричну функцію втрат $L_\theta(\hat{\theta})$, яка показує, наскільки добре ми їх оцінили за наявними у нас даними. Також випишемо наше наближення $\hat{\theta}$ за M ітерацій у вигляді суми: [17]

$$\hat{\theta} = \sum_{i=1}^{M} \widehat{\theta_i}, \tag{1.28}$$

$$L_\theta\left(\hat{\theta}\right) = \sum_{i=1}^{N} L(y_i, f\left(x, \hat{\theta}\right)) \tag{1.29}$$

Залишилося тільки взяти ітеративний алгоритм, за яким ми будемо мінімізувати значення $L_\theta\left(\hat{\theta}\right)$. Найпростіший варіант, що часто використовується — це градієнтний спуск. Для цього потрібно розрахувати градієнт $\nabla L_\theta\left(\hat{\theta}\right)$ і додавати наші ітеративні оцінки $\hat{\theta}$ (зі знаком мінус – тому що нам потрібно зменшувати помилку, бо градієнт показує напрямок найшвидшого росту). [17]

Тепер треба ініціалізувати наше перше наближення $\widehat{\theta_0}$ і обрати кількість ітерацій M, скільки ми цю процедуру будемо продовжувати. [17]

Сформуємо загальний вигляд постановки рішення:

1. Ініціалізувати початкове наближення параметрів $\widehat{\theta_0}$.
2. Для кожної ітерації t=1,…,M повторювати:
    1. Розрахувати градієнт функції втрат $\nabla L_\theta\left(\hat{\theta}\right)$ при теперішньому наближені параметрів $\hat{\theta}$.

$$\nabla L_\theta\left(\hat{\theta}\right) = \left[\frac{\partial L\left(y, f(x, \theta)\right)}{\partial \theta}\right]_{\theta=\widehat{\theta}} \tag{1.30}$$

    2. Задати теперішнє ітеративне наближення $\widehat{\theta_t}$ на основі розрахованого градієнта

$$\widehat{\theta_t} \leftarrow -\nabla L_\theta(\hat{\theta}) \tag{1.31}$$

3. Оновити наближення параметрів $\hat{\theta}$:

$$\hat{\theta} = \sum_{i=0}^{M} \hat{\theta}_i \tag{1.32}$$

На рисунку 1.15 показано ітераційне наближення розв'язку до оптимума.

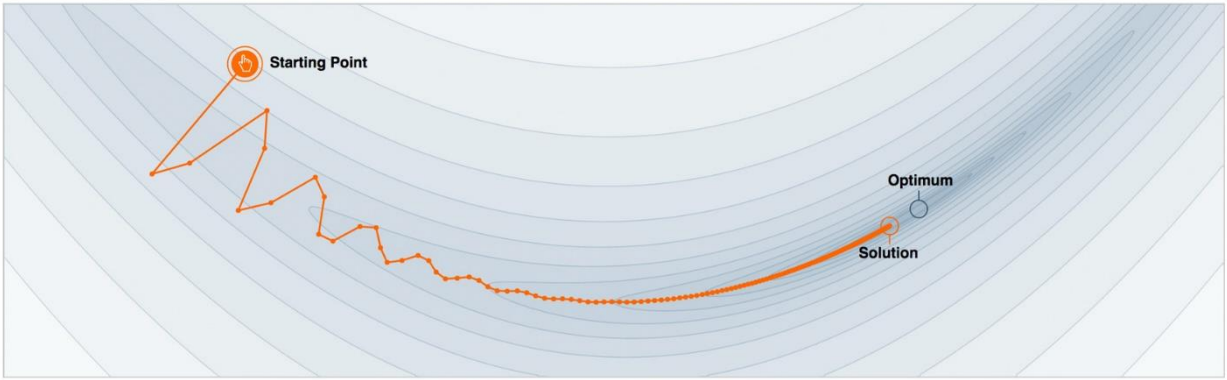

Рисунок 1.15. Наближення результату до оптимума.

Ми можемо спростити наш метод тим, що будемо шукати не параметри однієї великої моделі (як, наприклад, нейронна мережа), а будемо складати велику кількість функцій помножених на коефіцієнт. Таким чином початкове наближення $\hat{f}_0(x)$:

$$\hat{f}_0(x) = \sum_{i=0}^{M} \hat{f}_i(x) \tag{1.33}$$

Також, як і раніше, обмежимо наш пошук сімейством функцій $\hat{f}_i(x) = h(x, \theta)$. Але тепер треба враховувати коефіцієнт, який буде відповідати за вплив на загальну суму функції, позначимо його як $p \in R$. Для кроку t задача буде виглядати наступним чином: [17]

$$\hat{f}(x) = \sum_{i=0}^{t-1} \hat{f}_i(x) \tag{1.34}$$

$$(p_t, \theta_t) = \arg\min_{p,\theta} E_{x,y}\left[L\left(y, \hat{f}(x) + p \cdot h(x, \theta)\right)\right], \tag{1.35}$$

$$\hat{f}_t(x) = p_t \cdot h(x, \theta_t) \tag{1.36}$$

Наші завдання загалом виглядають так, ніби ми можемо просто так брати і навчати які завгодно моделі $h(x, \theta)$ щодо будь-яких функцій втрат

$L(y, f(x, \theta))$. На практиці це вкрай складно, тому було придумано простий спосіб звести завдання до чогось вирішуваного.

Знаючи вираз градієнта функції втрат, ми можемо визначити його значення на наших даних. Ми можемо навчати моделі так, щоб наші прогнози були найбільш скорельованими із цим градієнтом (зі знаком мінус). Тобто вирішуватимемо завдання МНК-регресії (метод найменших квадратів), намагаючись виправляти значення наступної функції значеннями залишків МНК-регресії. Ми будемо мінімізовувати квадрат різниці r (антіградієнт) з прогнозами на t кроці. Для кроку t задача буде виглядати наступним чином: [17]

$$\hat{f}(x) = \sum_{i=0}^{t-1} \hat{f}_i(x), \tag{1.37}$$

$$r_{i,t} = -\left[\frac{\partial L(y_i, f(x_i))}{\partial f(x_i)}\right]_{f(x)=\hat{f}(x)}, i = 1, .., n, \tag{1.38}$$

$$\theta_t = \arg\min_{\theta} \sum_{i=1}^{n} \left(r_{i,t} - h(x, \theta)\right)^2, \tag{1.39}$$

$$p_t = \arg\min_{p} \sum_{i=1}^{n} L\left(y, \hat{f}(x) + p \cdot h(x_i, \theta_t)\right) \tag{1.40}$$

На рисунку 1.16 показано як змінюється прогнозування відносно залишків, зроблених МНК-регресією. Зліва (синім кольором) показано цільову функцію, яку потрібно оптимізувати, праворуч (зеленим кольором) залишки функції, за яким виконується навчання моделі.

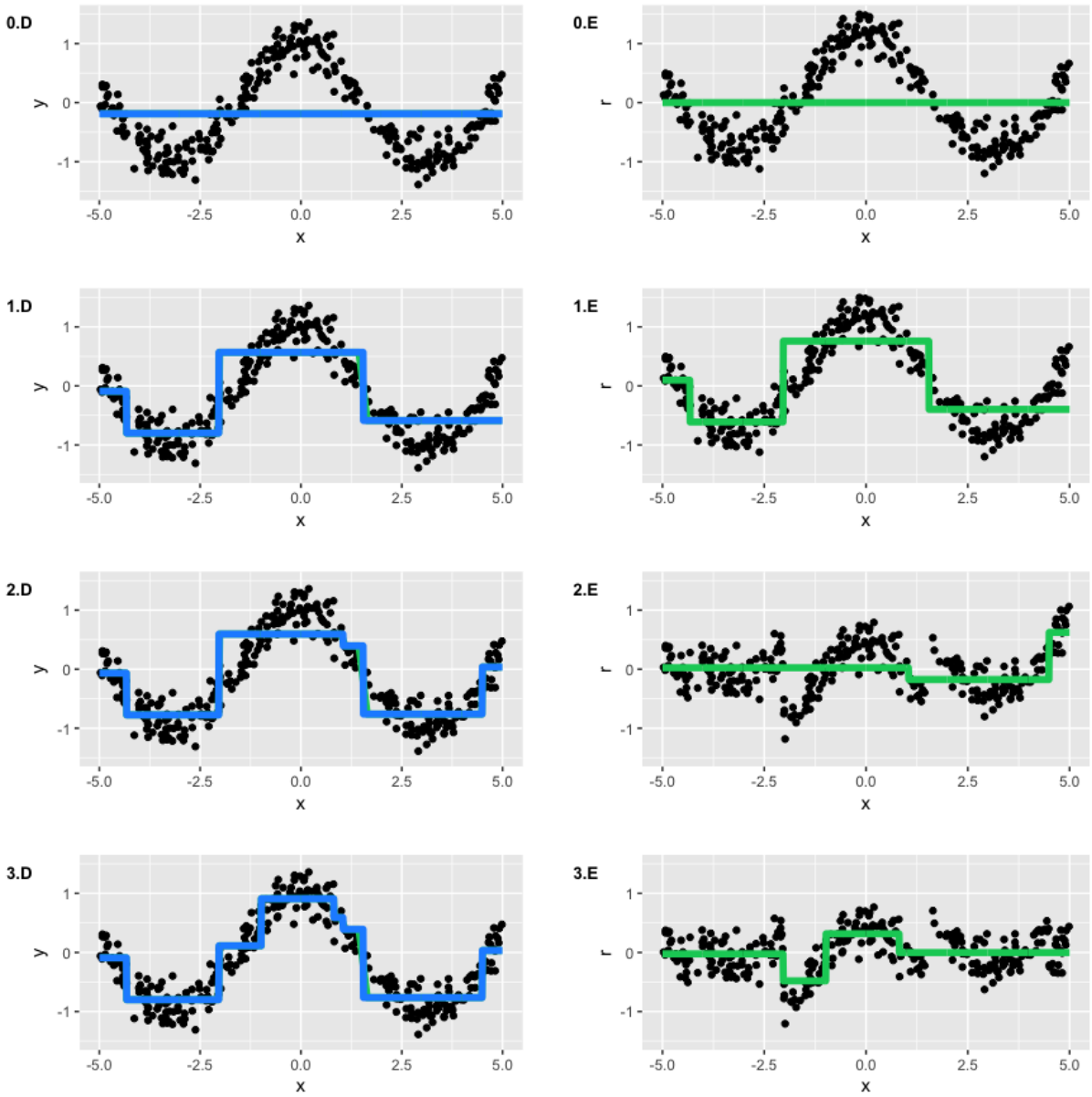

Рисунок 1.16. Демонстрація роботи алгоритму відносно залишків.

Єдиний момент, що залишився поза увагою, — початкове наближення $\hat{f}_0(x)$. Для простоти як ініціалізацію використовують просто константне значення γ. Спираючись на вищевикладені записи, сформулюємо загальний вигляд GBM (Gradient Boosting Method), який запропонував Джеромі Фрідман (Jerome Friedman) у 1999 році: [17]

1. Ініціалізувати GBM константним значенням $\hat{f}(x) = \hat{f}_0, \hat{f}_0 = \gamma, \gamma \in R$.

$$\hat{f}_0 = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$$

2. Для кожної ітерації t=1,…,M повторювати:

1. Розрахувати залишки r_t за формулою 1.38 або іншою формулою залишків.

2. Побудувати новий базовий алгоритм $h_t(x)$ як регресію на залишках r_t, використовуючи формулу 1.39.

3. Знайти оптимальний коефіцієнт p_t при $h_t(x)$ відносно вихідної функції втрат, використовуючи формулу 1.40.

4. Зберегти $\hat{f}_t(x) = p_t \cdot h_t(x)$.

5. Оновити поточне наближення $\hat{f}(x)$.

$$\hat{f}(x) \leftarrow \hat{f}(x) + \hat{f}_t(x) = \sum_{i=0}^{t} \hat{f}_i(x)$$

3. Скомпонувати підсумкову модель GBM

$$\hat{f}(x) = \sum_{i=0}^{M} \hat{f}_i(x)$$

Для різних задач роботи дерева використовують різну формулу залишків. В задачі регресії одну, в класифікації іншу. Наведемо список формул для кожного класу задач:

Для регресії:

1. $L(y, f) = (y - f)^2$, вона же $L_2 - loss$ або Gaussian loss. Класичне умовне середнє, варіант, який найчастіше використовують, якщо нема додаткової інформації.

2. $L(y, f) = |y - f|$, вона же $L_1 - loss$ або Laplacian loss. Визначає умовну медіану. Медіана більш стійка до викидів, тому в деяких завданнях ця функція втрат краща, оскільки вона не так сильно штрафує великі відхилення.

3. $L(y,f) = \begin{cases} (1-\alpha)\cdot|y-f|,\ if\ y-f \le 0 \\ \alpha\cdot|y-f|,\ if\ y-f > 0 \end{cases}$, $\alpha \in (0,1)$ вона же $\mathrm{L}_q loss$. Якби ми, припустимо, захотіли не умовну медіану з $L_1 - loss$, а умовний 75%-квантиль, ми скористалися б цим варіантом з $\alpha = 0{,}75$. Можна бачити, що ця функція асиметрична і більше штрафує спостереження, які знаходяться у потрібному нам квантилі.

Функція втрат для класифікації:

1. $L(y,f) = \log(1+\exp(-2yf))$, вона же $logistic\ loss$. Це найстандартніша і найчастіше використовувана функція втрат у бінарної класифікації.

2. $L(y,f) = \exp(-yf)$, вона же Adaboost loss. Так вийшло, що класичний Adaboost алгоритм еквівалентний GBM із цією функцією втрат. Концептуально ця функція втрат дуже схожа на Logistic loss, але має жорсткіший експоненційний штраф на помилки класифікації та використовується рідше.

Налаштування алгоритму не обмежується лише зміною функцію втрат. Також можна додавати коефіцієнт ваги для певної групи значень, наприклад, для навчання алгоритму під конкретну групу клієнтів. Наведемо приклад наступної функції втрат.[17]

$$L_w(y,f) = w \cdot L(y,f),$$

$$r_{i,t} = -w_i \left[\frac{\partial L\big(y_i, f(x_i)\big)}{\partial f(x_i)}\right]_{f(x)=\hat{f}(x)}, i = 1,..,n,$$

$$w = \begin{cases} 0.1, & x \le 0 \\ 0.1 + |\cos x|, & x > 0 \end{cases}$$

Графік такої функції втрат буде наступним (див. рисунок 1.17):

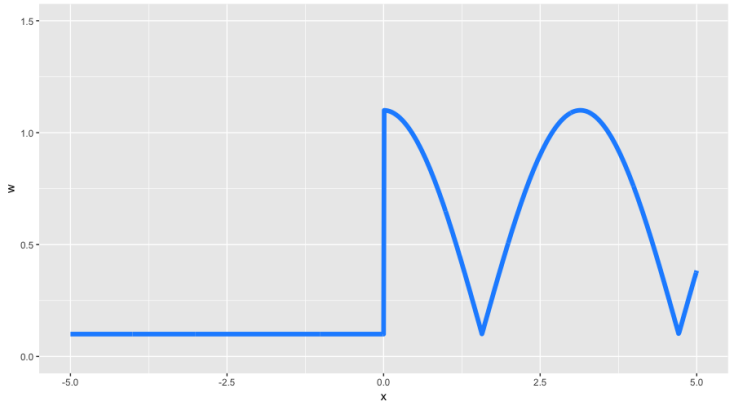

Рисунок 1.17. Функція втрат

Як видно, вага налаштована таким чином, що від’ємні значення майже ігноруються, а додатні приймають значення косинуса.

На рисунку 1.18 показано процес навчання моделі. Як видно, деталізується лише додатня частина координат.

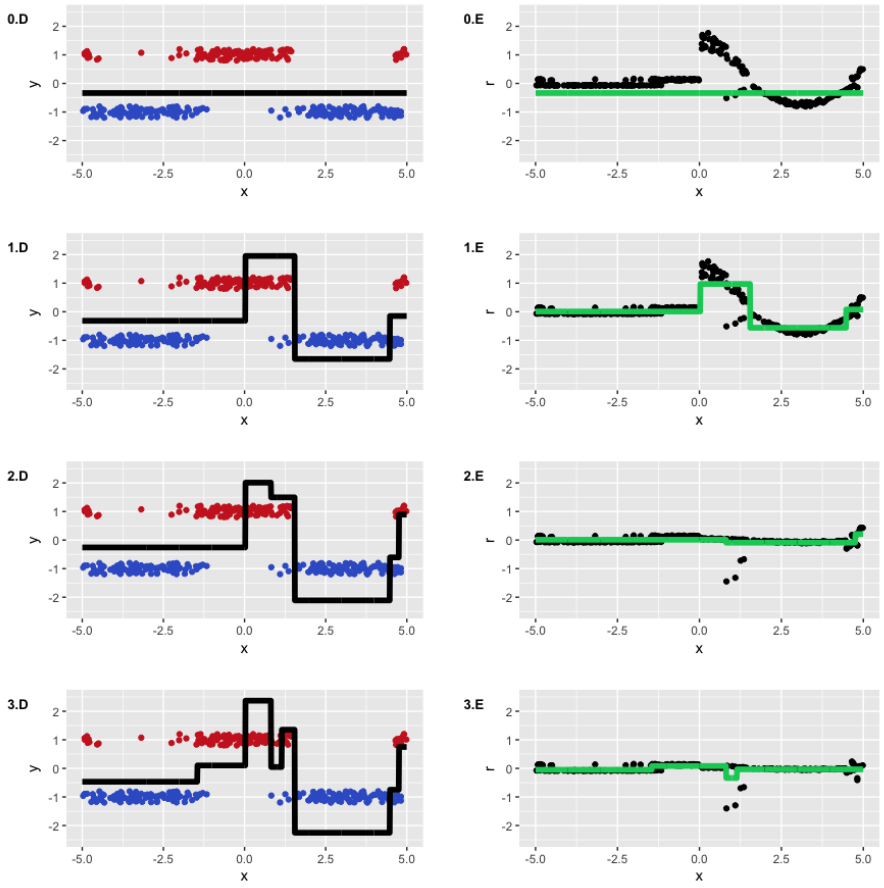

Рисунок 1.18. Процес навчання моделі з коефіцієнтами ваги

Gradient Boosting Machine це не просто конкретний алгоритм, а загальна методологія того як будувати ансамблі моделей. Методологія досить гнучка і розширювана: можна навчати велику кількість моделей з урахуванням різних функцій втрат і при цьому ще й додавати до них різні вагові функції. Як показує і практика, і досвід змагань з машинного навчання, в стандартних завданнях (усі крім картинок і аудіо, а також сильно розріджених даних), GBM дуже часто є найефективнішим алгоритмом. [17] GBM станом на зараз має декілька популярних бібліотек для машинного навчання моделей, серед них такі як: XGBoost, LightGBM.

4. Метод k-ближчих сусідів

Метод найближчих сусідів (k-Nearest Neighbor, далі - kNN) — найпростіший метричний класифікатор, що ґрунтується на оцінюванні подібності об'єктів. Об'єкт, що класифікується, відноситься до того класу, якому належать найближчі до нього об'єкти навчальної вибірки. [18]

Цей метод є, мабуть, найпростішим алгоритмом класифікації. Суть полягає в тому, що об'єкт відноситься до того класу, якому належить більшість з його сусідів - k найближчих до нього об'єктів навчальної вибірки X_i.

Алгоритм kNN складається із трьох послідовних етапів:

1. Обчислити відстань від цільового об'єкта (який необхідно класифікувати) до кожного з об'єктів навчальної вибірки (вже помічених будь-яким класом);

За замовчуванням алгоритм використовує метрику Мінковського, яка у разі ступеня p = 2 звертається у всьому відому зі шкільної геометрії Евклідову метрику - відстань між двома точками у просторі:[19]

$$dist = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2 + \cdots + (x_n - y_n)^2} \quad (2.41)$$

На рисунку 1.19 показано процес розрахунку відстаней.

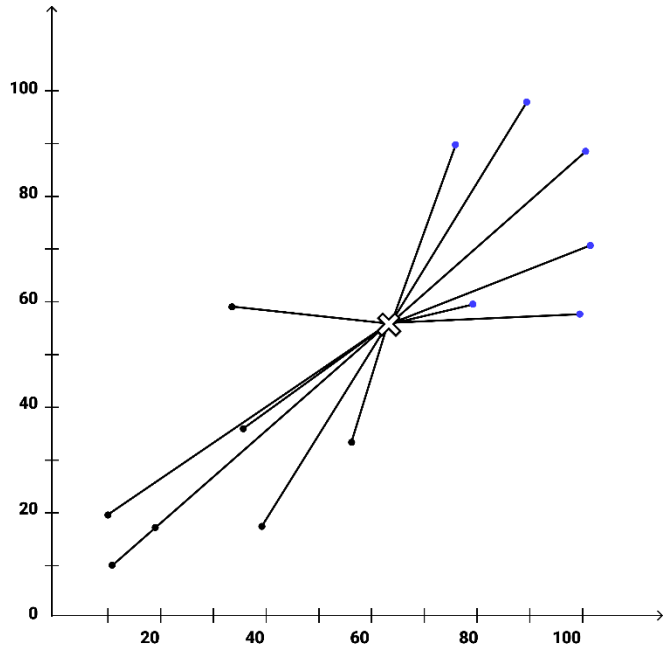

Рисунок 1.19. Розрахунок відстані від заданої до точок з вибірки.

2. Відібрати k об'єктів навчальної вибірки, відстані яких мінімальні (на першому етапі k вибирається довільно, потім ітеративно підбирається краще значення k з урахуванням точності отриманих прогнозів при кожному з обраних k).

На рисунку 1.20 показано як виглядає область для визначення $k-$ ближчих точок.

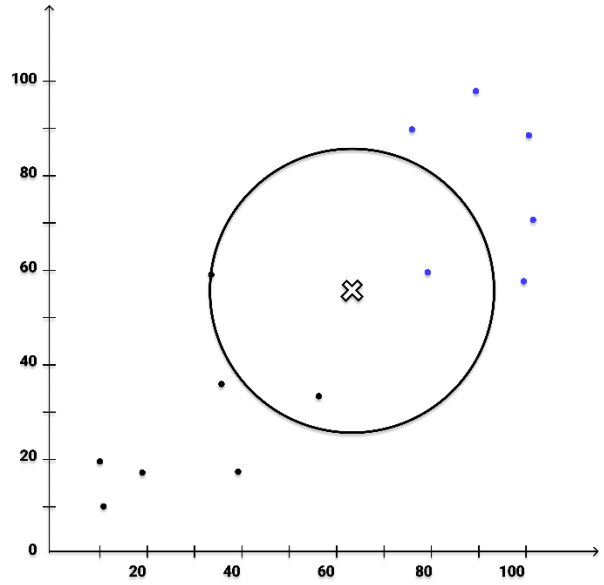

Рисунок 1.20. Область для порівняння точок.

3. Отримати клас об'єкта на основі сусідів, які найчастіше зустрічаються (це може бути число або назва класу в залежності від того, як були позначені класи - наприклад, у прикладі з безпілотними автомобілями це може бути "людина" або "бетонний блок"). На рисунку 1.21 зображено процес класифікації. [19]

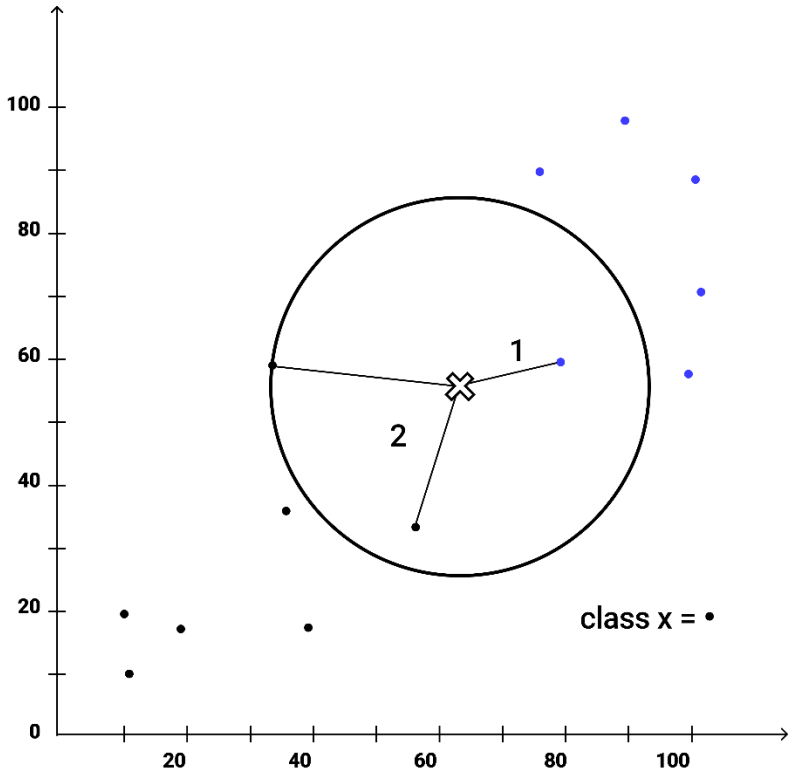

Рисунок 1.21. Класифікація точки.

Як видно на рисунку 1.21, представників класу 2 більше, ніж 1, тому точка буде мати клас 2.

В цілому, незважаючи на те, що алгоритм достатньо простий для розуміння та реалізації, він знайшов широке застосування в таких галузях, як рекомендаційні системи, семантичний пошук і виявленя аномалії. Однак для його застосування ми повинні мати можливість зберегти весь навчальний набір у пам’яті, а виконання класифікацій може бути обчислювально дорогим, оскільки алгоритм аналізує всі точки даних для кожної класифікації. З цієї причини kNN краще всього застосувати до невеликих наборів даних, які не мають величезного набору признаків. [19]

5. Нейронна мережа

Нейромережа - це сукупність пов'язаних нейронних блоків, що виконують обробку інформації. Вони базуються на алгоритмах глибокого навчання і сьогодні широко використовуються у вирішенні багатьох завдань: від класифікації даних, обробки зображень до розпізнавання мови. Сьогодні існує велика різноманітність НМ (нейронних мережей), включаючи згорткові та рекурентні структури. [20]

На рисунку 1.22 зображено концептуальну модель нейронної мережі.

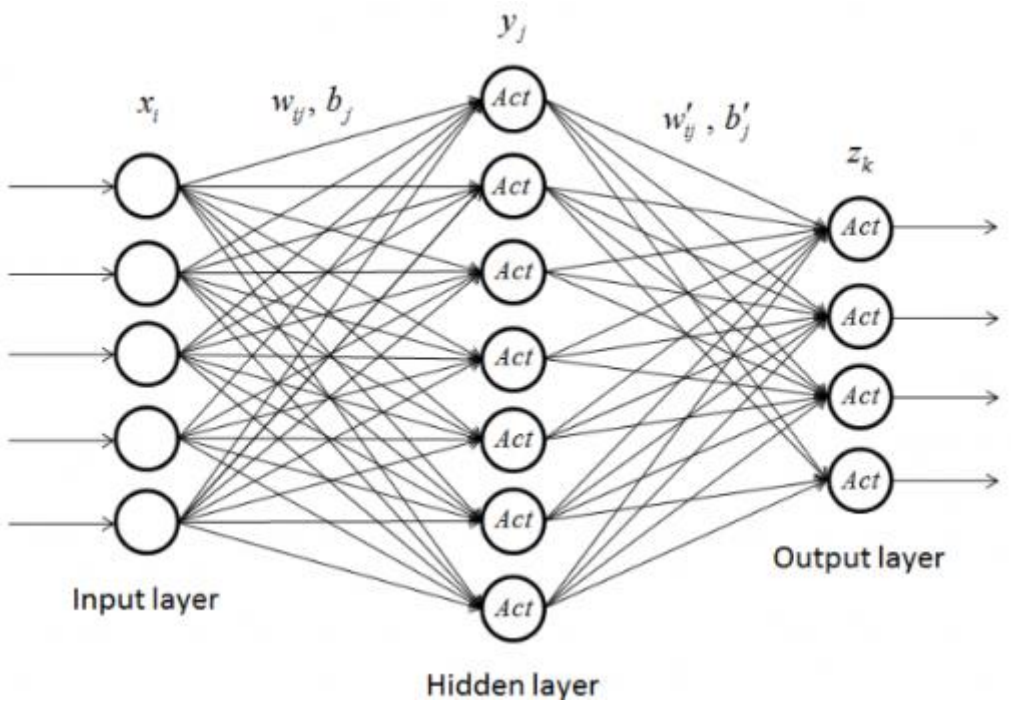

Рисунок 1.22. Концептуальна модель нейронної мережі.

Алгоритм роботи нейронної мережі буде записано нижче:

1. На вхідний шар надходять дані.

2. Синапси передають дані на наступний шар. Кожному синапсу надано певний коефіцієнт ваги і в кожного наступного нейрона може бути кілька вхідних синапсів.

3. Дані, що передаються наступному нейрону - це сума всіх даних у НМ, помножених на відповідні коефіцієнти ваги.

4. Отримані значення підставляються у функцію активації, що призводить до формування вихідних даних.

5. Передача даних буде продовжуватися, доки вона не досягне кінцевого виходу.

1. Розрахунок помилки.

Однією з особливостей нейронних мереж є здатність до навчання та вдосконалення. Для цього використовується середньоквадратична помилка втрат.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_{true} - y_{pred}\right)^2 \quad (1.42)$$

де $n -$ кількість прогнозів, $y_{true} -$ істинне значення, $y_{pred} -$ прогнозне значення.

Загальний квадрат помилки можна одержати за допомогою функції втрат. Варто розглядати втрати як функцію коефіцієнтів ваги і чим кращий прогноз, тим менші втрати. Таким чином, мета полягає у навчанні мережі для мінімізації втрат. Тепер у вас є можливість змінити коефіцієнти мережі, щоб вплинути на прогнози. Позначте кожну вагу в мережі, а потім запишіть втрати як багатовимірну функцію. [20]

У свою чергу стохастичний градієнтний спуск показує, як змінити ваги, щоб мінімізувати втрати. Це представлено у вигляді рівняння:

$$w_1 \leftarrow w_1 - \eta \frac{\partial L}{\partial w_1} \quad (1.43)$$

де $w_1 -$ коефіцієнт ваги, $\eta -$ константа, яка відповідає за швидкість навчання, $L -$ значення цільової функції

Коли значення $\frac{\partial L}{\partial w_1}$ додатнє, то w_1 буде зменшуватися, що призведе до зменшення L. Якщо значення $\frac{\partial L}{\partial w_1}$ буде від'ємне, тоді L буде зростати.

Виконання цього рівняння для кожного коефіцієнта ваги в мережі зменшить втрати та покращить мережу. У такому випадку важливо мати правильний тренувальний процес, такий як:

1. Вибір одного зразка з набору даних, щоб зробити його стохастичним градієнтним спуском, працюючи лише з одним зразком у певний час.
2. Обчислення всіх похідних втрат за вагами.
3. Використання рівняння оновлень для оновлення кожної ваги.
4. Повертаємось до кроку 1 і рухаємося вперед.
5. Коли всі описані процеси завершено, ви готові реалізувати повну НМ. Згадані кроки призводять до неухильного зниження втрат та підвищення точності. [20]

2. Функція активації є нелінійним перетворенням, яке поелементно використовується до вхідних даних. У цьому контексті передбачається, що вона додається до штучної нейронної мережі, щоб допомогти мережі вивчити складні закономірності даних. У порівнянні з моделлю, заснованою на нейронах, яка знаходиться в нашому мозку, функція активації зрештою вирішує, який наступний нейрон повинен бути запущений. [20]

Існує декілька типів функції активації:

1.Лінійна функція активації або активація ідентичності.

2.Двійкова ступінчаста функція або активація бінарного кроку.

3.Нелінійні функції активації.

Лінійна функція. Її діапазон від $-\infty$ до $+\infty$ та принцип роботи полягає в тому, що при виконанні підсумовується зважена сума вхідних даних та повертається результат. Нижче ви можете ознайомитися з графіком лінійної функції активації:

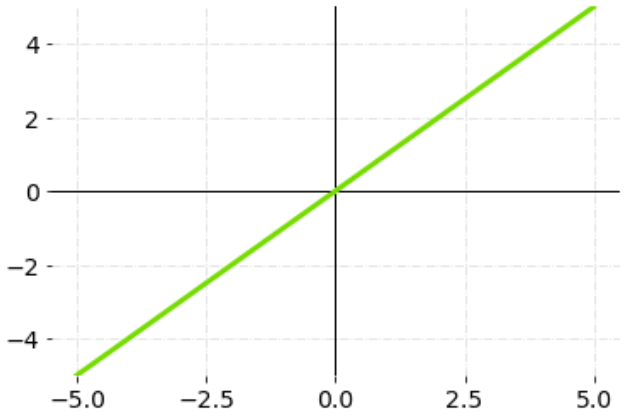

Рисунок 1.23. Лінійна функція активації

Математично це можна записати так:

$$f(x) = x \tag{1.44}$$

Це не бінарна функція і вона обмежена діапазоном активацій. Але вона має на увазі об'єднання кількох нейронів разом, щоб обчислити максимальне значення за наявності кількох активацій. Похідна цієї функції є постійною величиною, яка залежить від вхідного значення x.

Наступною функцією є бінарний крок, для виконання якого важливе граничне значення. Він дозволяє визначити, активувати нейрон чи ні. Нейрон активується у разі, якщо вхідне значення перевищує порогове. Але якщо він не активований, його вихідні дані не передаються на наступний або прихований шар. Нижче наведено графік, що ілюструє функцію активації бінарного кроку: [20]

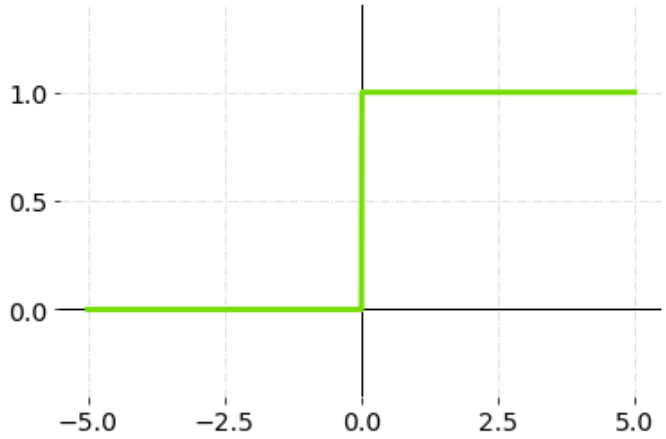

Рисунок 1.24. Бінарна функція активації.

Математично бінарну функцію активації можна представити у вигляді:

$$f(x) = \begin{cases} 0, x < 0 \\ 1, x \geq 0 \end{cases} \quad (1.45)$$

Важливо відзначити, що представлена можливість не здатна генерувати різноманітні вихідні дані з кількома значеннями. Наприклад, вона може використовуватися для класифікації у межах безлічі класів. Градієнт ступінчастої функції дорівнює нулю, що створює деякі труднощі при зворотному розповсюдженні. [20]

Нелінійна функція активації – це найпоширеніший тип, що дозволяє нейронним мережам легко пристосовуватися до різних даних та розділяти вихідні значення. Більш того, нелінійні функції активації дозволяють додавати кілька шарів нейронів, оскільки вихідні дані стають нелінійною комбінацією вхідних даних, що проходять через різні шари.

3. Функції втрат у глибокому навчанні використовуються, щоб виміряти те, наскільки добре працює модель нейромережі. Справа в тому, що всередині НМ відбуваються 2 можливі математичні операції - пряме та зворотне поширення з градієнтним спуском. Процес прямого поширення є

обчислювальною процедурою, спрямованої прогнозування вихідних даних для заданого вхідного вектора x. З іншого боку, зворотне поширення і градієнтний спуск є методами, що описують процес поліпшення ваг і зміщень в нейронній мережі, з метою досягнення більш точних прогнозів.[20]

Наприклад, є вектор x, для якого нейронна мережа передбачає вихід, який називається вектором передбачення y.

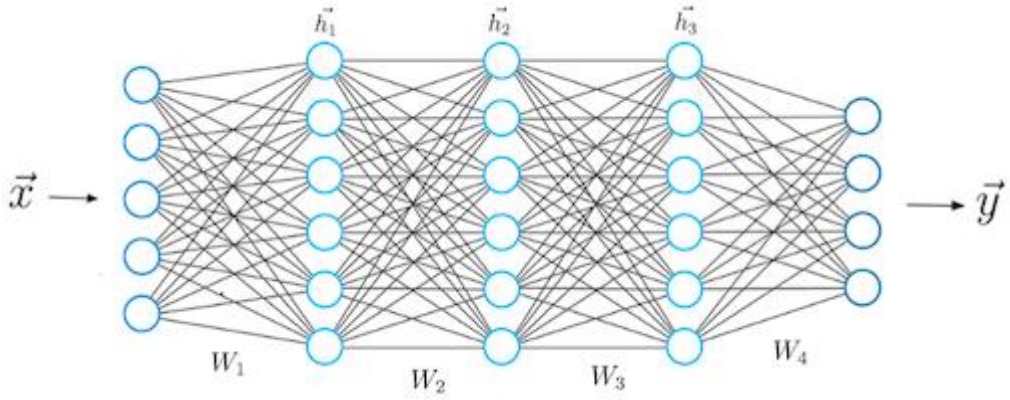

Рисунок 1.25. Концептуальна модель нейронної мережі.

Поліпшення функції втрат у нейронних мережах є ключовим аспектом досягнення високої ефективності під час виконання різних завдань, наприклад, регресія чи класифікація. Оцінка якості моделі нейронної мережі полягає в тому, як точно вона виконує поставлене завдання. У процесі зворотного поширення необхідно прагнути мінімізації значення функції втрат, оскільки це дозволяє поліпшити продуктивність та ефективність нейронної мережі. Результати зворотного розповсюдження будуть оптимальними за мінімального значення функції втрат. [20]

Одним із заключних етапів є покращення та оптимізація нейромережі, і він полягає на налаштуванні параметрів для підвищення продуктивності, а саме регуляризація та методи боротьби з перенавчанням. [20]

Перенавчання вважається однією з проблем, з якою стикаються нейромережі. Але завдяки регуляризації можна зменшити наслідки від перенавчання моделі. Варто відзначити, що є кілька методів регуляризації: L1 і L2-регуляризації, і Dropout.

L1-регуляризація передбачає додавання штрафу до функції втрат, який залежить від суми абсолютних значень ваги моделі. Цей підхід призводить до зменшення ваги, що дозволяє видалити несуттєві функції з моделі.

L2-регуляризація ґрунтується на додаванні штрафу до функції втрат, який залежить від суми квадратів ваги моделі. Це зменшує ваги, що призводить до зменшення чутливості моделі до шуму та викидів.

Dropout - це метод, який випадково відключає деякі нейрони під час навчання. Це дозволяє уникнути перенавчання та покращити узагальнюючу здатність моделі.

Наступним способом є підбір гіперпараметрів та використання сітки пошуку (Grid Search).

Гіперпараметри слід встановити ще до того, як почнеться навчання нейромережі. Це дозволить визначити архітектуру моделі, параметри оптимізації та інші налаштування. Але варто розуміти, що підбір гіперпараметрів не таке просте завдання, і тому доводиться вдаватися до використання сітки пошуку. Сітка пошуку, у свою чергу, робить перебір усіх можливих комбінацій, щоб підібрати найоптимальніші параметри. У кожній комбінації модель навчається та оцінюється на перевірочному наборі даних. Потім вибирає ту комбінацію, яка гарантує найкращий результат. [20]

1.5 Word Embedding

1.5.1 Основні відомості

Основна одиниця, з якою ми працюємо під час розв’язання задачі NLP, це слово. Або формальніше «токен». Ми використовуємо цей термін, тому що не дуже зрозуміло, що таке «2128506» - це слово чи ні? Відповідь не

очевидна. Токен зазвичай відділений від інших токенів пробілами або розділовими знаками, і тут з’являються складнощі зі знанням контексту. Є різні підходи, але у 95% випадках таким контекстом, що розглядається при роботі моделі, виступає речення, що містить цей токен. [21]

Багато завдань взагалі вирішуються лише на рівні речень. Наприклад, машинний переклад. Найчастіше ми просто перекладаємо одне речення і ніяк не використовуємо контекст ширшого рівня. Є завдання, де цього недостатньо, наприклад, діалогові системи. Тут важливо пам'ятати, про що запитували систему раніше, щоб вона могла відповісти на запитання. Проте речення — також основна одиниця, з якою ми працюємо.

Тому перші два кроки, які виконуються практично для вирішення будь-яких завдань – це сегментація та токенізація.

Сегментація – розподіл тексту на абзаци, абзаців на речення. Іншими словами розбиття складної структури на декілька менш складних.

Токенізація – перший крок при обробці тексту. Він полягає в розбитті довгих строк тексту на короткі: абзаци ділять на речення, речення на слова.

Далі слід обчислити ознаки кожного токена. Як правило, це відбувається у два етапи. Перший – обчислити контекстно-незалежні ознаки токена. Це набір ознак, які ніяк не залежать від оточуючих наш токен інших слів. Звичайні контекстно-незалежні ознаки – це:

- ембедінги
- символьні ознаки
- додаткові ознаки, спеціальні для конкретного завдання чи мови

Одна з найпоширеніших ознак - частина мови або POS-тег (part of speech). Такі ознаки можуть бути важливими для вирішення багатьох завдань, наприклад задачі синтаксичного парсингу. Для мов зі складною морфологією, типу російської, також важливі морфологічні ознаки: наприклад, у якому відмінку стоїть іменник, який рід прикметника. З цього

можна зробити різні висновки щодо структури речення. Також, морфологія потрібна для лемматизації (приведення слів до початкових форм), з допомогою якої ми можемо скоротити розмірність ознакового простору, і тому морфологічний аналіз активно використовується для більшості завдань NLP. [3]

Коли ми вирішуємо завдання, де важлива взаємодія між різними об'єктами (наприклад, у задачі relation extraction або при створенні питання-відповіді), нам потрібно багато знати про структуру речення - для цього потрібен синтаксичний аналіз.

Ще одним прикладом додаткової ознаки є позиція токена у тексті. Ми можемо апріорі знати, що якась сутність частіше зустрічається на початку тексту чи навпаки наприкінці.

Усі ці ознаки разом – ембедінги, символьні та додаткові ознаки – формують загальний вектор ознак токена, який не залежить від контексту.

Нагадаємо, що таке корпус - це підібрана і опрацьована сукупність текстів, які використовуються як база для дослідження мови.

Отже. наведемо алгоритм роботи сегментації і токенізації на наступному корпусі: *«I still haven't received what I bought. I've never spent so much time to receive a product purchased on the Lannister website.»*. Це речення буде розбито на 2 окремих речення: *«I still haven't received what I bought.»; «I've never spent so much time to receive a product purchased on the Lannister website.»*. Тепер ми можемо виконати токенізацію речень. У таблиці 1.2 буде записано токени для даних речень.

Таблиця 1.2

Токени утворені токенізацією наведених речень

1	2	3	4	5	6	7	8	9	10	11
I	still	haven't	received	what	I	bought	I've	never	spent	so

12	13	14	15	16	17	18	19	20	21	22
much	time	to	received	a	product	purchased	on	the	Lannister	website

Виконавши токенізацію для всіх речень корпусу, ми отримаємо масив токенів, який тепер треба представити комп’ютеру. На даному етапі ми можемо представити кожне слово структурою one-hot encoding – це вектор нулів розміром з кількість токенів, де в одній комірці стоїть 1. Індекс цієї комірки буде дорівнювати слову у векторі токенів. Таким чином ми можем представити будь-яке слово. Уявімо, що нам треба закодувати слово *«product».* У Таблиці 1.3 буде записано вектор-ідентифікатор цього слова у формі one-hot encoding.

Таблиця 1.3

Вектор-ідентифікатор слова «product»

I	still	haven't	received	what	I	bought	I've	never	spent	so
0	0	0	0	0	0	0	0	0	0	0

much	time	to	received	a	product	purchased	on	the	Lannister	website
0	0	0	0	0	1	0	0	0	0	0

Таким чином можна представити будь-яке слово у реченні, а значить ціле речення можна представити окремими векторами. Такий спосіб буде називатися *one-hot encoding*.

Розвитком такої форми представлення речень була наступна методика, яка має назву Bag of Words (BoW) або *«торба слів»*. Її сенс полягає в тому, що визначити кількість повторів слів у реченні і записати ці дані в один вектор, тобто сума повторів знаходиться у комірці до відповідного слова.

Наприклад, запишемо речення «The weather is quite hot today. She likes hot weather» методом BoW (див. Таблицю 1.4).

Таблиця 1.4

Bag of Words для речення

the	1
weather	2
is	1
quite	1
hot	2
today	1
She	1
Likes	1

Кожен ключ - це слово, а кожне значення – це число появ цього слова у тексті. Такий варіант запису слів не дозволяє відтворити осмислене речення, тому на практиці застосовується, як інструмент формування ознак.

Але незважаючи на втрату інформації навколо кожного слова, у такому вигляді ми вже здатні порівнювати тексти. Наприклад, за допомогою косинусної міри, використовуючи матрицю на рисунку 1.26.

$$cosine\ similarity = S_C(A,B) = \cos(\theta) = \frac{A*B}{\|A\|*\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} {A_i}^2} * \sqrt{\sum_{i=1}^{n} {B_i}^2}} \quad (1.46)$$

Terms	Documents								
	c1	c2	c3	c4	c5	m1	m2	m3	m4
computer	1	1	0	0	0	0	0	0	0
EPS	0	0	1	1	0	0	0	0	0
human	1	0	0	1	0	0	0	0	0
interface	1	0	1	0	0	0	0	0	0
response	0	1	0	0	1	0	0	0	0
system	0	1	1	2	0	0	0	0	0
time	0	1	0	0	1	0	0	0	0
user	0	1	1	0	1	0	0	0	0
graph	0	0	0	0	0	0	1	1	1
minors	0	0	0	0	0	0	0	1	1
survey	0	1	0	0	0	0	0	0	1
trees	0	0	0	0	0	1	1	1	0

Рисунок 1.26. Заповнена матриця повторів слів у реченнях (c_1, c_2, c_3 ...)

Перетворивши текст на «торбу слів», можна утворювати різні міри, що характеризують текст. Найбільш поширеним типом характеристик, або ознак, розрахованим за моделлю «торба слів», є частота термів, а саме, кількість разів, скільки терм з'являється в тексті.

Модернізація представлення слів продовжувалась, але значним недоліком «торби слів» був розмір такої торби. Токенізація тексту може розтягнутися на тисячі термів, а кількість повторів слів налічувати десятки або сотні. Проте, частоти термів (слова/токени) необов'язково є найкращими характеристиками тексту. Поширені слова, такі як «the», «a» майже завжди є термами з найвищою частотою в тексті. Таким чином, наявність високої частоти не обов'язково означає, що відповідне слово є більш важливим. Для вирішення цієї проблеми одним з найпопулярніших способів «нормалізації» частоти термів є вага терму до оберненої частоти документа, або TF-IDF (term frequency — inverse document frequency).

$$TF - IDF(w, d, C) = \frac{count(w,d)}{count(d)} * \log(\frac{1}{P(w,C)}) \quad (1.47)$$

Приклад використання такої моделі показано на рисунку 1.27.

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

Рисунок 1.27. Модель TF-IDF

Як можна побачити, коли терм (слово) зустрічається в двох реченнях (A і B), важливість такого терму дорівнює 0. А це значить, що під час визначення близькості двох речень A і B, цей терм не буде мати значення.

Вище описані методи стосувалися-контекстно-незалежних ознак токенів, але також існують методи формування ознак на основі контексту – це такі ознаки, або набір ознак, які містять інформацію не тільки про сам токен, а ще й про сусідні. Щоб знати як розподіляти значення у векторі, модель вирішує задачу визначення відсутнього слова, яка ґрунтується на $n-$грамах.

N-грама – представлення токенів у вигляді наборів токен з контекстом навколо нього, розмір якого визначається значенням N.

У таблиці 1.5 запишемо речення *«I still haven't received what I bought. I've never spent so much time to receive a product purchased on the Lannister website.»* у вигляді 3-грамм.

Таблиця 1.5

Запис речень у вигляді 3-грамм.

1	2	3
I	still	haven't
still	haven't	received
haven't	received	what
received	what	I
what	I	bought
I	bought	I've
bought	I've	never
I've	never	spent
never	spent	so
spent	so	much
so	much	time
much	time	to
time	to	receive
to	receive	a
receive	a	product
a	product	on
product	on	the
on	the	Lannister
the	Lannister	website

На основі такого набору послідовностей слів можна представляти контекст слова, що дає більшу обізнаність про речення і слово взагалом. На основі N-грам наборів можна предсказувати слова за методиками навчання *Continious bag of words* (CBoW) або *k-skip gramm*.

Continious bag of words (CBOW) – техніка навчання моделі векторізації слів. Принцип навчання полягає в прогнозуванні пропущеного слова на основі n-кількості слів. Іншими словами, спрогнозувати слово за поданим контекстом.

k-skip-gramm (skip gramm) - техніка навчання моделі векторізації слів. Принцип навчання полягає в прогнозуванні слів, які підходять за сенсом,

навколо заданого слова. Іншими словами, спрогнозувати контекст за поданим словом.

Вищеперераховані способи дозволяли зменшити розмір вибірок, збільшити значення окремих токенів під час аналізу. Проте такі способи запису слів не дають змоги знаходити близькі за значенням слова або слова, які використовуються в однакових контекстах. До цього також додається проблема масштабування вектору. Чим більше стає набір, тим більше часу потребується для доповнення новим значенням. У цій ситуації на допомогу нам прийде метод ембедінгу, який називається Word2Vec.

1.5.2 Word2Vec

У 2013 році тоді мало кому відомий чеський аспірант Томаш Міколов запропонував свій підхід до word embedding, який він назвав word2vec. Його підхід заснований на іншій важливій гіпотезі, яку в науці прийнято називати гіпотезою локальності - "слова, що зустрічаються в однакових оточеннях, мають близькі значення". Слова будуть поміщатися в ковзне вікно, для якого існує контекст N-го розміру слів. Таким чином буде утворюватися оточення для заданого слова, саме звідси і виходить принцип локальної близькості. Близькість у цьому випадку розуміється дуже широко, як те, що поруч можуть стояти лише слова, що поєднуються. Наприклад, ми можемо сказати «hot cream», але не можемо сказати «hot ice-cream» - ці слова не поєднуються.

Модель, запропонована Міколовим дуже проста — ми передбачатимемо ймовірність слова спираючись на його оточення (контекст). Тобто ми навчатимемо такі вектори слів, щоб ймовірність, яку присвоює модель слову, була близька до ймовірності зустріти це слово в цьому оточенні в реальному тексті.

$$P(w_0, w_c) = \frac{e^{S(w_0, w_c)}}{\sum_{w_i} e^{S(w_0, w_c)}} \tag{1.48}$$

де w_0 — вектор цільового слова, w_C — це певний вектор контексту, обчислений (наприклад, шляхом усереднення) векторів оточуючих потрібне слово інших слів. $S(w_1, w_2)$ — це функція, яка двом векторам зіставляє одне число. Наприклад, це може бути згадувана вище косинусна відстань.

Наведена формула називається softmax, тобто "м'який максимум", м'який - у сенсі диференційований. Це потрібно для того, щоб наша модель могла навчитися за допомогою backpropagation, тобто процесу зворотного розповсюдження помилки.

Формула softmax наступна:

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad (1.49)$$

де e^{z_i} − це ймовірність слова i бути рішенням задачі.

Як можна побачити, у знаменнику розраховується велике число по всьому словнику, щоб визначити одне значення. Такий підхід буде вимагати значних ресурсів для виконання обчислень. Наприклад, щоб розрахувати одне слово в 10 тисячному словнику, треба під час кожного рішення розраховувати значення по 10 тисячам слів. Щоб оптимізувати цей процес було запропоновано hierarchical softmax.

Припустимо, що ми можемо побудувати структуру дерева для всього корпусу, кожен лист у дереві представляє слово з корпусу. Ми обходимо дерево, щоб обчислити ймовірність слова. Ймовірність кожного слова буде добутком ймовірності вибору гілки, яка знаходиться на шляху від кореня до слова. [22]

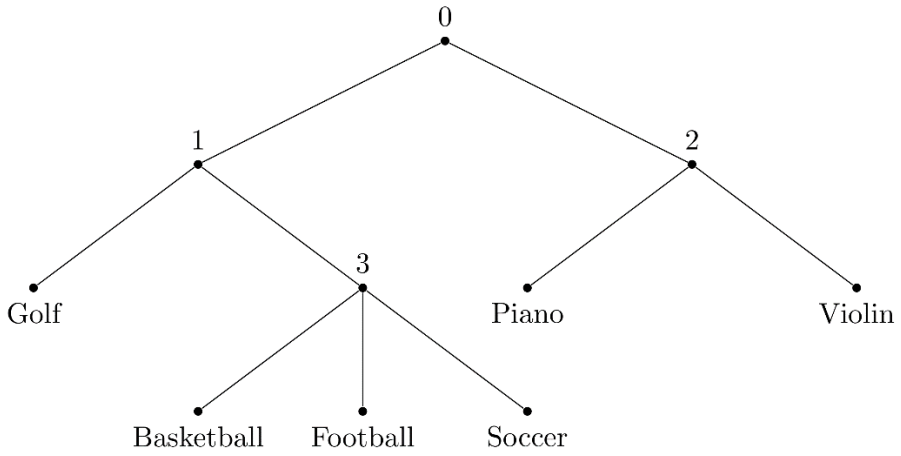

Рисунок 1.28. Дерево рішень для hierarchical softmax

Процес розрахування ймовірності слова «Football» буде виглядати наступним чином:

$$P_\theta^h(w = Football) = P_\theta^h(0 \to 1) \cdot P_\theta^h(1 \to 3) \cdot P_\theta^h(3 \to Football)$$

Запишемо математичну формулу для такого виразу:

$$P_\theta^h(w) = \prod_{j=1}^{L(w)-1} P_\theta^h\big(n(w,j) \to n(w,j+1)\big) \qquad (1.50)$$

де h −якийсь контекст навколо слова, $\theta -$ деякі параметри системи.

Таким чином, маючи корпус, якщо ми належним чином побудуємо дерево, ми можемо зменшити складність обчислення softmax з $O(N)$ до $O(\log N)$, де N розмірність корпусу. Наприклад, якщо у нас є корпус із 10 000 слів, ми будуємо двошаровий ієрархічний softmax, перший рівень складається зі 100 дочірніх вузлів, кожен вузол у першому шарі також складається зі 100 дочірніх вузлів. Щоб обчислити звичайний softmax, нам потрібно буде обчислити $e^{S(w_0,w_c)}$ 10 000 раз. Щоб обчислити ієрархічний softmax, нам просто потрібно обчислити $e^{S(w_0,w_c)}$ 100 раз на першому слої та 100 раз на другому, загалом 200 раз.

Процес тренування моделі відбувається наступним чином: ми беремо послідовно $(2k + 1)$ слів, слово у центрі є словом, яке має бути передбачено. А оточуючі слова є контекстом довжини k з кожної сторони. Така форма запису називається K-gram. Кожному слову у нашій моделі зіставлено унікальний вектор, який ми змінюємо у процесі навчання нашої моделі. Тепер починаємо видаляти одне слово із запису K-gram і починаємо прогнозувати відсутнє слово.

У цілому цей підхід називається $CBOW$ — continuous bag of words, continuous тому, що ми передаємо нашій моделі послідовно набори слів з тексту, а BoW тому що порядок слів у контексті не важливий.

Також Міколовим відразу було запропоновано інший підхід — прямо протилежний CBOW, який він назвав skip-gram, тобто словосполучення з пропуском. Ми намагаємося із заданого слова вгадати його контекст (точніше вектор контексту). В іншому модель не зазнає змін. На рисунку 1.29 наведемо концептуально моделі CBow та skip-gram.

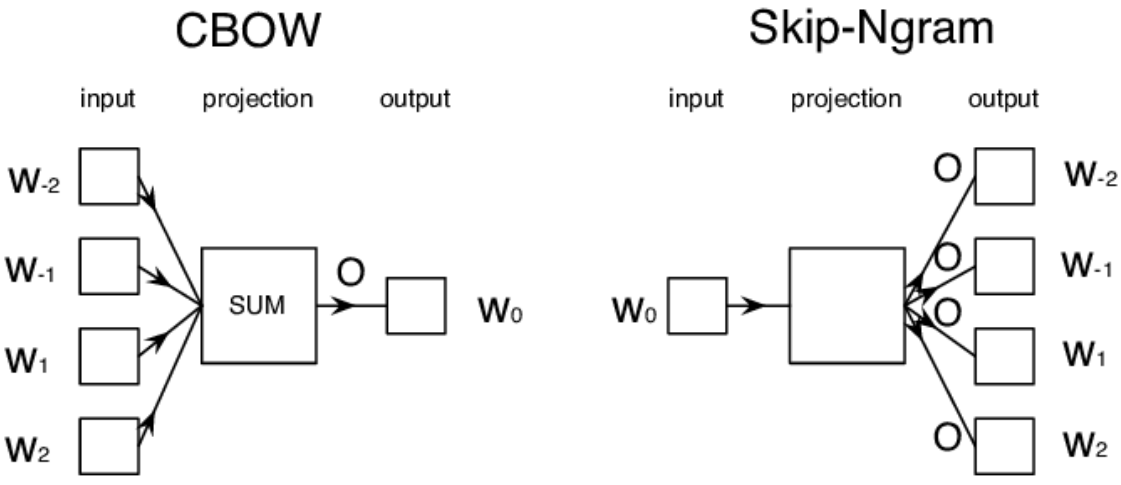

Рисунок 1.29. Концептуальна модель методів навчання.

Наведемо приклади, щоб зрозуміти алгоритм роботи технік. Візьмемо за основу 3-граму {«time»; «to»; «receive»}. Опишемо алгоритм CBOW - виберемо з 3-грами 2 токени «time» і «to». Тепер ми маємо сказати моделі

знайти токен, який потрібно поставити, щоб утворити правильну або схожу на правильну (за сенсом) 3-граму. Зазвичай така задача дається нейронній мережі, яка отримує на вхід $n-1$ токенів. Після цього проходить декілька схованих нейронних слоїв і виводить k можливих слів, з відповідною вірогідністю слова бути використаним в даній 3-грамі. Зазвичай обирається слово з максимальною вірогідністю.

Тепер опишемо алгоритм для skip-gramm. Візьмемо за основу ту саму 3-граму з прикладу вище - {«time»; «to»; «receive»}. Виберемо за основу токен «to» та покладемо його на вхід нейронної мережі. Тепер модель повинна перебрати усі можливі комбінації, щоб знайти вірогідний контекст навколо токену «to» - тобто вірогідні токени, які використовуються з токеном «to». Результати можуть бути наступними: {«time»; **«to»**; «eat»}, {«going»; **«to»**; «school»}, {«give»; **«to»**; «your»}. Заданий токен виділено жирним шрифтом, а варіанти контексту записано звичайним. Як видно, результат може бути абсолютно різними, але плюс який можна отримати з цього – це те, що модель може розуміти, що після токену не може бути прийменник «at», наприклад, {«give»; **«to»**; «at»}.

Таким чином, проходячи нейрону мережу, модель редагує значення векторів у словнику токенів, щоб під час наступного тесту виводити потрібну інформацію. Таким чином заміняється старий one-hot encoding на безперервний $N-$вимірний простір.

Наприклад, переводячи наш вектор токенів у 10-вимірний дійсний простір, ми будемо представляти слово «product» наступним чином (див. Таблиця 1.6).

Таблиця 1.6

Кодування слова «product» після ембедінгу методом word2vec

1	2	3	4	5	6	7	8	9	10
-3	1,8	3,9	-5	1,9	4,9	0,3	1,1	2,4	3,2

Як можна побачити, після виконання процедури Word Embedding, кількість параметрів для опису 1 слова зменшилась кратно, наприклад, у порівнянні з one-hot encoding з 1000 слів. При цьому, щоб описати інше слово або додати нове, не потрібно розширяти вектор, як у випадку one-hot encoding.

Для того, щоб покращити результат розв'язання задачі вибору слова та тренування можливості відрізняти шум від потрібних слів, використовують методику negative sampling. Ця методика тренування припускає заповнення множини допустимих значень неправильними значеннями. У таблиці 1.7 зображено приклад такої множини. Для речення «I put my backpack on and will go to school» було обрано розбиття 4-gramm. Правильні грами було позначено значенням «1» у стовпчику result, неправильні – «0».

Таблиця 1.7

4-gram для задачі

1	2	3	4	result
I	put	my	back	1
put	my	back	on	1
my	back	on	and	1
back	on	and	will	1
on	and	will	go	1
and	will	go	to	1
will	go	to	school	1
go	to	school	on	1
an	sh	go	to	0

d	ould			
back	on	the	will	0
carry	put	my	back	0

Тепер, коли модель буде навчатися розпізнавати слова за допомогою контекста, вона буде відрізняти слова, які треба ставити на визначенному місці, а які не треба.

Повернемося до самої ідеї заповнення дійсного простору значень. Результатом такого способу заповнення вектора з 10 параметрів є те, що між окремому слову відповідає вектор, за допомогою якого можна розрахувати скалярну відстань, яка буде співпадати між словами одного роду або синонімами. Міколов демонструє у своїй роботі наступний малюнок (див. Рисунок 1.30), відстань між словами {“man”, “woman”}, {“uncle”, “aunt”}, {“king”,”queen”} приблизно однакова, що показує те, що модель, запропонована ним, може відрізняти слова антоніми, а різниця між підвидами цих антонімів буде приблизно однакова.

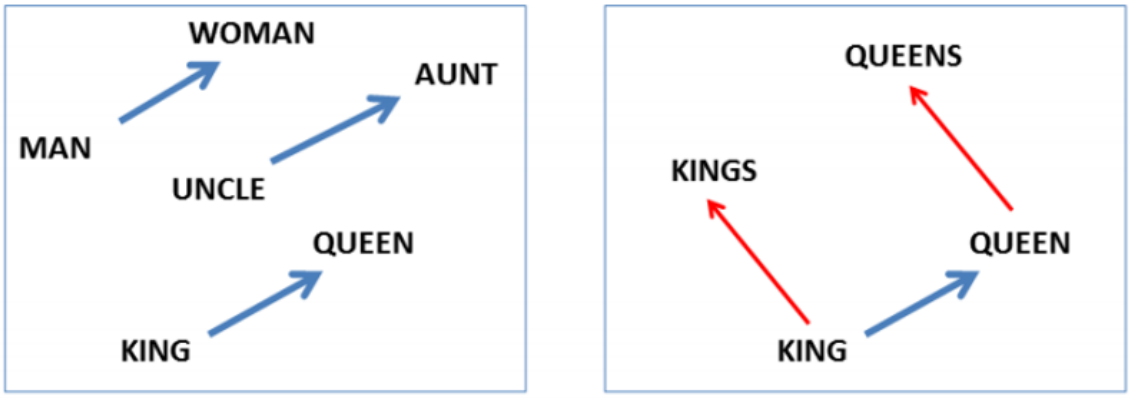

Рисунок 1.30. Приклад семантичного зв’язку токенів.

Цікавий приклад схожості деяких значень векторів між словами одного роду наведено у рисунку 1.31. Уявімо, що схожі кольори відповідають близості дійсних чисел координат вектору ембедінгу слова.

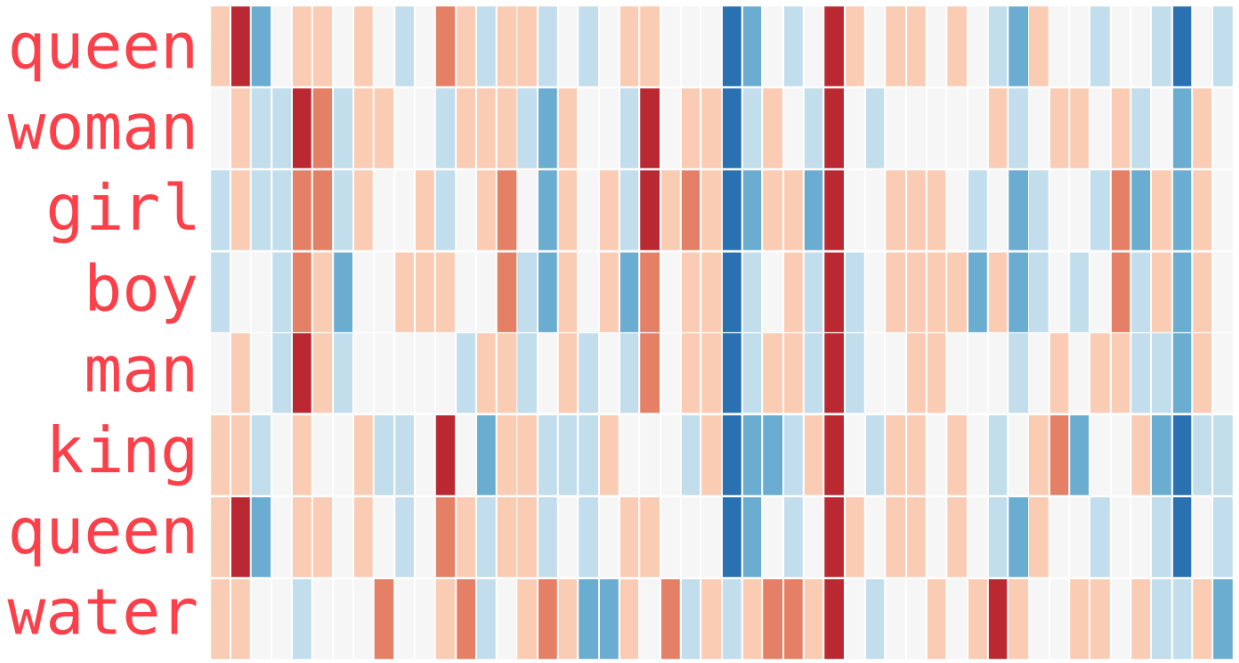

Рисунок 1.31 Представлення слів за допомогою вектору кольорів.

Можна помітити кілька речей:

1.Через усі слова проходить одна червона колонка. Тобто ці слова схожі у цьому конкретному вимірі (і ми не знаємо, що в ньому закодовано).

2.Ви можете побачити, що «жінка» та «дівчина» багато в чому схожі. Те саме з «чоловіком» і «хлопчиком».

3.«Хлопчик» та «дівчинка» теж схожі у деяких вимірах, але відрізняються від «жінки» та «чоловіка». Можливо, це закодоване невиразне уявлення про молодість.

4.Все, крім останнього слова, — це уявлення людей. Було додано об'єкт (воду), щоб показати різницю між категоріями. Наприклад, ви можете побачити, як синій стовпець йде вниз і зупиняється перед вектором води.

5.Є точні виміри, де «король» і «королева» схожі друг на друга і від інших. Можливо, там закодовано розпливчасту концепцію королівської влади.[23]

1.5.2 GloVe

Метод ембедінгу GloVe схожий за своїми властивостями з попереднім методом word2vec. Цей метод був представлений розробниками університету Стенфорду пізніше на рік від word2vec.

Назва методу розшифровується як Global Vectors. Діло в тому, що алгоритм використовує для навчання конкретного слова не тільки локальний контекст речення (ковзне вікно N розміру), а також весь корпус моделі або глобальні вектори. Це робиться для того, щоб розрахувати кількість повторів кожного слова, а також повтори контекстних слів по всьому тексту. Це дозволяє визначити ширше значення слова, наприклад, у великому корпусі слово «тверда речовина» частіше зустрічається разом із словом «лід», ніж «пара», але слово «газ», ймовірно, зустрічається разом із словом «пара» частіше, ніж слово «лід». Підхід порівнювати дані на всьому тексті дають змогу знаходити більше синонімів до слів, тому що аналізується ширший контекст використання слів. [24]

У порівнянні з word2vec, метод Global Vectors має математичне тлумачення для виявлення семантичних зв’язків між словами, можна сказати це покращена версія того методу.

На початку роботи завантажується корпус тексту, який розбивається на менші частини: абзаци, речення. Після цього речення розбивається на токени і формується квадратна матриця взаємного розташування, де у нас $i -$ це задане слово, а $j -$ контекстне слово, яке зустрічається разом зі словом, яке стоїть під індексом i. Відповідно на перетині i та j буде знаходитится $X_{i,j}$, що буде містити в собі кількість повторів використання цих слів разом. [24]

Для наглядності продемонструємо як використовується сума використання слів у тексті. У таблиці 1.8 запишемо ймовірність використання слова в контексті з заданим словом. Заданими словами будуть: «ice» (лід) та «steam» (пар). Слова, які використовуються у контексті

наступні: «solid»(твердий), «gas»(газ), «water»(вода), «fashion» (мода). Як видно, використання слова «solid» у контексті до слова «ice» більше, ніж до слова «steam». Протилежна ситуація зі словом «steam» - слово «ice» рідше зустрічається з цим словом. Далі ми можемо порівняти ці два слова у міжу викоистані використані. Таким чином, якщо відношення більше 1, то головне слово ймовірніше слово зверху, якщо менше 1 – слово знизу. [24]

Таблиця 1.8

Таблиця вірогідностей слів

Probability and Ratio	$k = solid$	$k = gas$	$k = water$	$k = fashion$
$P(k\|ice)$	1.9×10^{-4}	6.6×10^{-5}	3.0×10^{-3}	1.7×10^{-5}
$P(k\|steam)$	2.2×10^{-5}	7.8×10^{-4}	2.2×10^{-3}	1.8×10^{-5}
$P(k\|ice)/P(k\|steam)$	8.9	8.5×10^{-2}	1.36	0.96

Далі запишемо «м’яке» обмеження для кожної пари слів.

$$w_i{}^T w_j + b_i + b_j = \log(X_{i,j}) \tag{1.51}$$

де w_i – вектор головного слова з матриці, w_j – вектор слова, яке зустрічається в контексті, b_i, b_j – скалярні зміщення для головних і контекстних слів.

Після цього ми можемо записати функцію втрат для моделі GloVe.

$$J = \sum_{i=1}^{V} \sum_{j=1}^{V} f\left(X_{i,j}\right) \cdot \left(w_i^T w_j + b_i + b_j - \log X_{ij}\right)^2 \tag{1.52}$$

де $f(X)$ – це функція зважування пари слів, яка дозволяє обмежувати пари слів, які часто зустрічаються між собою.

$$f\left(X_{i,j}\right) = \begin{cases} \left(\frac{X_{i,j}}{X_{max}}\right)^{\alpha}, \text{якщо } X_{i,j} < X_{max} \\ 1, \text{якщо } X_{i,j} \geq X_{max} \end{cases} \tag{1.53}$$

Цей метод, разом з word2vec, є на «озброєнні» сучасних розробників NLP, тому що не потребує довгої обробки та навчання, потрібно лише завантажити корпус з речень, на яких потрібно лише обучити модель.

1.6 Висновок до розділу

У цьому розділі було описано методи обробки природньої мови, методи класифікації. Класифікація може виконуватися за різними темами, тобто можна класифікувати дані по провідній темі, по тональності, по відношенню до групи.

До алгоритмів класифікації відносяться метод наївного баєсу, де виконується розрахунок ймовірності гіпотези за умови виконання дії; метод опорних векторів, де виконується пошук гіперплощини, яка максимально розділяє множину між класами даних; метод дерев прийняття рішень, де вузлами є умови для признаків, які відповідно визначають результат класифікації; метод K-ближчих сусідів, де клас обирається по більшості співпадінь у визначеному колі. На основі вищевикладених матеріалів підведемо підсумок всьому розділу і сформуємо алгоритм для реалізації класифікатора відгуків для наступного розділу.

Після цього було описано представлення природньої мови у вигляді векторів, а також методи переведення у векторний простір меншої розмірності (ембедінг). Серед методів ембедінгу першим є one-hot encoding, вектор цілочислений і складається зі всіх слів у тексті, недоліком методу є складність до масштабування і втрата контексту навколо слова. Наступним методом, який описувався був метод «мішок слів», де розмірність вектору вже зменшилась, але контекст навколо слів все рівно втрачений. Повернення контексту з’явилося у методі ембедінгу Міколова, який називається word2vec. Цей метод обробляє локальний контекст навколо слова, за допомогою ковзного вікна відповідної розмірності. Метод вже вміє знаходити синоніми і пов’язувати слова, які використовуються разом. Розвинутою версією цього методу став GloVe, який замість використання

локального контексту, використовує весь корпус текстів, тому він може знаходити ще більше синонімів і слів, які можуть використовуватися разом.

На основі вищевикладених даних можна описати алгоритм створення класифікатора відгуків споживачів, який буде використано для практичної реалізації у наступному розділі.

Перед тим, як почати класифікувати відгуки, потрібно створити і навчити модель, яка зможе обробляти природню мову, а саме виконати ембедінг корпусу текстів - перевести масив текстів у векторне представлення. Для виконання цього завдання, спочатку треба підготувати датасет, у нашому нашому випадку це буде датасет відгуків. Очистити його від зайвих елементів і неточностей, а також виконати розмітку по класам тональності: позитивний і негативний. Далі потрібно виконати процедуру ембедінгу за одним із методів, які було описано у розділі 2.4. Для кожного з них існує бібліотека написана на мові програмування Python, назви бібліотек наступні: word2vec розроблена Gensim, GloVe розроблена у Stanford.

Після виконання процедури ембедінгу, наступним етапом є побудова класифікатора. У розділі 2.3 було описано декілька способів класифікації, проте популярними імплеменатціями класифікаторів є методи дерев прийняття рішень, а саме метод екстримального градієнтного бустингу. Популярними бібліотеками, які можна використовувати, будуть XGBoost (тільки градієнтний бустинг), scikit-learn (представлені всі методи класифікації, описані у розділі 2.3).

Перед порівнянням моделей введемо метрики, за якими будемо порівнювати їх якість класифікації і визначити кращу. Серед метрик будемо використовувати значення True Positive, True Negative, False Positive, False Negative rates; F-міри; ROC-curve; ROC-AUC.

True Positive Rate – кількість позитивних елементів, які правильно класифіковано. Відповідно False Negative Rate – це кількість позитивних

елементів, які неправильно класифіковано. Для True Negative Rate і False Positive Rate ситуація аналогічна, тільки замість позитивних елементів – негативні. Показники TPR (Recall) та TNR (Specificity) це показники повноти та специфічності. Вони показують яку частку множини було правильно класифіковано. Показник False Negative називається помилкою першого роду, а False Positive – помилка другого роду.

$$TPR(Recall) = \frac{TP}{TP+FN} \tag{1.54}$$

$$FPR = 1 - TPR \tag{1.55}$$

$$TNR(\text{Specificity}) = \frac{TN}{TN+FP} \tag{1.56}$$

$$FNR = 1 - TNR \tag{1.57}$$

Для того щоб визначити точність прогнозу, його влучність використовуються метрики Accuracy та Precision.

$$Accuracy = \frac{TP+TN}{TP+FN+TN+FP} \tag{1.58}$$

$$Precision = \frac{TP}{TP+FP} \tag{1.59}$$

На основі цих показників, ми можемо розрахувати значення F-міри. Цей показник гармонічно усереднює значення точності за допомогою мір Precision і Recall.

$$F - score = \frac{2TP}{2TP+FP+FN} \tag{1.60}$$

Наступним показником буде ROC крива, яка показує зміну False Positive Rate при зміні True Positive Rate. Мірилом є лінія y=x, яка займає половина площини та називається випадковим класифікатором, тому що вірогідність отримати True Positive чи False Positive постійно дорівнює 0,5. Суть в тому, що при збільшенні TPR показник FPR зростав повільніше. Приклад ROC кривої випадкового класифікатора і кращого, ніж випадкового, зображено на рисунку 1.32.

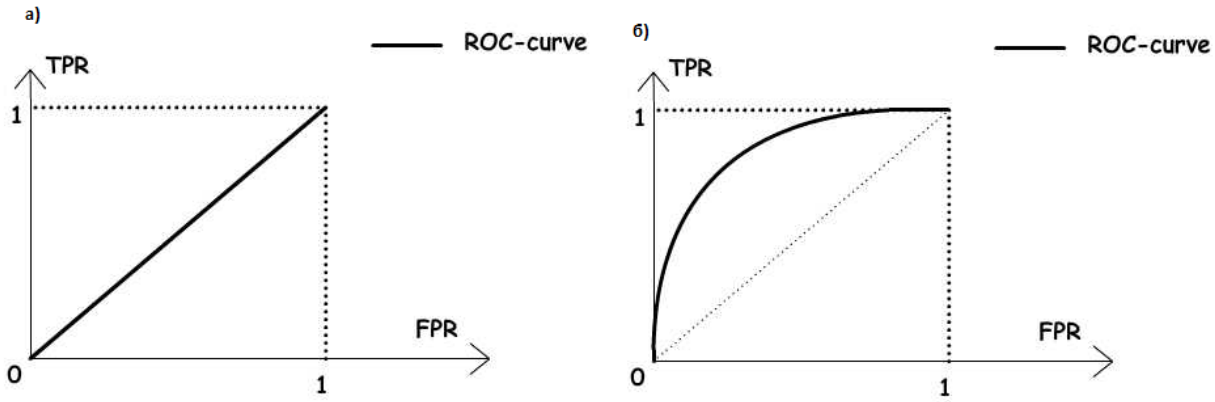

Рисунок 1.32. Приклад ROC кривої випадкового (а) та трохи кращого класифікатора (варіант б).

Ще одним показником, який будується за допомогою ROC кривої, це ROC-AUC (Area Under Curve або площа під кривою). Полягає в тому що чим більше площину займає, тим сильніше тягнеться крива до лівого кута графіку, а саме до точки (0,1). А цей факт каже про те, що при збільшені TPR, значення FPR зростає меншими темпами.

2. СПЕЦІАЛЬНИЙ РОЗДІЛ

2.1 Опис об’єкту дослідження

Об’єкт дослідження є бразильский маркетплейс Olist (інтернет-магазин), через який продавці зі всієї країни реалізують різноманітну продукцію, починаючи з телефонів, спортивного знаряддя, закінчуючи товарами для автомобілів. Продавці можуть виставляти на продаж товари, фотографії цього товару, а також збирати відгуки від інших покупців, які вже отримали цей товар. Обсяг замовлень за період з 2017 – серпень 2018 сягає майже 100 тисяч замовлень, а загальна сума продажів склала 7 млн бразильских реалів або майже 2 млн доларів США. Показник продаж монотонно зростає на всьому відрізку дослідження, що свідчить про розвиток бізнесу. База даних містить дані про майже сто тисяч покупців (99442), 3 тисячі продавців і налічує майже сто тисяч замовлень (99440), кількість відгуків трохи менше за число замовлень (99224).

Сайт окремо накопичував інформацію про свою діяльність. Було сформовано вісім наборів даних, серед яких: дані про замовлення, дані з чого складається замовлення, дані про відгуки, дані про оплату, дані продавців, дані покупців, дані географічних положень, дані про товар. Використовуючи накопичені дані, ми будемо досліджувати поведінку споживачів, а саме використовуючи їхні відгуки про замовлення.

2.2 Структура моделі даних

Перед початком роботи з моделлю даних, ознайомимся зі структурою маркетплейсу і особливостями кожного елемента його моделі. Кожен елемент зберігає дані у відповідному стандарті оформлення, які відносяться до одного із процесів діяльності. Наприклад, інформація про клієнтів зберігається в одній таблиці, а список їх замовлень в іншій. Ці таблиці можна поєднати у модель даних, використовуючи ключи для ідентифікації зв’язку рядку однієї таблиці з рядком з іншої таблиці.

Опишемо таблиці, які є складовими моделі, та розглянемо декілька записів з них.

Таблиця клієнтів (customers_dataset) і продавців (sellers_dataset) складається з наступним полів:

1. Унікальний ідентифікатор особи (інформація про ім’я, прізвище відсутня через політику конфіденційності);
2. Індекс міста, де мешкає клієнт;
3. Назва міста;
4. Назва регіону країни;

Приклади записів з таблиці зображено на рисунку 2.1 – 2.2.

	customer_id	customer_zip_code_prefix	customer_city	customer_state
1	06b8999e2fba1a1fbc88172c00ba8bc7	14409	franca	SP
2	18955e83d337fd6b2def6b18a428ac...	09790	sao bernardo do campo	SP
3	4e7b3e00288586ebd08712fdd0374...	01151	sao paulo	SP
4	b2b6027bc5c5109e529d4dc6358b1...	08775	mogi das cruzes	SP
5	4f2d8ab171c80ec8364f7c12e35b23ad	13056	campinas	SP
6	879864dab9bc3047522c92c82e1212...	89254	jaragua do sul	SC
7	fd826e7cf63160e536e0908c76c3f441	04534	sao paulo	SP
8	5e274e7a0c3809e14aba7ad5aae0d4...	35182	timoteo	MG
9	5adf08e34b2e993982a47070956c5c...	81560	curitiba	PR
10	4b7139f34592b3a31687243a302fa7...	30575	belo horizonte	MG
11	9fb35e4ed6f0a14a4977cd9aea4042...	39400	montes claros	MG
12	5aa9e4fdd4dfd20959cad2d7725095...	20231	rio de janeiro	RJ

Рисунок 2.1. Приклад записів з таблиці покупців.

	seller_id	seller_zip_code_prefix	seller_city	seller_state
1	3442f8959a84dea7ee197c632cb2df15	13023	campinas	SP
2	d1b65fc7debc3361ea86b5f14c68d2e2	13844	mogi guacu	SP
3	ce3ad9de960102d0677a81f5d0bb7b...	20031	rio de janeiro	RJ
4	c0f3eea2e14555b6faeea3dd58c1b1c3	4195	sao paulo	SP
5	51a04a8a6bdcb23deccc82b0b80742cf	12914	braganca paulista	SP
6	c240c4061717ac1806ae6ee72be353...	20920	rio de janeiro	RJ
7	e49c26c3edfa46d227d5121a6b6e4d...	55325	brejao	PE
8	1b938a7ec6ac5061a66a3766e0e75f...	16304	penapolis	SP
9	768a86e36ad6aae3d03ee3c6433d6...	1529	sao paulo	SP
10	ccc4bbb5f32a6ab2b7066a4130f114e3	80310	curitiba	PR

Рисунок 2.2. Приклад записів з таблиці продавців.

Наступна таблиця geolocations_dataset буде описувати міста, які підписані відповідають кожному індексу міста:

1. Індекс міста;
2. Координата за шириною;
3. Координата за довжиною;
4. Назва міста;
5. Регіон країни;

Приклади записів з таблиці зображено на рисунку 2.3.

geolocation_zip_code_prefix	geolocation_lat	geolocation_lng	geolocation_city	geolocation_state
1037	-23.54562128115268	-46.63929204800168	sao paulo	SP
1046	-23.546081127035535	-46.64482029837157	sao paulo	SP
1041	-23.5443921648681	-46.63949930627844	sao paulo	SP
1035	-23.541577961711493	-46.64160722329613	sao paulo	SP
1012	-23.547762303364266	-46.63536053788448	sГЈo paulo	SP
1047	-23.546273112412678	-46.64122516971552	sao paulo	SP
1013	-23.546923208436723	-46.6342636964915	sao paulo	SP
1029	-23.543769055769133	-46.63427784085132	sao paulo	SP
1011	-23.547639550320632	-46.63603162315495	sao paulo	SP

Рисунок 2.3. Приклад записів з таблиці геолокацій.

Щоб зв’язати покупця і продавця, потрібно додати таблицю товарів, які продаються, і таблицю продажів, де об’єднано ці 3 складові.

Таблиця товарів products_dataset буде мати наступні поля:

1. Індекс товару;
2. Назва товару;
3. Довжина назви;
4. Довжина опису;
5. Кількість фотографій;
6. Вага товару;
7. Довжина товару;
8. Висота товару;
9. Ширина товару;

Приклади записів з таблиці зображено на рисунку 2.4.

	product_id	product_category_name	product_na...	product_des...	produ...	product_weig...	product_leng...	product_heigh...	product_wi...
1	1e9e8ef04dbcff4541ed26657ea517e5	perfumaria	40	287	1	225	16	10	14
2	3aa071139cb16b67ca9e5dea641aaa...	artes	44	276	1	1000	30	18	20
3	96bd76ec8810374ed1b65e2919757...	esporte_lazer	46	250	1	154	18	9	15
4	cef67bcfe19066a932b7673e239eb23d	bebes	27	261	1	371	26	4	26
5	9dc1a7de274444849c219cff195d0b71	utilidades_domesticas	37	402	4	625	20	17	13
6	41d3672d4792049fa1779bb35283e...	instrumentos_musicais	60	745	1	200	38	5	11
7	732bd381ad09e530fe0a5f457d81be...	cool_stuff	56	1272	4	18350	70	24	44
8	2548af3e6e77a690cf3eb6368e9ab61e	moveis_decoracao	56	184	2	900	40	8	40
9	37cc742be07708b53a98702e77a21a...	eletrodomesticos	57	163	1	400	27	13	17
10	8c92109888e8cdf9d66dc7e4630255...	brinquedos	36	1156	1	600	17	10	12
11	14aa47b7fe5c25522b47b4b29c98dc...	cama_mesa_banho	54	630	1	1100	16	10	16
12	03b63c5fc166915305 86ae020c3455...	bebes	49	728	4	7150	50	19	45
13	cf55509ea8edaaac1d28fdb16e48fc22	instrumentos_musicais	43	1827	3	250	17	7	17
14	7bb6f29c2be57716194f96496660c7c2	moveis_decoracao	51	2083	2	600	68	11	13
15	eb31436580a610f202c859463d8c74...	construcao_ferramentas_seguranca	59	1602	4	200	17	7	17
16	3bb7f144022e6732727d8d838a7b1...	esporte_lazer	22	3021	1	800	16	2	11
17	6a2fb4dd53d2cdb88e0432f1284a00...	perfumaria	39	346	2	400	27	5	20
18	a1b71017a84f92fd8da4aeefba108a24	informatica_acessorios	59	636	1	900	40	15	20
19	a0736b92e52f6cead290e30b578413...	moveis_decoracao	56	296	2	1700	100	7	15
20	f53103a77d9cf245e579ea37e5ec51f0	cama_mesa_banho	52	206	1	500	16	10	16
21	1c1890ba1779090cd54008a3c33029...	moveis_decoracao	27	158	4	2550	29	24	45

Рисунок 2.4. Приклад записів з таблиці товарів.

Таблиця, яка буде поєднувати товар з продавцем, буде мати назву order_items_dataset і містити наступні поля:

1. Ідентифікатор замовлення;
2. Ідентифікатор продукту;
3. Ідентифікатор продавця;
4. Ціна;
5. Ціна перевезення.

Приклади записів з таблиці зображено на рисунку 2.5.

	order_id	order_item_id	product_id	seller_id	shipping_limit_date	price	freight_value
1	00010242fe8c5a6d1ba2dd792cb162...	1	4244733e06e7ecb4970a6e2683c13...	48436dade18ac8b2bce089ec2a0412...	19.09.2017 09:45:35	58,9	13,29
2	00018f77f2f0320c557190d7a144bdd3	1	e5f2d52b802189ee658865ca93d83a...	dd7ddc04e1b6c2c614352b383efe2d...	03.05.2017 11:05:13	239,9	19,93
3	000229ec398224ef6ca0657da4fc703e	1	c777355d18b72b67abbeef9df44fd0fd	5b51032eddd242adc84c38acab88f2...	18.01.2018 14:48:30	199	17,87
4	00024acbcdf0a6daa1e931b038114c...	1	7634da152a4610f1595efa32f14722fc	9d7a1d34a5052409006425275ba1c...	15.08.2018 10:10:18	12,99	12,79
5	00042b26cf59d7ce69dfabb4e55b4fd9	1	ac6c3623068f30de03045865e4e100...	df560393f3a51e74553ab94004ba5c...	13.02.2017 13:57:51	199,9	18,14
6	00048cc3ae777c65dbb7d2a0634bc1...	1	ef92defde845ab8450f9d70c526ef70f	6426d21aca402a131fc0a5d0960a3c90	23.05.2017 03:55:27	21,9	12,69
7	00054e8431b9d7675808bcb819fb4a...	1	8d4f2bb7e93e6710a28f34fa83ee7d28	7040e82f899a04d1b434b795a43b4...	14.12.2017 12:10:31	19,9	11,85
8	000576fe39319847cbb9d288c5617fa6	1	557d850972a7d6f792fd18ae1400d9...	5996cddab893a4652a15592fb58ab8...	10.07.2018 12:30:45	810	70,75
9	0005a1a1728c9d785b8e2b08b9045...	1	310ae3c140ff94b03219ad0adc3c778f	a416b6a846a11724393025641d4ed...	26.03.2018 18:31:29	145,95	11,65
10	0005f50442cb953dcd1d21e1fb9234...	1	4535b0e1091c278dfd193e5a1d63b3...	ba143b05f0110f0dc71ad71b4466ce...	06.07.2018 14:10:56	53,99	11,4
11	00061f2a7bc09da83e415a52dc8a4af1	1	d63c1011f49d98b976c352955b1c4b...	cc419e0650a3c5ba77189a1882b755...	29.03.2018 22:28:09	59,99	8,88
12	00063b381e2406b52ad429470734e...	1	f177554ea93259a5b282f24e33f65ab6	8602a61d680a10a82cceeeda0d99ea...	31.07.2018 17:30:39	45	12,98
13	0006ec9db01a64e59a68b2c340bf65...	1	99a4788cb24856965c36a24e339b6...	4a3ca9315b744ce9f8e93743614938...	26.07.2018 17:24:20	74	23,32

Рисунок 2.5. Приклад записів з таблиці поєднання товару і продавця.

Таблиця з замовленнями буде мати назву orders_dataset і містити наступні поля:

1. Ідентифікатор замовлення;
2. Ідентифікатор покупця;

3. Статус замовлення (доставлено, скасовано, створено, доставляється, обробляється, недоступно);
4. Дата і час покупки;
5. Дата і час підтвердження;
6. Дата і час надходження до сервісу доставки;
7. Дата і час доставки до клієнта (фактично);
8. Дата і час доставки до клієнта (приблизно);

Приклади записів з таблиці зображено на рисунку 2.6.

	order_id	customer_id	o...	order_purchase_timestamp	order_approved_at	order_delivered_carrier_date	order_delivered_custo...	order_estimated_delive...
1	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a9...	delivered	02.10.2017 10:56:33	02.10.2017 11:07:15	04.10.2017 19:55:00	10.10.2017 21:25:13	18.10.2017 00:00:00
2	53cdb2fc8bc7dce0b6741e21502734...	b0830fb4747a6c6d20dea0b8c802d7...	delivered	24.07.2018 20:41:37	26.07.2018 03:24:27	26.07.2018 14:31:00	07.08.2018 15:27:45	13.08.2018 00:00:00
3	47770eb9100c2d0c44946d9cf07ec6...	41ce2a54c0b03bf3443c3d931a3670...	delivered	08.08.2018 08:38:49	08.08.2018 08:55:23	08.08.2018 13:50:00	17.08.2018 18:06:29	04.09.2018 00:00:00
4	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d...	delivered	18.11.2017 19:28:06	18.11.2017 19:45:59	22.11.2017 13:39:59	02.12.2017 00:28:42	15.12.2017 00:00:00
5	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad...	delivered	13.02.2018 21:18:39	13.02.2018 22:20:29	14.02.2018 19:46:34	16.02.2018 18:17:02	26.02.2018 00:00:00
6	a4591c265e18cb1dcee52889e2d8ac...	503740e9ca751ccdda7ba28e9ab8f6...	delivered	09.07.2017 21:57:05	09.07.2017 22:10:13	11.07.2017 14:58:04	26.07.2017 10:57:55	01.08.2017 00:00:00
7	136cce7faa42fdb2cefd53fdc79a6098	ed0271e0b7da060a393796590e7b7...	invoiced	11.04.2017 12:22:08	13.04.2017 13:25:17	null	null	09.05.2017 00:00:00
8	6514b8ad8028c9f2cc2374ded245783f	9bdf08b4b3b52b5526ff42d37d47f222	delivered	16.05.2017 13:10:30	16.05.2017 13:22:11	22.05.2017 10:07:46	26.05.2017 12:55:51	07.06.2017 00:00:00
9	76c6e866289321a7c93b82b54852dc...	f54a9f0e6b351c431402b8461ea519...	delivered	23.01.2017 18:29:09	25.01.2017 02:50:47	26.01.2017 14:16:31	02.02.2017 14:08:10	06.03.2017 00:00:00
10	e69bfb5eb88e0ed6a785585b27e16...	31ad1d1b63eb9962463f764d4e6e0c...	delivered	29.07.2017 11:55:02	29.07.2017 12:05:32	10.08.2017 19:45:24	16.08.2017 17:14:30	23.08.2017 00:00:00

Рисунок 2.6. Приклад записів з таблиці замовлень.

Модель містить таблицю з інформацією про оплату товару, таблиця має назву payments_dataset і містить наступні поля:

1. Ідентифікатор замовлення;
2. Номер оплати по замовленню (існує можливість оплатити частину купонами, частину картою тощо);
3. Вид оплати (не визначено, купон, дебітова карта, кредитна карта, boleto – оплата готівкою в авторизованих місцях, один із популярних методів оплати в Бразилії);
4. Номер оплати під час розстрочення платежу;
5. Сума оплати.

Приклади записів з таблиці зображено на рисунку 2.7.

	order_id	payment_sequential	payment_type	payment_installments	payment_value
1	b81ef226f3fe1789b1e8b2acac839d17	1	credit_card	8	99,33
2	a9810da82917af2d9aefd1278f1dcfa0	1	credit_card	1	24,39
3	25e8ea4e93396b6fa0d3dd708e76c1...	1	credit_card	1	65,71
4	ba78997921bbcdc1373bb41e913ab...	1	credit_card	8	107,78
5	42fdf880ba16b47b59251dd489d444...	1	credit_card	2	128,45
6	298fcdf1f73eb413e4d26d01b25bc1cd	1	credit_card	2	96,12
7	771ee386b001f06208a7419e4fc1bb...	1	credit_card	1	81,16
8	3d7239c394a212faae122962df514ac7	1	credit_card	3	51,84
9	1f78449c87a54faf9e96e88ba1491fa9	1	credit_card	6	341,09
10	0573b5e23cbd798006520e1d5b4c6...	1	boleto	1	51,95
11	d88e0d5fa41661ce03cf6cf336527646	1	credit_card	8	188,73
12	2480f727e869fdeb397244a21b721b...	1	credit_card	1	141,9

Рисунок 2.7. Приклад записів з таблиці платежів.

Сайт має змогу накопичувати зворотній зв’язок покупців про товар, який вони купують. Зворотній зв’язок містить в собі оцінку по п’ятибальній системі і текстовий коментар. Відгуки зберігаються у таблиці reviews, яка має наступні поля:

1. Ідентифікатор відгуку;
2. Ідентифікатор замовлення;
3. Оцінка відгуку (за 5-ти бальною системою);
4. Заголовок відгуку;
5. Текст відгуку;
6. Дата і час створення відгуку;
7. Дата і час відповіді на відгук.

Приклади записів з таблиці зображено на рисунку 2.8.

	review_id	order_id	r...	review_comment_title	review_comment_message	review_creation_date	review_answer_timestamp
64	ca7402594d96f3231ee1bbae27da1e...	36782087b0d4f8cd6daa3408720062...	5			20.02.2018 00:00:00	21.02.2018 01:40:57
65	c0dbf9a325d2a2fece43873be47191e1	b324f4270850cbba23533b9a9d3434...	4		Г“tima loja para parceria: rГЎpidГssima, produtos bem embalados e de...	04.04.2018 00:00:00	06.04.2018 15:43:39
66	9924ac6169f2edc95ca3394e529f9580	a3ba7fa5266d62188949da3d6ccc38...	5			27.02.2018 00:00:00	02.03.2018 08:33:35
67	e12a8d938f3092e010627d001f91d1...	b3dca2e199b13ddabab4ed77d79ee...	5			03.06.2018 00:00:00	06.06.2018 12:41:24
68	109b5ce2dd11bb8460eff3b86da6fefc	25362fbf6aac4b01a28dee1e076acc26	5	Г“timo Produto!	Recomendo o vendedor....	17.08.2018 00:00:00	17.08.2018 21:47:08
69	6d06808638ec0701bccd70bc8d462c...	97d2f8fe76f2f253b8291e17b5383884	1		O produto nГЈo chegou no prazo estipulado e causou transtorno, pq pr...	09.12.2016 00:00:00	19.12.2016 14:10:43
70	750d4311e8ddb6cba549ee86f06f0cca	4b32a4cdfc1b80066a8ee3a4b7a2094f	5		Produto entregue como solicitado, e com muita brevidade. ParabГ©ns!	09.08.2017 00:00:00	09.08.2017 12:03:45
71	951a3591f5bcc0b0c558461c9bf49d8e	801524a8fbd37484e8d8c23448f113...	3			14.07.2018 00:00:00	14.07.2018 21:04:31
72	77ed4f94b3638d1f0b0a2dd781c11a...	5f8772295e1adb4e4832c988d6b631...	5			10.04.2018 00:00:00	12.04.2018 10:28:20
73	2a3eba09af2fbcd3a20580e62e7b7465	366df9c0b0d5d46f8afc476dc4ca7671	4			31.01.2018 00:00:00	01.02.2018 02:09:13
74	8b230a1373c6dc4bd867099fda1d70...	071251fe3b3493294536f03737a8a6...	3		Eu comprei duas unidades e sГł recebi uma e agora o que faГ§o?	09.11.2017 00:00:00	10.11.2017 10:38:37
75	4a1c0e7a111f33f2be53506db4887b...	0feb8ce954124d3e3d8338c573a9c5...	5			28.08.2018 00:00:00	29.08.2018 02:00:24
76	cb2fc3e5711b5ae85e0491ee18af63ed	34e6d418f368f8079ae152bc178bc66a	3		Produto bom, porГ©m o que veio para mim nГЃo condiz com a foto do...	31.01.2018 00:00:00	31.01.2018 23:29:21
77	60c714ed14cef913944a3147094a47...	9ac05114800f02bfaa783bd76842db...	1		Produto muito inferior, mal acabado.	03.07.2018 00:00:00	05.07.2018 19:03:52
78	b42460e68676135205af8b8c13baf6...	5b2e79428198e5ce482f9ec1bd1014...	5			29.07.2017 00:00:00	01.08.2017 18:43:09
79	505587fad6e7130f579fe9f02017ede4	64ff87188034c4f3dcd52e63db7d045f	4			14.06.2018 00:00:00	18.06.2018 22:53:04
80	c45811d9f90e22a81155b3a1e4a5c2...	491f193fc52075598871cfe6f19976d4	5	RECOMENDO SEMPRE!!	O kit mochila patrulha canina Г© lindo!! Meu netinho vai amar!! Obrig...	19.06.2018 00:00:00	19.06.2018 17:44:05
81	e29ab4a671400c41322e4855de633...	17b296b73ebefd6213299342b09f09...	5		Maravilha	09.08.2017 00:00:00	11.08.2017 23:28:46
82	c577faa817a66c6f1492728435f53cd0	97e11582eb8cdd7b83e35a4243d81f...	5			12.01.2018 00:00:00	14.01.2018 22:44:58

Рисунок 2.8. Приклад записів з таблиці відгуків.

Як було сказано раніше, кожна з таблиць має свої поля-ідентифікатори, які дозволяють поєднати всі таблиці в одну модель. На рисунку 2.9 зображено схему з назвою ключів-ідентифікаторів, яка дозволяє, наприклад, прив’язувати відгук до окремого продавця, при тому що рядок у таблиці відгуків не буде містити ідентифікатор продавця. Відгук буде прив’язано до продавця через додаткову таблицю.

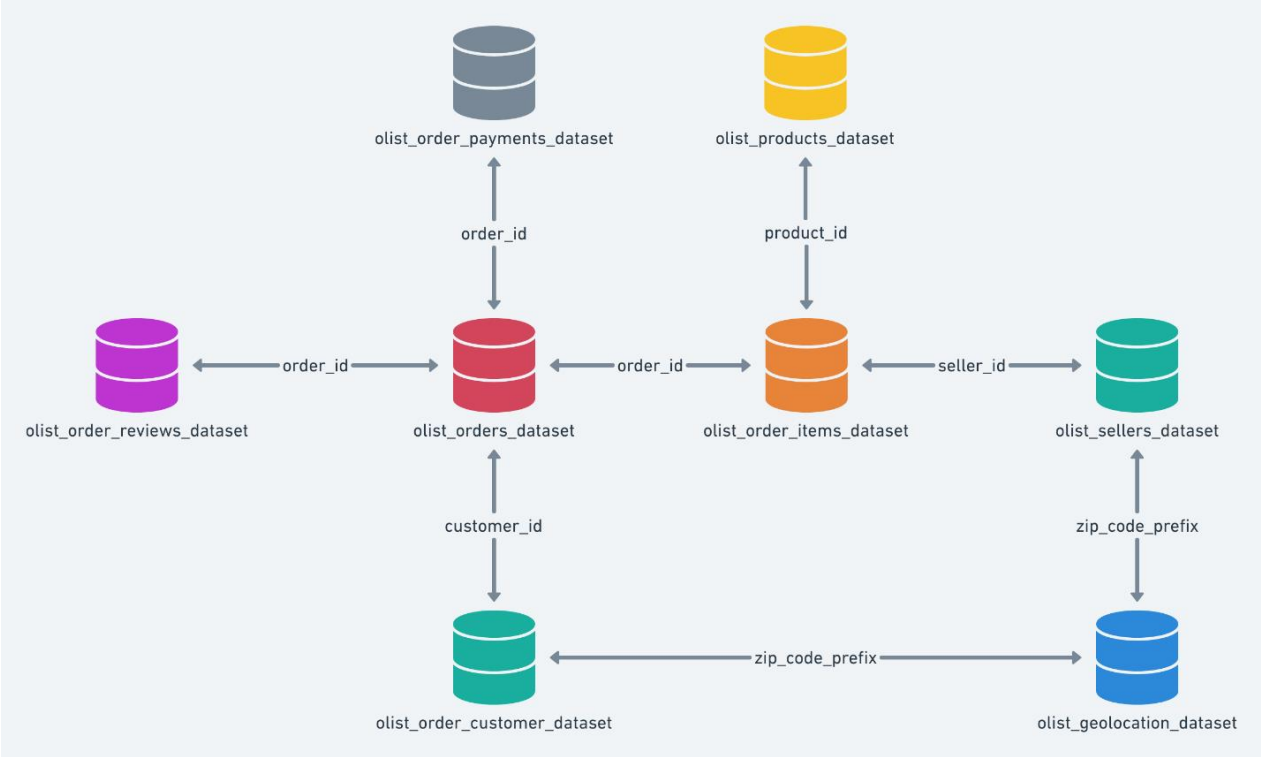

Рисунок 2.9. Схема зв’язків між таблицями у моделі даних.

2.3 Попередня обробка даних

Як було вказано, датасет представлено бразильскою компанією Olist. Тому логічно, що відгуки у таблиці будуть записані не англійською, а португальською мовою. Для того щоб розуміти сенс написаного у відгуках, було перекладено тексти відгуків на англійську за допомогою існуючих бібліотек програмування, а саме googletrans для мови програмування Python. Також для цього використовувалися бібліотеки NumPy та Pandas (лістинг

програми знаходиться у Додатку В). У таблиці 2.1 записано приклад перекладу на англійську мову, узятий з датасету.

Таблиця 2.1

Переклад з португальської на англійську

португальська	англійська
Recebi bem antes do prazo estipulado.	I received it well before the stipulated time.
ParabГ©ns lojas lannister adorei comprar pela Internet seguro e prГЎtico ParabГ©ns a todos feliz PGЎscoa	Congratulations Lannister stores I loved shopping online, safe and practical Congratulations to all Happy Easter
aparelho eficiente. no site a marca do aparelho esta impresso como 3desinfector e ao chegar esta com outro nome...atualizar com a marca correta uma vez que Г© o mesmo aparelho	efficient device. on the website the brand of the device is printed as 3disinfector and when it arrives it has another name...update with the correct brand since it is the same device
Mas um pouco ,travando...pelo valor ta Boa.	But a little slowing down...for the price it's good.
Vendedor confiГЎvel, produto ok e entrega antes do prazo.	Reliable seller, ok product and delivery on time.
GOSTARIA DE SABER O QUE HOUVE, SEMPRE RECEBI E ESSA COMPRA AGORA ME DECPCIONOU	I WOULD LIKE TO KNOW WHAT HAPPENED, I ALWAYS RECEIVED AND THIS PURCHASE NOW DISAPPOINTED ME
PГ©ssimo	Terrible
obrigado pela atenГ§ao amim dispensada	thanks for the attention given to me
A compra foi realizada facilmente. A entrega foi efetuada muito antes do prazo dado. O produto jГЎ comeГ§ou a ser usado e atГ© o presente, sem problemas.	The purchase was made easily. Delivery was made much earlier than the given deadline. The product has already started to be used and up to the present, no problem.
relГigio muito bonito e barato.	very nice and cheap watch.

Незрозумілі символи у мові оригіналу відповідають за букви зі специфічною вимовою, наприклад, quê, ésse тощо. Переклад на англійську мову підвищує нашу обізнаність про замовлення, які до цього ми могли розуміти лише через оцінку і список товарів. З цього моменту ми можемо досліджувати інші види помилок в датасеті, такі як: дублікати, невідповідність, загублені записи тощо.

Для того, щоб пов’язати таблиці між собою, запис однієї таблиці повинен однозначно відповідати іншому запису в іншій таблиці. Існує декілька видів зв’язків, а саме: 1 до 1, 1 до багатьох, багато до 1, багато до багатьох.

Якщо це таблиця клієнтів, то одному клієнту може відповідати кілька записів у таблиці замовлень – такий зв’язок буде називатися 1 до багатьох. Якщо дивитися у зворотньому порядку, то зв’язок таблиці замовлень до таблиці клієнтів буде - багато до 1. Якщо роздивлятися таблиці замовлень і відгуків, то логічно, о на 1 замовлення буде 1 відгук.

Отже почнемо дослідження з таблиці відгуків, першим ділом перевіримо наявність рядків з пустими ідентифікаторами. В нашому випадку таких рядків не знайшлось. Тому переходимо до наступного етапу – очищення від дублікатів. Цю задачу було поділено декілька етапів. Перерахуємо проблеми, які виникли під час пошуку дублікатів:

Проблема 1) дублікати відгуків у таблиці відгуків;

На рисунку 2.10 зображемо як виглядають дублікати відгуків.

	A	B	C	D	E
1	review_id	order_id	review_score	review_comment_title	review_comment_message
836	0ad57b9e6894d34814464772b6de54fe	d22f0b06e1519065d7eddb2ba6b79b63	1		
1927	0ad57b9e68d4d34814464772b6de54fe	75213e459cfe44a61c496c864e0a8c63	1		
3372	168b708c0e1d50f89c9737b1160121d6	92ef3140d9d92187a04b1bd643f32dc1	5		
4711	168b708c0e1d90f89c9737b1160121d6	fbb3296cbee389d9961a820220eb62a3	5		
6613	6ec93e77f444e0b1703740a69122e35d	e1fdc6e9d1ca132377e862593a7c0bd4	5		Seller's commitment to the customer
7213	6ec95e77f444e0b1703740a69122e35d	d8cbc79d1dd84f3c87cb2ca7663c930e	5		Seller's commitment to the customer
10317	8ee90ac383cf825bb7f4756130d4e74a	75d5d3d16567a27eefc5752aeb063072	5		I recommend
15711	8ee90ac383cf825bb7f4796130d4e74a	8e350e1e4254bd7c68913b98bde7d3a7	5		I recommend
23405	a20c24976e5130fdc8ac99310cd6b649b	2d7112fbe786c9c56050866c853e3a7f	2		The product arrived torn
40638	a20c24976e5130fdc8ac993ecd6b649b	061f2692f309a270bc04537e0a2508bb	2		The product arrived torn
48950	dc7a14d3e1964bfc11b484b841697096	8c6a3fab1ed272b02f23bb7dc9061d9a	4	I don't want to evaluate	
98578	dc7a14d3e1984bfc11b484b841697096	0544030711e50ec2cb6c15764d22891a	4	I don't want to evaluate	

Рисунок 2.10. Приклад дублікатів відгуків

Як можна побачити, ідентифікатори відгуку дублюються, але ідентифікатор замовлення різний. Якщо копнути глибше, а саме - дістатися до деталей замовлень, які дублюють відгуки, ми можемо побачити, що покупка виконується в один час, але ідентифікатор покупця різний. На рисунку 2.11 зображено результат виведення інформації про замовлення.

1	order_id	customer_id	order_status	order_purchase_timestamp
4230	8e350e1e4254bd7c68913b98bde7d3a7	f7060c3a8e6970cabb0af29d2864f4fd	delivered	19.05.2017 15:41
14460	75d5d3d16567a27eefc5752aeb063072	9909994172bb92df37805f2d2442faa1	delivered	19.05.2017 15:41
14804	fbb3296cbee389d9961a820220eb62a3	9dc80a62589e4f4f71f88cb98a4836bd	delivered	28.08.2017 15:30
18575	92ef3140d9d92187a04b1bd643f32dc1	35852763df46e976c158536d6b99b982	delivered	28.08.2017 15:30
38882	e1fdc6e9d1ca132377e862593a7c0bd4	d4587cf99c5521ba43b6acf080a396b4	delivered	18.09.2017 00:28
40084	d8cbc79d1dd84f3c87cb2ca7663c930e	258dfc4e0ec8a8bfb253914f9550a544	delivered	18.09.2017 00:28
42837	75213e459cfe44a61c496c864e0a8c63	9c11910f46b5a9a3d8366b09a0514c97	shipped	23.11.2017 22:44
52286	d22f0b06e1519065d7eddb2ba6b79b63	3c3335d4c88928de4ec0f90fae9d1507	delivered	23.11.2017 22:44
59331	2d7112fbe786c9c56050866c853e3a7f	2a0ae78342ed766c8dfd9f7b09e300d5	delivered	05.12.2017 19:03
81559	061f2692f309a270bc04537e0a2508bb	4479c24d672d4eb166c60188425da673	delivered	05.12.2017 19:03
89029	8c6a3fab1ed272b02f23bb7dc9061d9a	9084b0a94916b534ac6e626011092567	delivered	12.04.2018 17:36
96566	0544030711e50ec2cb6c15764d22891a	9eb585e566edc8b94a17ac5d6076d136	delivered	12.04.2018 17:36

Рисунок 2.11. Детальна інформація замовлень, відгуки яких мають дублікати.

Якщо зробити пошук за ідентифікатором замовлення у записі-дублікаті, ми можемо також знайти інші відгуки до цього замовлення, які записані під іншим ідентифікатором відгуку. У розглянутому прикладі текст відгуку однаковий між дублікатами-відгуками (див. рисунок 2.12).

1	review_id	order_id	review_score	review_comment_tit	review_comment_message
4713	8ee90ac383cf825bb7f4756130d4e74a	8e350e1e4254bd7c68913b98bde7d3a7	5		I recommend
7215	8ee90ac383cf825bb7f4756130d4e74a	75d5d3d16567a27eefc5752aeb063072	5		I recommend
33092	260b8badf10666253ac8a8eda99119f9	75d5d3d16567a27eefc5752aeb063072	1		I did not receive the complete order, 1 set of towels was missing. I bought 2 sets of Invoice C
74135	260b8badf10666253ac8a8eda99119f9	8e350e1e4254bd7c68913b98bde7d3a7	1		I did not receive the complete order, 1 set of towels was missing. I bought 2 sets of Invoice C
99226					

Рисунок 2.12. Пошук відгуків за ідентифікатором замовлення запису, яке є дублікатом.

Аналізуючи наведені приклади, можна припустити, що через недоліки у системі автоматизованого заповнення таблиць, ідентифікатор деяких покупців не зберігається. У результаті цього створюються різні ідентифікатори замовлень, які призводять до появлення дублікатів відгуків.

Розв’язання цієї проблеми відбувалося наступним чином - знаходячи дублікати відгуку, ми будемо шукати за ідентифікатором знайденого

дубліката усі записи. Після цього будемо замінювати у знайдених записах ідентифікатор замовлення на один обраний. Таку заміну будемо виконувати також у списку товарів і об’єднувати два і більше замовлень в одне.

На рисунку 2.12 зображено список товарів, які відповідають відгукам на рядках 10322,15724.

order_id	order_item_id	product_id	seller_id
75d5d3d16567a27eefc5752aeb063072	1	b90973ce80ba5d1fbc046fd8471acaef	41b39e28db005d9731d9d4
8e350e1e4254bd7c68913b98bde7d3a7	1	35afc973633aaeb6b877ff57b2793310	4a3ca9315b744ce9f8e9374
8e350e1e4254bd7c68913b98bde7d3a7	2	99a4788cb24856965c36a24e339b6058	4a3ca9315b744ce9f8e9374

Рисунок 2.12. Початковий вигляд запису.

Обираємо будь-який ідентифікатор замовлення, наприклад перший з набору двох. Після цього у списку товарів ми замінюємо значення order_id на обраний ідентифікатор замовлення. Теж саме робимо у таблиці з відгуками – у дублюючих рядках замінюємо значення order_id на те, яке ми обрали. У таблиці замовлень поєднуємо інформацію записів за наступним принципом: якщо це дата отримання – обираємо максимальне значення, якщо це дата замовлення – обираємо мінімальне значення. Тобто робимо так, щоб при поєднанні зберегти діапазон дат, в якому існує замовлення.

Ідентифікатор клієнта зберігаємо той, чий запис order_id буде присутнім в таблиці списків товарів, тому що існування ідентифікатора в таблиці замовлень не гарантує існування запису до нього в таблиці товарів до замовлення. В результаті ми отримаємо наступний запис (див. рисунок 2.13-15).

1	order_id	order_item_id	product_id	selle
47073	75d5d3d16567a27eefc5752aeb063072	1	b90973ce80ba5d1fbc046fd8471acaef	41b3
56566	75d5d3d16567a27eefc5752aeb063072	1	35afc973633aaeb6b877ff57b2793310	4a3c
56567	75d5d3d16567a27eefc5752aeb063072	2	99a4788cb24856965c36a24e339b6058	4a3c

Рисунок 2.13. Відредагований запис у таблиці списку товарів.

	A	B	C	
1	review_id	order_id	review_score	review_comment_tit
10322	8ee90ac383cf825bb7f4756130d4e74a	75d5d3d16567a27eefc5752aeb063072	5	
15724	8ee90ac383cf825bb7f4756130d4e74a	75d5d3d16567a27eefc5752aeb063072	5	

Рисунок 2.14. Відредагований запис у таблиці відгуків.

	A	B	C	D	E	F	G
1	order_id	customer_id	order_sta	order_purchase	order_approve	order_delivered	order_delivere
4230	75d5d3d16567a27eefc5752aeb063072	f7060c3a8e6970cabb0af29d2864f4fd	delivered	19.05.2017 15:41	20.05.2017 15:50	23.05.2017 15:48	31.05.2017 06:25
14460	75d5d3d16567a27eefc5752aeb063072	9909994172bb92df37805f2d2442faa1	delivered	19.05.2017 15:41	20.05.2017 15:50	23.05.2017 13:16	29.05.2017 14:13

Рисунок 2.15. Відредагований запис у таблиці замовлень.

Після виконання заміни у всіх можливих таблицях, ми можемо видалити опрацьований дублікат в полі review_id.

Розв’язування цієї проблеми відбувалося за допомогою додатку, написаного на мові програмування Python, серед бібліотек було використано NumPy, Pandas. Лістинг програми знаходиться у Додатку Г.

Проблема 2) посилання на втрачені ідентифікатори замовлення у таблиці відгуків;

Також проблемою стали записи відгуків, які посилаються на неіснуючі замовлення. За допомогою формули ВПР у табличному процесорі Excel, було зроблено пошук замовлення за ідентифікатором замовлення у таблиці замовлень, і було отримано наступний результат: якщо замовлення існує в іншій таблиці, в полі пошуку ідентифікатора буде посилання на замовлення, в іншому випадку – помилка (іншими словами запис #НД). Результат зображено на рисунку 2.16.

review_id	finded_match	order_id	review_score	review_comm	review_c
7bc2406110b926393aa56f80a40eba40	73fc7af87114b39712e6da79b0a377eb	73fc7af87114b39712e6da79b0a377eb	4		
80e641a11e56f04c1ad469d5645fdfde	a548910a1c6147796b98fdf73dbeba33	a548910a1c6147796b98fdf73dbeba33	5		
228ce5500dc1d8e020d8d1322874b6f0	f9e4b658b201a9f2ecdecbb34bed034b	f9e4b658b201a9f2ecdecbb34bed034b	5		
e64fb393e7b32834bb789ff8bb30750e	658677c97b385a9be170737859d3511b	658677c97b385a9be170737859d3511b	5		I received
f7c4243c7fe1938f181bec41a392bdeb	8e6bfb81e283fa7e4f11123a3fb894f1	8e6bfb81e283fa7e4f11123a3fb894f1	5		Congratu
15197aa66ff4d0650b5434f1b46cda19	b18dcdf73be66366873cd26c5724d1dc	b18dcdf73be66366873cd26c5724d1dc	1		
07f9bee5d1b850860defd761afa7ff16	e48aa0d2dcec3a2e87348811bcfdf22b	e48aa0d2dcec3a2e87348811bcfdf22b	5		
7c6400515c67679fbee952a7525281ef	c31a859e34e3adac22f376954e19b39d	c31a859e34e3adac22f376954e19b39d	5		
a3f6f7f6f433de0aefbb97da197c554c	9c214ac970e84273583ab523dfafd09b	9c214ac970e84273583ab523dfafd09b	5		
8670d52e15e00043ae7de4c01cc2fe06	b9bf720beb4ab3728760088589c62129	b9bf720beb4ab3728760088589c62129	4	I recommend	efficient
c9cfd2d5ab5911836ababae136c3a10c	cdf9aa68e72324eeb25c7de974696ee2	cdf9aa68e72324eeb25c7de974696ee2	5		

Рисунок 2.16. Результат успішного пошуку за ідентифікатором.

Якщо замовлення не існує, то в результаті буде виведено помилку посилання.

review_id	finded_match	order_id	review_score	review_comm	review_comment_message	rev
6280ca6a9e8c3a56b668ec8c5396caaf	#Н/Д	534a0fbe91a806a1dc6bf09b497e3d6f	1		The scheduled days passed.	
81648dc10c6e3f763cdc0dbcd073d055	#Н/Д	6e7e891760708803a2f4203981f2d69d	1			
90018c08b2ca29dddcf499cfa5ec54d8	#Н/Д	4b460251c3d2ba44dd61b0b6c3410ac6	3		Good Guys, the Lannister attendant was very atten	
1a8dab4afb8c80d8d257fe0156d764b1	#Н/Д	f4a8cf5dcabdbe8dd7e65e1674db3015	1		I simply didn't receive my product and I don't have	
8f78ff92924bc0a30d0cd9b0b1adae64	#Н/Д	d49363a0cc2a1915a5a11f85ea08ea48	1			
cab29f1f7496f39211eae61c4cf92a0e	#Н/Д	309688f286ca9fb39735664ba1e37b29	2			
b741b5f3776a645b8d3df3300b2f484a	#Н/Д	e66e29f65af321ea428495d66066e809	1		They sold me a product they didn't have to deliver	
161f01713c98308528107791c17bed8f	#Н/Д	6d928ae5792f61a71bb3e3e528bf4a28	1		I HOPE I NEVER BUY AGAIN AND HAVE TO ANSWER '	
ee4e8f9fd8213819903417595a9f9c04	#Н/Д	b07abc8b9acaf00e79b4657419f469f3	1		When I went to buy the product, it said it was in stc	
97f53b172d8ccec5dfb07184b35f3691	#Н/Д	42202323e8e5319c107e361d55ad0de6	1			
75ed80f68d731eeb23e6b508e5cb9fbd	#Н/Д	ccbd198eb55e22f3b2240d1605961c5f	2		I liked it but it came with the leaf support spring br	

Рисунок 2.17. Результат невдалого пошуку за ідентифікатором.

Записи, які не вдається знайти в таблиці замовлень було вирішено видалити, тому що не можливо відстежити покупця і продавця, які фігурують в замовлені.

Розв’язування цієї проблеми відбувалося за допомогою PowerQuery, а саме було спочатку використано пошук за ідентифікатором, а після цього видалено рядки з помилкою у вказаному стовпчику.

Проблема 3) двічі оцінене замовлення;

Подібна проблема зустрічалась раніше, але вона стосувалась повторів записів у полі відгуків. В цьому випадку проблема полягає в тому, що замовник залишає відгук через деякий час після першого. Наприклад, перший відгук стосувався більше перших емоцій від товару (зовнішній вигляд, швидкість доставки), а другий відгук стосується якості товару (чи не пошкодився товар під час експлуатації або відгук щодо його роботи).

На рисунку 2.18 зображено приклад таких відгуків.

	A	B	C	D	E	F
1	review_id	order_id	review_scor	review_comr	review_comment_message	review_creation_date
1990	03a6a25db577d0689440933055111	2acfdc5131ff2cf4433e668454c9784c	5		Very good! I really liked it! Fabric Great! Met My Expectations! рu'Џрu'Џрu'Џ Super R	15.12.2017 00:00
5769	ec3817f305c65d0e88b6d5a00253e	2acfdc5131ff2cf4433e668454c9784c	5		Great product! Arrived ahead of schedule!!!	08.12.2017 00:00
9904	64b798072e372c1c60cc8044df9eb0	bd859dd7a6a1b8c1df9fcb75c3604eaf	2		regular productbut the prints are beautiful	29.03.2018 00:00
11809	2af839ae66a65959d6ae07775d0e7	bd859dd7a6a1b8c1df9fcb75c3604eaf	3		bom	28.03.2018 00:00
24372	1a6308ac2462f451538e9c28587142	f9fe6e11e33871e71f921d5051e72bb4	1		I bought 3 paintings and only one arrived, the guitar. Poor quality frame, very low res	18.03.2018 00:00
54778	7b36905f84d4f18f84aa570b0b54f0	f9fe6e11e33871e71f921d5051e72bb4	2		The frames are of poor quality.The resolution of the images leaves much to be desire	21.03.2018 00:00

Рисунок 2.18. Приклад відгуків до одного і того самого замовлення.

Щоб наша модель мала уявлення про замовлення, як про одне ціле, а не декілька різних відгуків, було прийнято рішення про поєднання відгуків під один ідентифікатор з поєднанням коментарів і розрахуванням середнього арифметичного оцінки замовлення.

Результат такого рішення зображено на рисунку 2.19.

review_id	order_id	review_score	review_comment_title	review_comment_message	review_creation_date	review_answer_timestamp
ec3817f305c65d0e88b6d5a00:	2acfdc5131ff2cf4433e668454c	5		Great product! Arrived ahead of schedule!!!;Very good! I really liked it! Fabric Great! Met My Expectations! рu'Uрu'Uрu'Ц Super Recommend! рu'К	08.12.2017	10.12.2017
64b798072e372c1c60cc8044df	bd859dd7a6a1b8c1df9fcb75c3	2		regular product but the prints are beautiful; bom	29.03.2018	04.04.2018
1a6308ac2462f451538e9c2858	f9fe6e11e33871e71f921d5051	1		I bought 3 paintings and only one arrived, the guitar. Poor quality frame, very low resolution printing. ; The frames are of poor quality. The resolution of the images leaves much to be desired.	18.03.2018	19.03.2018

Рисунок 2.19. Приклад поєднання відгуків.

Таким чином ми отримали один запис відгуку на одне замовлення.

Розв’язування цієї проблеми відбувалося за допомогою додатку, написаного на мові програмування Python, серед бібліотек було використано NumPy, Pandas. Лістинг програми знаходиться у Додатку Ґ.

Також вагомим недоліком датасету є те, що ми не маємо точного уявлення про товар, який описується у відгуку. Рядок запису у таблиці відгуків не має посилання на ідентифікатор товару або продавця, на чий товар було створенно відгук. Поєднання відгуків виконується через ідентифікатор замовлення. Якщо в замовленні буде два різних товари з двома різними відгуками, то, користуючись такими даними, ми не будемо мати уявлення ні про продавця, ні про товар, який оцінюється. Можливо людина, яка буде читати відгук і бачити список товарів, зможе зрозуміти про який саме товар йдеться мова у тексті, проте читати масив з таких даних не є раціональним рішення - особливо якщо дані накопичувалися роками. З часом дані про замовлення стають в нагоді для компанії, тому що з них можна дістати insight.

Insight – це цінна інформація, отримана за допомогою аналізу, як під час Розвідувального аналізу даних (EDA), так і під час тренування моделі машинного навчання (ML). Такі відкриття, отриманні за допомогою

аналітики, можуть бути потужним стимулятором розвитку бізнесу, оскільки дозволяють виявляти нові можливості.

Наприклад, при моделюванні репутаційної оцінки продавця на сайті на основі відгуків покупців, можна дізнатися популярні слова, які використовують покупці у тексті. За цими даними ми можемо знайти, що більше впливає на репутацію продавця, а для керівництва ця інформація допоможе вносити корективи в бізнес-процеси компанії.

2.4 Створення корпусу відгуків

Після завершення попередньої обробки датасету, під час якої вирішено всі неточності, можна приступати до завершального етапу створення корпусу відгуків для навчання і тестування моделі обробки відгуків. Отже, видалимо усі пусті відгуки з нашого датасету, і тепер ми можемо використовувати цей набір даних для ембедінгу токенів, з яких складаються наші відгуки.

У нашому датасеті відгуки представлені оцінками п’ятибальної системи. Щоб розділити вибірку на позитивні і негативні відгуки, було прийнято рішення позначити позитивними всі відгуки, оцінка яких більше 3. Тобто негативні відгуки мають оцінку 1,2,3 за п’ятибальною системою. Таким розподілом ми отримали 26235 позитивних відгуків та 13782 негативних.

У додатку, який розробляється як практична реалізація задачі, використовується метод Word2Vec. Він формує простір розміром $100 \times k$, де $k -$ кількість токенів у словнику (у нашому випадку це 517517 слів). Кожне слово представляється за принципом, який було описано у розділі 1.5.

Щоб розподілити токени у словнику, треба використати одну з двох задач: пошук відсутнього слова за наявним контекстом (CBoW) або пошук контексту за наявним словом (k-skip gram). Розмір ковзного вікна у нашому випадку буде дорівнювати 10. Також, щоб покращити результат роботи, було вирішено додати неправильні k-gram-и (*negative sampling*) – це допоможе навчити модель відрізняти шум у словах. Іншим способом покращення моделі, замість *negative sampling*, виступила модифікація *«hierarchical softmax»*. Загалом ми отримаємо чотири варіанти словника, які будуть конкурувати між собою за право бути основою для моделі класифікатора.

Реалізація побудови моделі ембедінгів відбувалося за допомогою додатку, написаного на мові програмування Python, серед бібліотек було використано NumPy, Pandas, Gensim,nltk. Лістинг програми знаходиться у

Додатку Д. Відповідно до налаштувань створювалась окремий словник, який потім зберігався в окремий файл і імпортувався в програму при потребі.

Порівняння корпусів буде відбуватися за перевіркою слів на схожість синонімів, антонімів і непов’язаних між собою. У таблиці 2.2 записано результат порівнянь слів за чотирма моделями. У записах, де виділено зеленим – правильне рішення або більше схоже на правду, ніж в інших моделях , якщо червоне – неправильно або погано розрізняє два слова. Наприклад, слова «правильно» і «коробка» несумісні між собою, але відмічають як сумісні, а слова «швидко» і «повільно», «добре» і «погано» мають приблизно однакову схожість, тому що використовуються в однакових контекстах. Приблизно таким самим чином пояснюється близкість слів «швидко» і «доставка», «погано» і «якість» - ці слова розташовані близько один до одного і використовуються в одному контексті. А ось вже слова «підробка» і «доставка», «швидко» і «спорядження» несумісні між собою, і навіть можуть не використовуватися в одному контексті.

Таблиця 2.2

Результат порівняння слів за схожість за чотирма моделями словників

Слово1	Слово2	sgram_ng_s	sgram_hs	cbow_ng_s	cbow_hs
good' (добре)	bad'(погано)	0,66	0,46	0,56	0,55
fast'(швидко)	slow'(повільно)	0,67	0,29	0,48	0,29
fast'(швидко)	long'(довго)	0,49	0,09	0,12	0,14
true'(правильно)	box'(коробка)	0,54	0,01	0,26	0,07
bad'(погано)	quality'(якість)	0,47	0,12	0,16	0,01
fast'(швидко)	delivery'(доставка)	0,58	0,16	0,12	0,34
order'(замовлення)	item'(предмет)	0,75	0,44	0,61	0,36
box'(коробка)	delivery'(доставка)	0,32	0,22	0,1	0,32
counterfeit'(підробка)	delivery'(доставка)	0,4	0,44	0,22	0,66
counterfeit'(підробка)	fake'(підробка)	0,86	0,57	0,92	0,63
fast'(швидко)	equipment'(спорядження)	0,52	0,36	0,15	0,82

item'(предмет)	equipment'(спорядження)	0,67	0,08	0,39	0,1
good'(добре)	nice'(добре)	0,76	0,49	0,7	0,67
delivery'(доставка)	post'(пошта)	0,53	0,17	0,28	0,02

Позначення моделей у таблиці: sgram_ng_s – skip gram з negative sampling; sgram_hs – skip gram з hierarchical softmax; cbow_ng_s – CBoW з negative sampling; cbow_hs – CBoW з hierarchical softmax.

За результатами порівняння можна виділити дві моделі словника: sgram_ng_s і cbow_ng_s, тобто моделі з використанням negative sampling. Наочно можна побачити, що саме ці моделі наближаються до зеленого кольору, тому ми можемо ставити їх пріоритетними під час побудови класифікатора.

2.5 Побудова класифікатора

Після побудови словника на основі наших відгуків, ми можемо перейти до самої задачі класифікації. У розділі 1.4. було описано моделі, які можна навчити класифікувати відгуки за тональністю тексту. Розробка додатку відбувалась на мові програмування Python, а реалізація моделей була побудована на вже готових системах, які знаходяться у бібліотеці для машинного навчання scikit-learn. Разом із цим активно використовувалась бібліотека Pandas, NumPy, matplotlib. Готові рішення дають змогу швидше виконувати обчислення, тому що при розробці бібліотек, враховуються особливості роботи з мовами програмування і комп’ютером загалом. Тому таке рішення цілком виправдано, спираючись на те який масив даних оброблюється комп’ютером і скільки часу займає навчання моделей.

Перед навчанням моделей розділимо вибірку на навчальну і тестову. Принцип полягає в тому, щоб навчити модель на одній вибірці, а перевірку якості виконувати на даних, які модель ще не бачила. Навчальній вибірці

призначимо 70 відсотків початкової вибірки, залишок відійде тестовій вибірці, на якій будемо перевіряти якість моделі.

Першим на черзі класифікаторів буде метод Наївного Байєса, який розраховує ймовірність гіпотези B, при наявності виконаної події A. У нашому випадку гіпотеза це позитивний настрій тексту, при наявності вектору тексту відгука. Реалізація виконувалась за допомогою класу GaussianNaiveBayes з бібліотеки від scikit-learn (sklearn).

Характеристика даного методу в цілому негативна, тому що на 2 різних словниках метод показує поганий результат, в особливості великий показник False-negative rate, який відповідає за відсоток неправильного охарактеризованих позитивних відгуків. На рисунку 2.21 і 2.23 зображено матрицю плутанини для класифікатора Наївного Байєса за кожним словником.

Матриця плутанини – матриця результату класифікації, де можна подивитися кількість True Positive, False Positive, True Negative, False Negative значень.

Особливо виділяється кількість помилок першого роду (False Negative), при цьому низький показник помилки другого роду (False Positive). Такий, можна сказати, ідеальний відсоток (4 відсотки тестової вибірки потрапили у другу помилку) не можна вважати за ідеальний результат, коли майже половина вибірки потрапило у помилку першого роду. Фактично модель не навчилась відрізняти позитивні відгуки і сприймає все як негативні. На рисунку 2.21 показано матрицю плутанини.

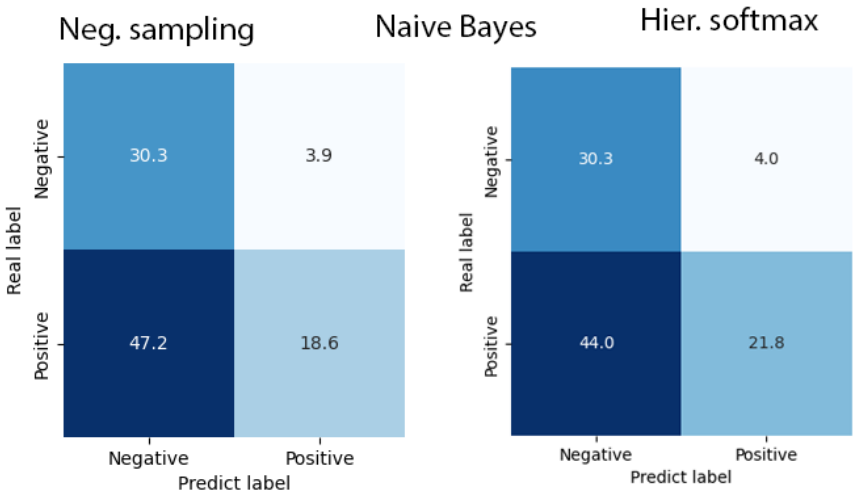

Рисунок 2.21 Матриця плутанини для моделі Наївного Баєса.

Переходимо до наступних показників якості (див. рисунок 2.27, 2.28) - F1-score знаходиться в інтервалі 0,42-0,48, що значить гармонічну середню точність методу менше половини. ROC-AUC близько до 0,6 тобто площина під кривою дорівнює трохи більше половини графіку, а це значить, що майже нічим не відрізняється від випадкової класифікації.

Друга модель була побудована за методом опорних векторів. Для пошуку розв’язку кращого, ніж стандартний, було прийнято рішення виконати випадковий пошук гіперпараметрів методом випадкової. Для кожного параметру було створено масив з можливих значень, після цього ці значення комбінувалися і завантажувалися у модель для виконання тренувань. Було використано 3 гіперпараметри з 5 варіантами значень в середньому для кожного. Тобто ми отримали 5^3 або 125 варіантів налаштувань системи. До цього додається ще ітерації під час навчання, їх кількість варіюється і деякі спроби оптимізації можуть займати години. Тому було прийнято рішення скоротити кількість словників до двох: skip gram negative sampling та skip gram hierarchical softmax. На рисунку 2.22 показано матрицю плутанини.

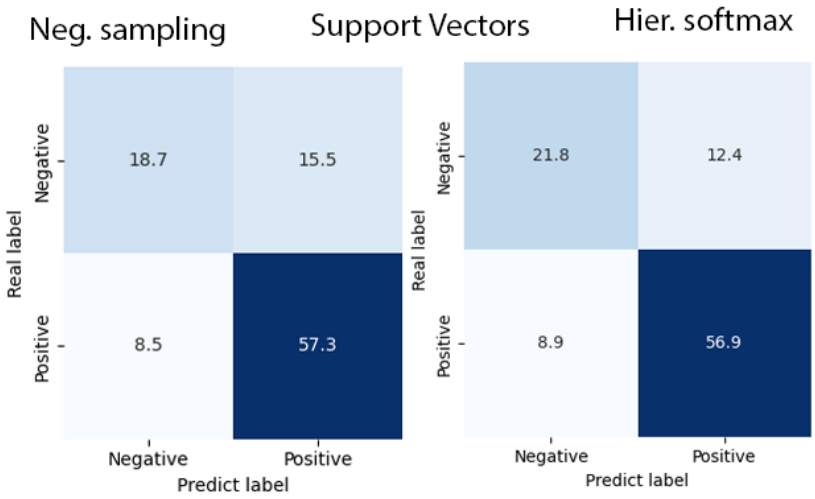

Рисунок 2.22 Матриця плутанини для моделі Опорних векторів.

Отже, найкраще себе показала ця модель на словнику skip gram hier. softmax. У нашому випадку, краще більше допускати помилок у False-Negative випадках, тому що негативний відгук (які можуть бути розпізнані позитивними, відповідно False-Positive rate) несе в собі більше інформації і саме він буде основним рушієм змін у бізнесі: поганий відгук – сигнал появи проблеми. Порівнюючи результат між словниками, мінливість показника FPR складає 3,1% на проміжку [12.4;15.5].

При цьому крива ROC (див. рисунок 3.27, 3.28) вигнута в лівий кут графіку, що свідчить про кращий прогноз, ніж випадковий. Показник F1-score дорівнює 0,83 або 83% - це середня точність моделі. Значення ROC-AUC дорівнює 0,73 або 73% на обох словниках в середньому.

Третя модель була побудована за методом K-найближчих сусідів. Алгоритм досить простий для розуміння і був описаний у розділі 2.3. Як і в минулому методі, будемо налаштовувати 4 гіперпараметри на словнику skip gram hier. softmax. У кожному параметрі приблизно по 3 варіанти відповіді, тому всього комбінацій параметрів буде 3^4 або 81. Мінливість признаку False Positive дорівнює 7% , а проміжок значень дорівнює [10.8;17.8] при незмінних параметрах, тобто тих, які ми знайшли на словнику skip gram hier. softmax. На рисунку 2.23 показано матрицю плутанини.

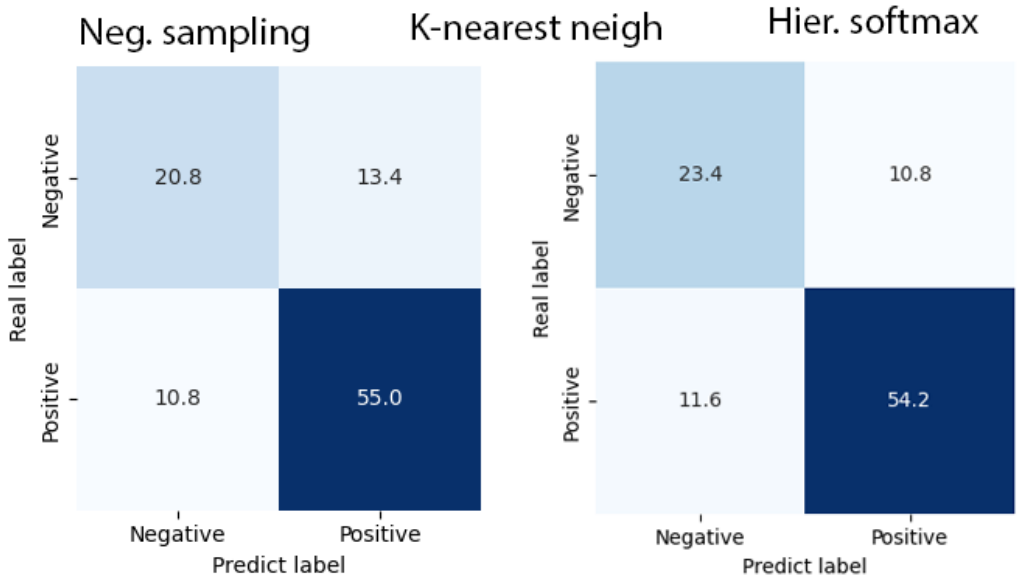

Рисунок 2.23 Матриця плутанини для моделі найближчих сусідів.

При цьому крива ROC показово вигнута в лівий бік, що свідчить про менший ріст показника FPR при зростанні TPR. Показник F1-score дорівнює 0,82 або 82% - це середня точність моделі. Значення ROC-AUC дорівнює 0,73 або 73% координатної площини в середньому за двома моделями.

Четверта модель була побудована за методом Екстремальний Градієнтний Бустинг, який запрограмовано в окрему бібліотеку, назва бібліотеки - XGBoost. Під час опису методу Бустингу згадувалося, що ця імплементація методу є найпоширенішою серед сучасних рішень регресії/класифікації методами машинного навчання. Налаштування гіперпараметрів проводилося все за тією самою технікою, змінювалися 4 параметри, які мали по 3 варіанти значень.

Модель показує стабільно задовільний результат без залежності від словника. Особливістю цього варіанта класифікатора є низький рівень, відносно інших, показника False-positive rate, який показує кількість відгуків, які розпізнано як позитивні, хоча вони є негативнимии. Найкращий результат получили на словнику skip gram hier. Softmax. Показник Specificity (TNR) буде дорівнювати 0,73 або 73% всіх негативних відгуків було правильно

ідентифіковано, і Recall(TPR) – 0,8 або 80%. На рисунку 2.24 показано матрицю плутанини.

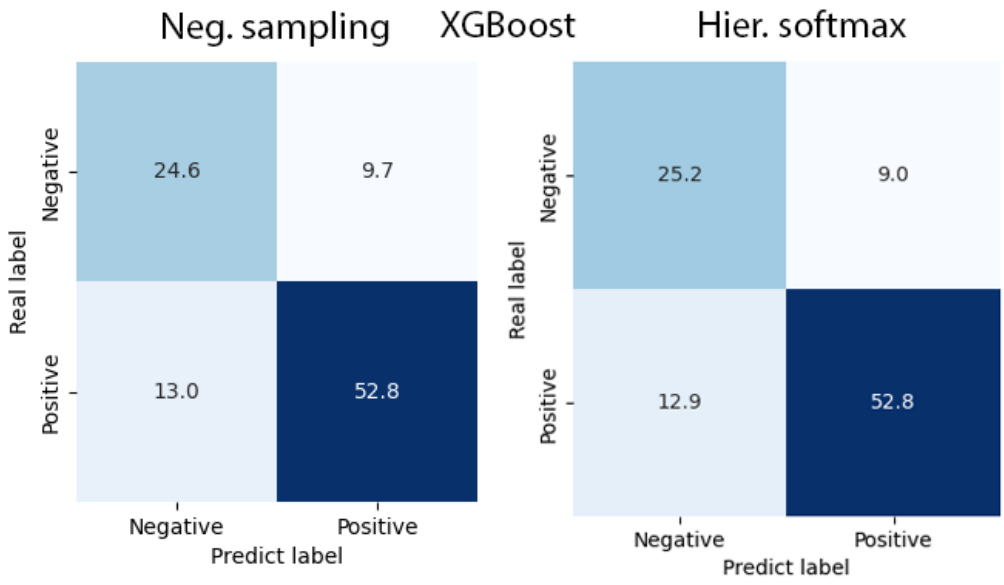

Рисунок 2.24 Матриця плутанини для моделі Екстримального Градієнтного бустингу.

Далі розглянемо інші показники якості, такі як: F1-score, ROC крива, ROC-AUC. На рисунках 3.27, 3.28 можна переглянути ці показники для кожного словника. ROC-AUC лежить на відрізку [0,76;0,77] по всім моделям. Значення F1-score лежить на відрізку [0,82-0,83] тобто приблизно 0,83 або 83% значень було правильно класифіковано.

Переходимо до іншого виду розв’язку деревами рішень – Random Forest Classification. Відмінність Random Forest і XGBoost в тому, що бустинг покращує модель послідовно, а бегінг (у випадку Random Forest) – паралельно. Налаштування гіперпараметрів виконувалися за допомогою рандомайзеру. Кількість параметрів дорівнює 6, а середнє значення у варіантах значень дорівнює 5. Після завершення випадкового навчання було отримано наступні результати: за матрицею плутанини можна побачити, що класифікатор добре розрізняє негативні відгуки, але на словнику hier. Softmax 1/3 частина негативних відгуків було прийнято за позитивні, а на

словнику negative sampling це значення піднімається майже до ½. На рисунку 3.25 зображено матрицю плутанини для цієї моделі.

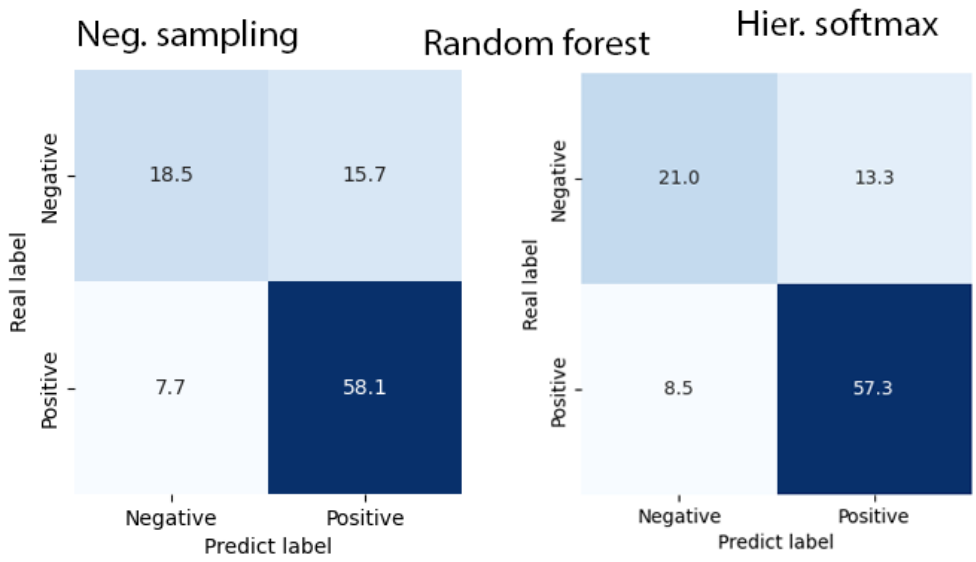

Рисунок 2.25 Матриця плутанини для моделі Випадкового лісу.

Взглянемо на ROC криву на рисунках 2.27, 2.28. Крива вигнута в лівий кут координатної площини, що свідчить про менше зростання FPR(False Positive Rate), ніж TPR (True Positive Rate). Перейдемо до інших показників якості. Значення F1-score дорівнює 0,84, що свідчить про добру якість прогнозування, ROC-AUC - 0,72 або 72% площини знаходиться під кривою, що свідчить про те, що модель обробляє дані краще, ніж випадковий класифікатор, у якого площа дорівнює 0,5 або 50%.

Наступним і останнім класифікатором буде Нейронна мережа. Алгоритм був описаний у розділі 1.4. Як і в минулих методах, ми будемо налаштовувати 4 гіперпараметри на словнику skip gram negative sampling та skip gram hier. softmax. У кожному параметрі приблизно по 4 варіанти відповіді, тому всього комбінацій параметрів буде 4^4 або 256. Ця модель показала конкурентний результат за показником FPR (False Positive Rate) – 9,4%. Такий результат було досягнуто на словнику skip gram negative sampling, і він може позмагатися за право бути обраним кращим

класифікатором. Тобто майже ¼ всіх негативних відгуків було неправильно помічено. На рисунку 2.27 і 2.28 можна подивитися ROC криву для цієї моделі. Вона вигнута в лівий кут площини, що свідчить про гарну роботу. Інші показники, такі F1 показали значення 0,82 або 82% відсотки точності, значень площі під кривою дорівнює 75%.

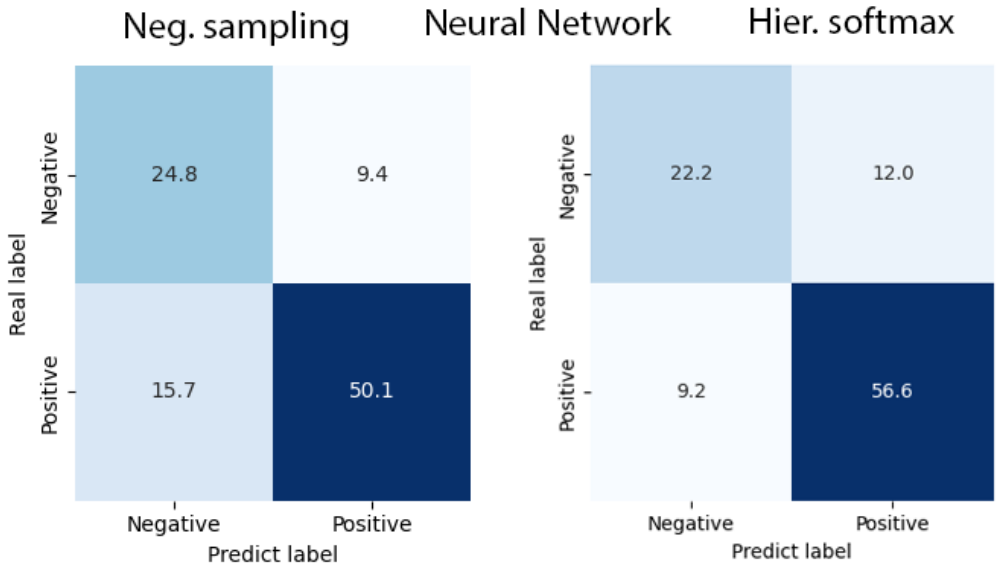

Рисунок 2.26. Матриця плутанини для моделі Нейронної мережі.

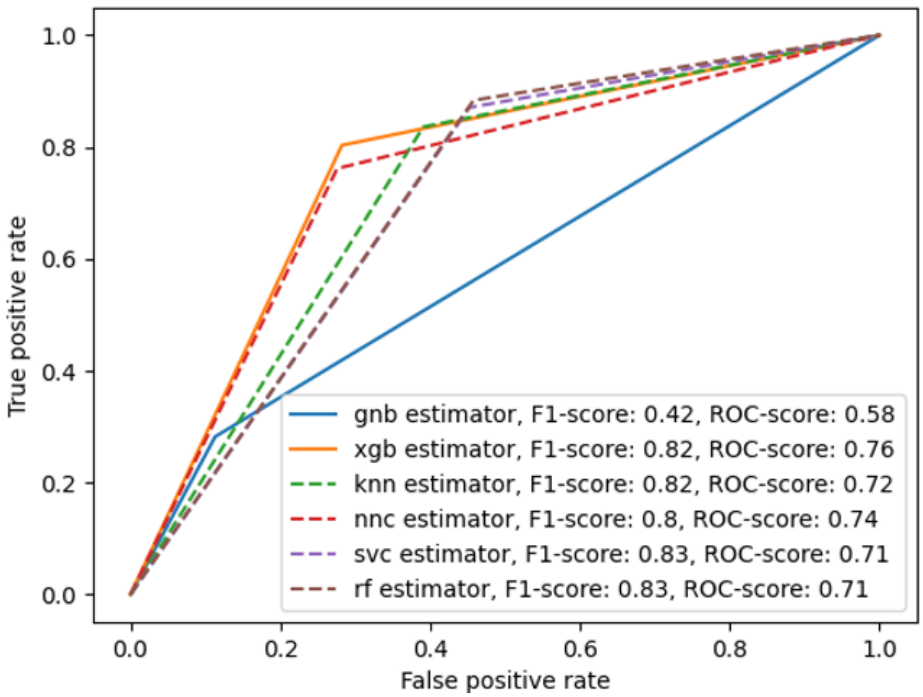

Рисунок 2.27. ROC крива для моделей побудованих на словнику skip gram negative sampling.

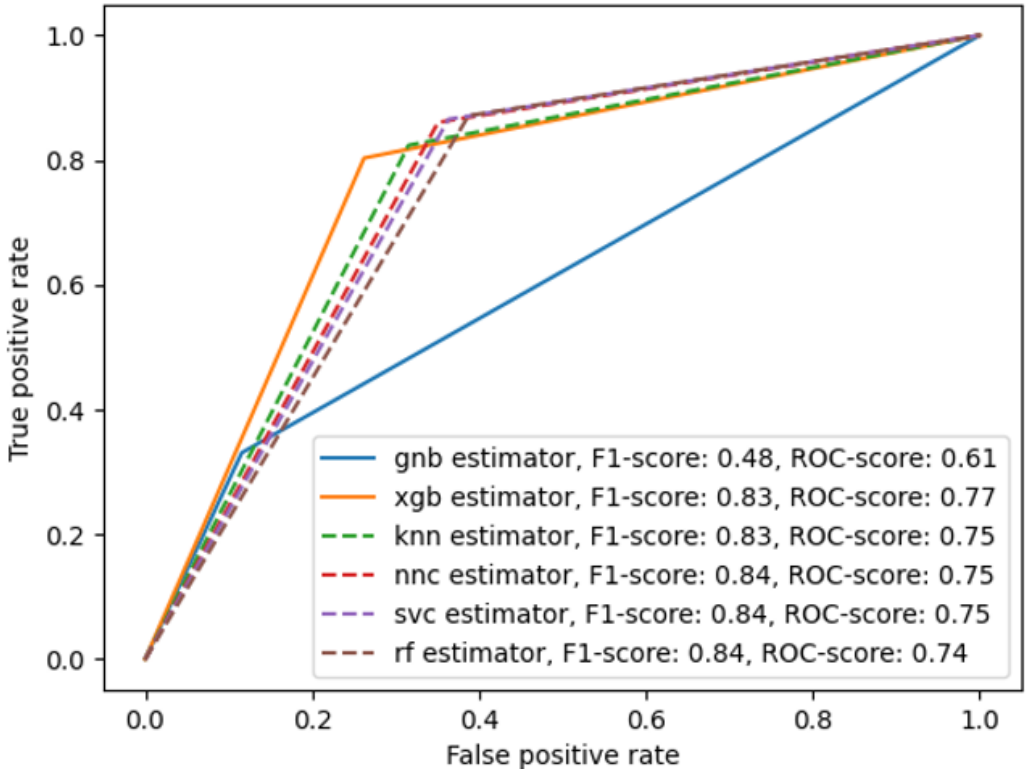

Рисунок 2.28. ROC крива для моделей побудованих на словнику skip gram hierarchical softmax.

Реалізація побудови класифікаторів і їх навчання відбувалося за допомогою додатку, написаного на мові програмування Python, серед бібліотек було використано NumPy, Pandas, Sklearn (sckilit-learn), XGBoost. Кожну модель поміщено в окремий Пайплайн, який дозволяє автоматизувати однакові процеси за допомогою однієї структури. Лістинг програми знаходиться у Додатку Е.

2.6 Вибір моделі

Описавши кожну модель класифікації і результат її роботи, ми маємо уявлення про кожен варіант вирішення задачі класифікації споживачів за відгуком товару. Сконсолідуємо вище вказані результати і додамо до цього дані про витрати часу.

Так як більшу користь для нас, як для бізнеса, становлять негативні відгуки, тому що, як було вказано, вони допомагають виявити проблеми і

становляться рушієм змін – ми будемо орієнтуватися на модель, яка має найменший показник помилок другого роду (False Positive Rate). Розглянемо матрицю плутанини для кожної моделі побудованої за словником skip gram negative sampling зображена на рисунку 2.29.

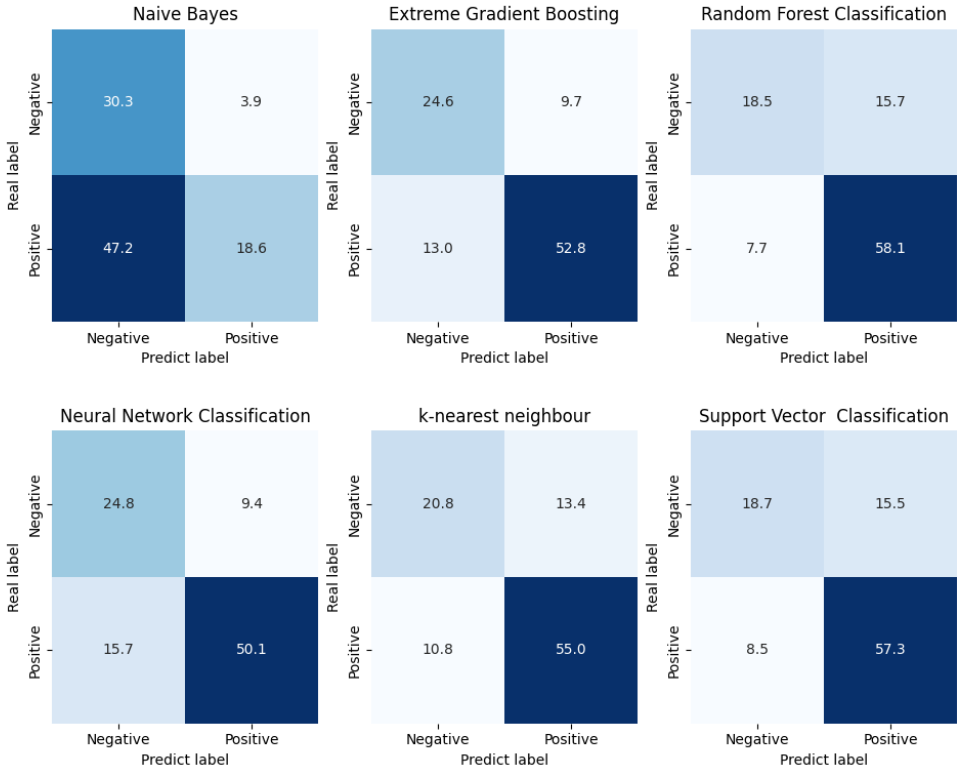

Рисунок 2.29. Матриця плутанини для моделей на skip gram negative sampling.

Порівнюючи показники, можна відмітити низький відсоток помилок другого роду у моделях XGBoost та Neural Network. Також нас, як розробників, може заінтересувати модель K-nearest neighbor, але в ній перевага надається помилкам першого роду (False Negative).

Обравши три моделі з першого словника, перейдемо до другого. Порівняємо показники між моделями, виведемо найцікавіші, тобто з мінімальними помилками у класифікації. На рисунку 2.30 зображено

матрицю плутанини для кожної моделі, яку побудовано за словником skip gram hierarchical softmax.

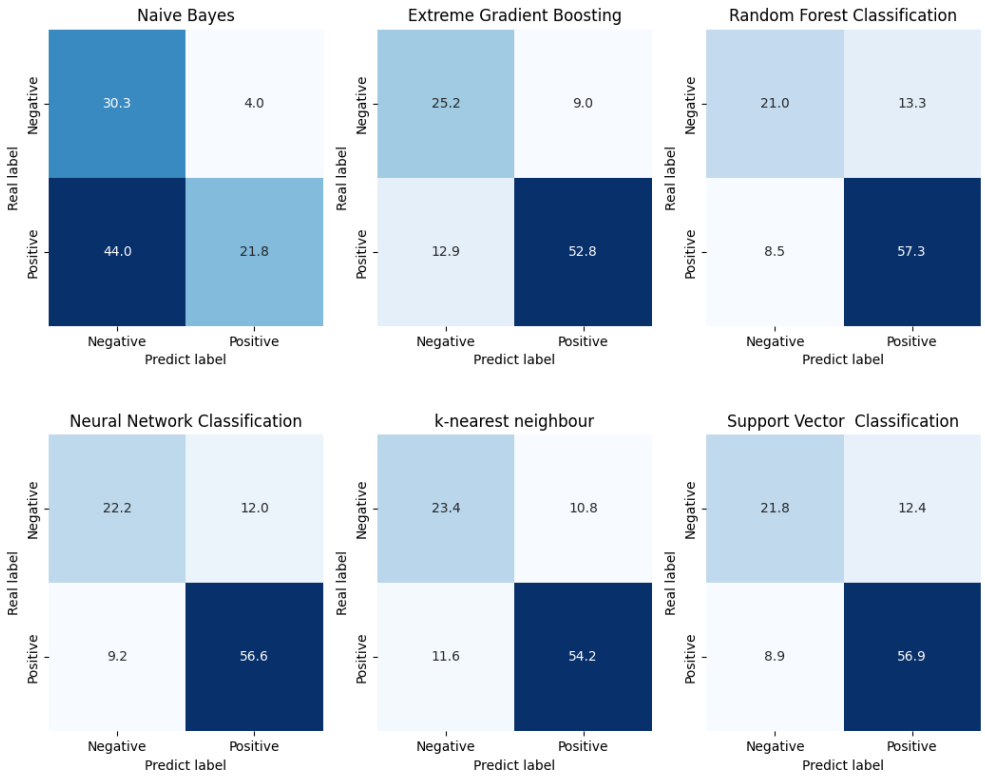

Рисунок 2.30. Матриця плутанини для моделей на skip gram hierarchical softmax.

Порівнюючи показники, можна відмітити низький відсоток помилок другого роду у моделях XGBoost та K-nearest neighbor. Також нас, як розробників, може заінтересувати моделі Random Forest, Neural Network та Support Vector, але в них перевага надається помилкам першого роду (False Negative).

Як можна було побачити на обох рисунках матриць, модель Naive Bayes має низький рівень False Positive класифікацій. Проте такий низький рівень нівелюється тим, що помилка першого роду (False Negative) сягає більше 40% всієї тестової вибірки. Виходячи з цього можна вважати, що модель

взагалі не вміє відрізняти позитивні відгуки і класифікує більшість коментарів як негативні.

Зобразимо вищевказані висновки за допомогою гістограми з сумою помилок першого і другого роду (див. рисунок 2.31).

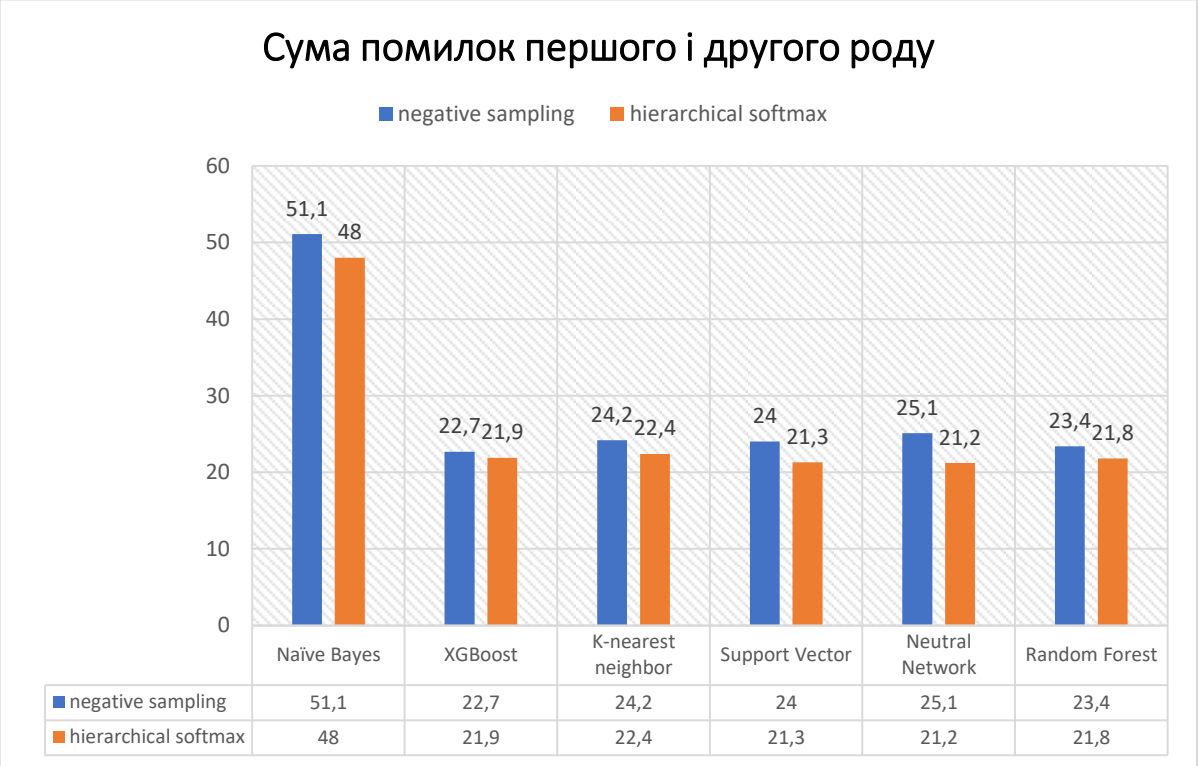

Рисунок 2.31. Гістограма помилок першого і другого роду.

Як можна побачити, модель XGBoost має найменшу суму помилок, при цьому, як ми побачили, частка помилок першого роду складає більшість цієї суми. Наступною інтересуючою моделлю була Neural Network, яка побудована на словнику negative sampling, та K-nearest neighbor, яка побудована на словнику hierarchical softmax.

Наступним параметром для остаточного вибору буде час, який витрачається на роботу моделі. На жаль, у цьому параметрі майже неможливо відстежити час, який витрачається на знаходження оптимальних значень гіперпараметрів, що є також складовою навчання моделі. Неможливість відстеження часу полягає у випадковості, при якій випадково згенерована комбінація не покращує результат навчання. Отже, намалюємо

графік затраченого часу для навчання моделі та прогнозування відгуків, результат розмістимо на рисунках 2.32 та 2.33 (для різних словників).

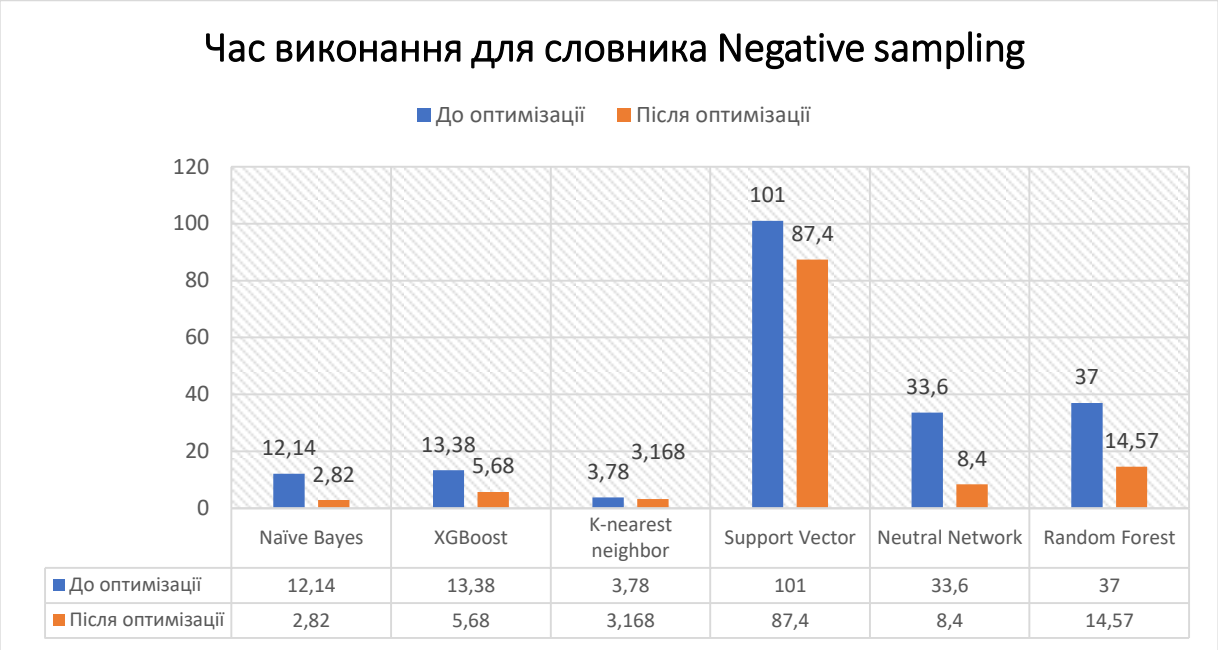

Рисунок 2.32. Час затрачений на навчання і класифікацію моделі на основі словника skip gram negative sampling.

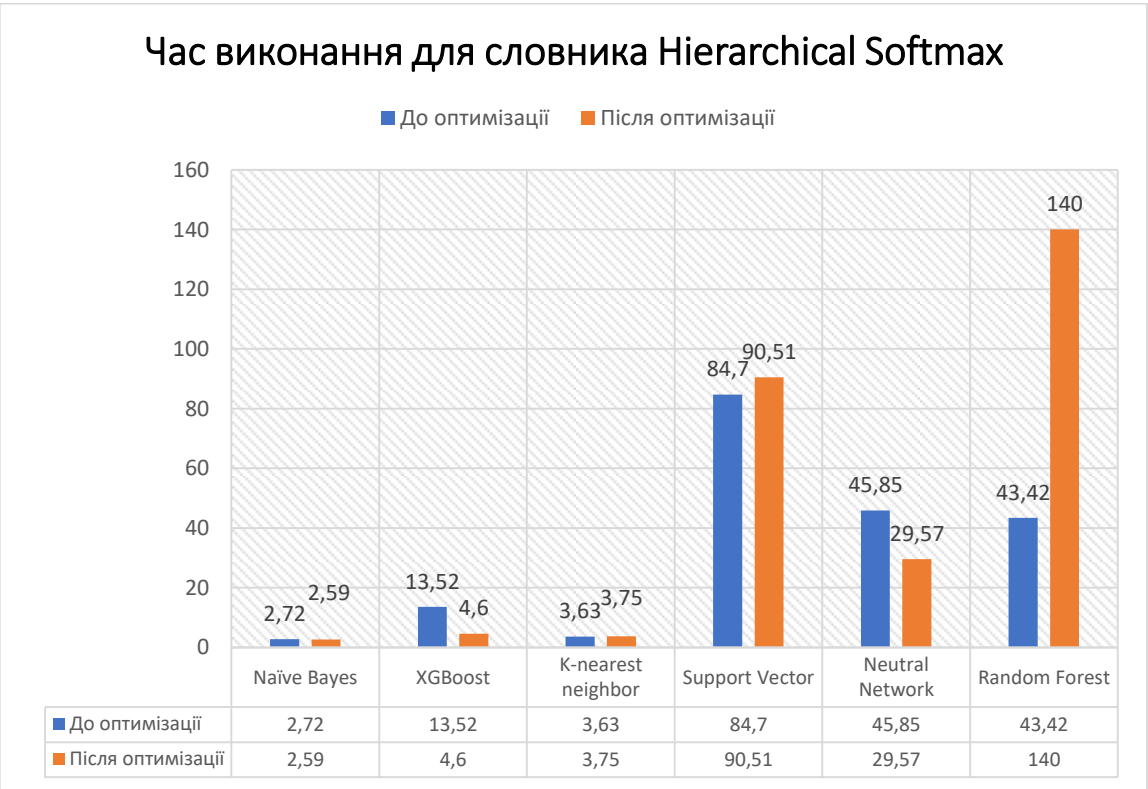

Рисунок 2.33. Час затрачений на навчання і класифікацію моделі на основі словника skip gram hierarchical softmax.

Аналізуючи три рисунки, найкращим опиняється метод XGBoost, який має найменшу суму помилок та відносно швидке виконання (4 секунди). Інша модель, K-nearest neighbor, також може бути використана, тому що практично не відрізняється за результатами від класифікатора XGBoost, але за якістю рішень вона все одно буде гірше.

Перші дві моделі швидко виконували розрахунки, на відміну від моделі Neutral Network, яка має лише 1 результат, де затрачено менше 10 секунд, серед чотирьох «прогонів» (з різною комбінацією гіперпараметрів для різних словників). Скоріш за все це пов’язано з тим, що модель потребувала донавчання і займалась саме цим процесом, а коли ми передали їй кращі гіперпараметри одразу - вона показала відносно швидкий результат. Проте не треба забувати, що до цього було витрачено час на пошук цих гіперпараметрів. Це значить, що якщо налаштовувати 4 гіперпараметри з 4 варіантами в кожному, ми отримаємо 256 комбінацій параметрів для «прогонів». При цьому 1 «прогон» займає, як ми побачили, приблизно 30 секунд. Тобто загалом процес пошуку оптимального набору параметрів буде відбуватися приблизно 2 години. Не забуваємо, що з ростом корпусу буде зростати також і час його обробки.

2.7 Висновок до розділу

Отже, ми визначилися з кращим класифікатором, розглянемо ще пару характеристик цих моделей (популярні слова) та підведемо підсумок роботи. Кінцевою задачею цієї роботи була дослідити і навчитися класифікувати споживачів за тональністю їх відгуків. Етап класифікації було показано у підрозділах вище, а дослідження споживачів буде викладено нижче у двох рисунках, які зображують часто вживані слова до кожної тональності відгуку.

На рисунку 2.34 зображено хмару слів для категорії споживачів, які позитивно оцінили товар.

Рисунок 2.34. Хмара часто вживаних слів у позитивних відгуках.

На рисунку вище добре виділяються слова, за допомогою кольору та розміру. На ньому ми можемо побачити наступні слова: «recommend», «Great», «good», «satisfied», «beautiful», «quality product», «perfect», «fast delivery». Тобто слова, які позитивно характеризують товар або щось з ним пов’язане, наприклад, швидка доставка.

Тепер перейдемо іншого класу відгуків – негативних. На рисунку 2.35 ми можемо продивитися хмару слів, які найчастіше використовуються у таких відгуках, і побачити контраст вживаних слів.

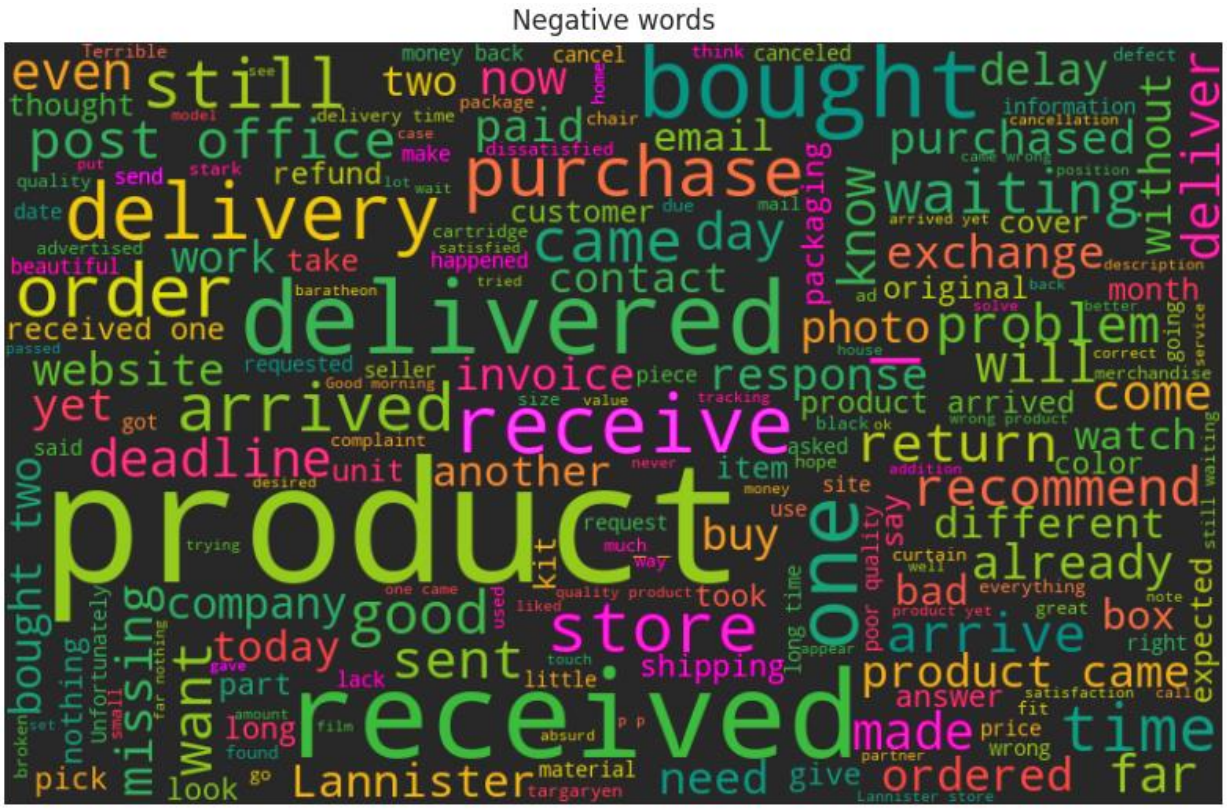

Рисунок 2.35. Хмара часто вживаних слів у негативних відгуках.

На цій хмарі слова вже не так яскраво виділяються, як на хмарі з позитивними відгуками. Це пов’язано з тим, що деякі слова вирвані із словосполучень і замість «not good» ми отримаємо просто «good». Проте ми все рівно можемо побачити негативно окрашені слова: «problem», «deadline», «return», «missing», «packaging», «waiting», «still», «delay». Аналізуючи ці слова ми можемо припустити, що клієнт мав проблеми з кінцевим строком доставки, затримками, возвратами посилок або навіть втратами товару. Також можливі повідомлені щодо поганої упаковки для транспортування.

Загалом, наявність цих хмар слів може дозволити нам робити пошук окремих випадків за тегами: погана упаковка, втрата товару, затримка доставки, погана якість або підробка тощо. Проте у цій роботі пошук негативних відгуків відбувався не лише за тегами, а також і контекстом цього тегу. Це відбувалося за допомогою аналізу локального контексту визначеної ширини (кількість слів, які входять до контексту), а саме методу word2vec.

Існує також метод глобального аналізу контексту (GloVe), проте і це рішення не межа існуючих можливостей. Для того, щоб покращити результат нашої класифікації, ми можемо скористатися рекурентними нейронними мережами, тому що довжина наших відгуків не призведе до втрати градієнту нейронної мережі. У купі до цього, також можна вручну перебрати навчальну вибірку для більш точного налаштування під конкретні задачі, наприклад, визначати проблеми з упаковками або доставками.

ВИСНОВКИ

У цій кваліфікаційній роботі було розглянуто питання дослідження поведінки споживача за допомогою методів машинного навчання. Метою цієї роботи було підвищення ефективності роботи бразильського сайту маркетплейс «Olist» за рахунок класифікації споживачів на основі їхніх відгуків щодо купленого товару.

Об'єктом дослідження виступає результат роботи сайту за 2017-2018 роки, який накопичив за цей час приблизно сто тисяч відгуків, а сума продаж сягнула близько 2 млн. доларів США.

Предметом дослідження є створення автоматизованого класифікатора реакцій споживачів на основі їхніх відгуків щодо купленого товару на маркетплейс-сайті «Olist» за допомогою методів машинного навчання.

Виконання роботи було поділено на задачі, вирішення яких було послідовно описано у першому та другому розділі. Першочергово було проаналізовано сучасні підходи обробки природномовних текстів, так як це єдині дані про споживача, з якими буде взаємодіяти наш майбутній класифікатор. У першому розділі було описано алгоритм роботи методів word2vec та GloVe.

Наступним етапом роботи був аналіз існуючих методів класифікації, які використовуються у галузі NLP. Список наявних методів обширний, але було обрано всього 6. Серед них: Метод Наївного Баєсу, Метод k-Найближчих Сусідів (kNN), дерево рішень, Випадковий ліс, XGBoost, нейронна мережа. Кожен з методів було описано у першому розділі. Загалом методи відрізняються між собою за алгоритмом, проте деякі з них були модифікаціями до інших. Результат їхньої роботи було досліджено на етапі побудови класифікатора.

Перед початком будь-яких дій пов'язаних зі створенням словника та класифікатора, потрібно впевнитися, що дані, які будуть використовуватися,

не будуть створювати неточності або ставити причиною зупинки алгоритму. Для вирішення цієї проблеми було виконано комплекс дій, які направлені на видалення дублікатів, знаходження втрачених записів, поєднання відгуків, видалення пустих відгуків. В результаті такої обробки початковий розмір корпус відгуків (99224 од.) було зменшено майже в 2,5 рази (40017 од.).

Для побудови словника було обрано модель word2vec. Для більшого вибору було побудовано 4 варіанти реалізації, які відрізнялися між собою за способом навчання. Їх порівняли за якістю визначення близькості між словами. У результаті було обрано два варіанти словників, які навчалися за методом прогнозування контексту (skip-gram), але відрізнялися тим, що одна модель мала додаткові неправильні варіанти відповідей (negative sampling), а інша (hierarchical softmax) - ні.

Побудований словник буде виступати для класифікатора помічник у розумінні природньої мови, тому тільки зараз можна будувати класифікатор. Для більшого вибору було побудовано загалом 12 моделей, на кожний варіант словника (2 варіанти загалом) виконується оптимізація гіперпараметрів класифікатора. Обмежений вибір словників пов’язаний з великими затратами обчислювальних ресурсів, які можуть не знайти кращого рішення, ніж стандартне. Причинами цього є випадковість вибору гіперпараметрів, які перед генерацією не беруть до уваги можливі результати - тому вони можуть бути гіршими, ніж ті, які пропонують розробники бібліотек, тобто стандартні.

Для порівняння методів використовувалися декілька показників такі як F1-score, ROC крива, ROC-AUC, час роботи. На словнику без неправильних варіантів (hierarchical softmax), усі методи, окрім Наївного Баєсу, показали майже однакову ROC криву, звідси випливає, що ROC-AUC також приблизно однаковий. Тому вирішальними для нас були F1-score та час роботи. F1-score відрізнявся між реалізаціями в межах 5 пунктів, проте для нас було важливим не допустити високий показник помилок другого роду

(неправильно класифікованих негативних відгуків), тому що саме їх пошук є пріоритетним. Як було зазначено, негативні відгуки є рушієм змін у підприємстві. Для першого словника сума помилок першого і другого роду майже однакова, проте всередині суми частка помилок мінялась. Тобто в одному класифікаторі більше помилок першого роду, а в іншому – другого роду, але в сумі ці помилки дорівнювали приблизно одному числу. Отже, серед методів з мінімальним числом помилок другого роду було відмічено наступні: XGBoost та kNN. Також було помічено екстремально низький відсоток помилок у методі Наївного Баєсу, але кількість помилок першого роду ставила під сумнів взагалі можливість коректної роботи такого класифікатора. За часом виконання kNN був швидше, але за сумою помилок і кількістю помилок другого роду був краще XGBoost, таким чином він вважається кращим класифікатором на словнику hierarchical softmax.

Словник negative sampling мав більш різноманітний результат і визначення кращого методу було легшим. У цьому варіанті виділяються методи XGBoost та класифікатор нейронною мережею. За показником ROC кривих було видно, що ці два методи виділяються серед інших тим, що вигнуті в сторону лівого кута, показник ROC-AUC це підтверджує. За значенням F1-score методи приблизно однакові, а якщо поглибитися у відношення помилок першого і другого роду, то значення других співпадають, а ось помилок першого роду більше у методі нейронної мережі. За часом виконанням класифікатор XGBoost справляється набагато швидше, ніж класифікатор нейронною мережею, можливою причиною може бути довчання моделі після підбору гіперпараметрів.

Отже, за підсумками аналізу методів, які базуються на двох словниках, метод XGBoost показав кращий результат, а до використання рекомендується словник skip gram з negative sampling. У кінці роботи було наведено хмари слів, які використовуються у випадку позитивних та негативних відгуків. У випадку позитивних відгуків це були слова: «Great», «good», «satisfied»,

«beautiful», «quality product», «perfect», «fast delivery». Тобто слова, які позитивно характеризують товар або щось з ним пов’язане, наприклад, швидка доставка. Для негативних відгуків застосовувались наступні слова: «problem», «deadline», «return», «missing», «packaging», «waiting», «still», «delay». Аналізуючи цей набір слів ми можемо припустити, що клієнт мав проблеми з кінцевим строком доставки, затримками, возвратами посилок або навіть втратами товару. Також можливі повідомлені щодо поганої упаковки для транспортування.

Наявність цих хмар слів може дозволити нам робити пошук окремих випадків за тегами: погана упаковка, втрата товару, затримка доставки, погана якість або підробка тощо. Проте також важливий і контекст використання слів. У цій роботі було розглянуто метод машинного навчання word2vec, проте для покращення результату можна використовувати рекуренту нейронну мережу, яка має пам’ять про контекст, що дасть змогу більш детально досліджувати текст.

Загалом, на даному етапі розроблена система досягла поставленої мети дослідження. Система може бути вбудована у роботу підприємства, щоб автоматично обробляти відгуки і знаходити проблемні замовлення, що буде достатнім для збору даних і початку пошуку рішень виявлених проблем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Історія машинного перекладу, URL : https://uk.wikipedia.org/wiki/Історія_машинного_перекладу (дата звернення: 30.05.2023)
2. Machine translation, URL: https://en.wikipedia.org/wiki/Machine_translation (дата звернення: 30.05.2023)
3. NLP. Основи. Техніки Саморозвиток. Частина 1, URL: https://habr.com/ru/companies/contentai/articles/437008/ (дата звернення: 30.05.2023)
4. 5 методів обробки природної мови, які стрімко змінюють навколишній світ, URL: https://neurohive.io/ru/osnovy-data-science/5-metodov-v-nlp-kotorye-izmenjat-obshhenie-v-budushhem/ (дата звернення: 30.05.2023)
5. Обломова О. Natural language processing , URL: https://4brain.ru/blog/natural-language-processing (дата звернення: 30.05.2023)
6. Chapter 4. Text Classification URL: https://www.oreilly.com/library/view/practical-natural-language/9781492054047/ch04.html (дата звернення: 30.05.2023)
7. Байєсіанство, URL: https://ru.wikipedia.org/wiki/Байесианство (дата звернення: 30.05.2023)
8. Теорема Байеса URL: https://ru.wikipedia.org/wiki/Теорема_Байеса (дата звернення: 30.05.2023)
9. Наївний байєсовський класифікатор, URL: https://ru.wikipedia.org/wiki/Наивный_байесовский_классификатор (дата звернення: 30.05.2023)
10. Наївний Байєс, або про те, як математика дозволяє фільтрувати спам , URL: https://habr.com/ru/articles/415963/ (дата звернення: 30.05.2023)

11. Метод опорних векторів , URL: https://uk.wikipedia.org/wiki/Метод_опорних_векторів (дата звернення: 30.05.2023)
12. Дерево рішень, URL: https://ru.wikipedia.org/wiki/Дерево_решений (дата звернення: 30.05.2023)
13. Що таке дерево рішень та де його використовують? URL: https://habr.com/ru/companies/productstar/articles/523044 (дата звернення: 30.05.2023)
14. Введення у бегінг у машинному навчанні, URL: https://www.codecamp.ru/blog/bagging-machine-learning (дата звернення: 30.05.2023)
15. Випадкові дерева та випадковий ліс. Бутстреп та бегінг ,URL: https://proproprogs.ru/ml/ml-sluchaynye-derevya-i-sluchaynyy-les-bootstrap-i-bagging (дата звернення: 30.05.2023)
16. Ансамблеві методи машинного навчання, URL: https://habr.com/ru/articles/571296 (дата звернення: 30.05.2023)
17. Відкритий курс машинного навчання. Тема 10. Градієнтний бустинг, URL: https://habr.com/ru/companies/ods/articles/327250 (дата звернення: 30.05.2023)
18. Метод найближчих сусідів , URL: http://www.machinelearning.ru/wiki/index.php?title=KNN (дата звернення: 30.05.2023)
19. Метод K-Nearest Neighbors. Розбір без використання бібілотек та з використанням бібліотек, URL: https://habr.com/ru/articles/680004 (дата звернення: 30.05.2023)
20. Нейрона мережа, що це таке і як створити свою? Детальна інструкція, URL: https://habr.com/ru/articles/736466 (дата звернення: 30.05.2023)
21. Чудовий світ Word Embeddings: які вони бувають і для чого потрібні? URL: https://habr.com/ru/companies/ods/articles/329410 (дата звернення: 30.05.2023)

22. Lei Mao, Hierarchical Softmax, URL: https://leimao.github.io/article/Hierarchical-Softmax (дата звернення: 30.05.2023)
23. Word2vec в картинках, URL: https://habr.com/ru/articles/446530 (дата звернення: 30.05.2023)
24. J.Pennington, R.Socher, C.D.Manning GloVe: Global Vectors for Word Representation. 2014. DOI: https://doi.org/10.3115/v1/D14-1162
25. Відкритий курс машинного навчання. Тема 5. Композиції: бегінг, випадковий ліс, URL: https://habr.com/ru/companies/ods/articles/324402 (дата звернення: 30.05.2023)
26. Алгоритм AdaBoost, URL: https://habr.com/ru/companies/otus/articles/503888/ (дата звернення: 30.05.2023)
27. Бібліотека Gensim, реалізація word2vec, URL: https://github.com/RaRe-Technologies/gensim (дата звернення: 30.05.2023)
28. Аналіз тональності тексту з використанням word2vec та реалізацією в pipeline Python ,URL: https://edwvb.blogspot.com/2018/07/analiz-tonalnosti-teksta-s-ispolzovaniem-word2vec-i-realizaciej-v-pipeline-python.html (дата звернення: 30.05.2023)
29. T. Mikolov, K.Chen, G.Corrado, J.Dean. Efficient Estimation of Word Representations in Vector Space. 2013. с. 12. DOI: https://doi.org/10.1109/ICOSEC49089.2020.9215319
30. Brazilian E-Commerce Public Dataset by Olist, URL: https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce (дата звернення 30.05.2023)

Додатки

Додаток А

Відомості матеріалів дипломної роботи

		Позначення	Найменування	Кількість	Примітка
1					
2			Документація		
3					
4		САУ.КР.23.02.ПЗ	Пояснювальна записка	141	Формат А4
5					
6		САУ.КР.23.02.ДМ	Демонстраційні матеріали	15	Презентація на CD-R
7					
8		САУ.КР.23.02.КР	Копія роботи	1	Диск CD-R

					САУ.КР.23.02.ДА.ПЗ			
Зм	Лист	№ докум.	Підпис	Дата				
Розроб.		Говоруха			**Матеріали кваліфікаційної роботи**	Літ	Аркуш	Аркушів
К. розд.		Желдак						
Керівн.		Желдак				НТУ «ДП», 12; 124-19-2		
Н. контр.		Хом’як						
Зав. каф.		Желдак						

Додаток Б

Відгук
на кваліфікаційну роботу бакалавра
студента гр. 124-19-2
Говорухи Андрія Юрійовича

Тема кваліфікаційної роботи: «Дослідження поведінки споживача за допомогою методів машинного навчання».

Обсяг кваліфікаційної роботи 141 стор.

Мета кваліфікаційної роботи: підвищення ефективності роботи сайту маркетплейсу «Olist» за рахунок класифікація споживачів на основі їхніх відгуків щодо купленого товару.

Актуальність теми обумовлена все зростаючою важливістю обробки і врахування в роботі великих торгівельних мереж в інтернет відгуків клієнтів. Останні викладені природною мовою, що потребує застосування методів обробки природньої мови (NLP), а також методів класифікації за допомогою машинного навчання.

Тема кваліфікаційної роботи безпосередньо пов'язана з об'єктом діяльності бакалавра спеціальності 124 Системний аналіз, оскільки предметом дослідження є створення автоматизованого класифікатора реакцій споживачів на основі їхніх відгуків щодо купленого товару на маркетплейс-сайті *«Olist»* за допомогою методів машинного навчання.

Виконані в кваліфікаційній роботі завдання відповідають вимогам ступеня бакалавра. Оригінальність наукових рішень полягає в реалізації інформаційних систем побудови словників для обробки природньої мови та класифікаторів на їх основі.

Практичне значення результатів кваліфікаційної роботи полягає в тому, що в ній запропонована автоматизація процесу обробки зворотної реакції споживача, що зекономить грошові ресурси підприємства і дозволить перенаправити їх на вирішення виявлених проблем.

Оформлення пояснювальної записки та демонстраційного матеріалу до неї виконано згідно з вимогами. Роботу виконано самостійно, відповідно до завдання та у повному обсязі.

У роботі відзначено такі **недоліки**:

1. Значна частина інформаційно-аналітичного розділу присвячена аналізу методів класифікації текстів, які надалі не використовуються або визнаються автором недостатньо якісними, що переобтяжує текст.

Кваліфікаційна робота в цілому заслуговує оцінки: «відмінно» (98 балів).

З урахуванням висловлених зауважень авторка заслуговує присвоєння освітньої кваліфікації «бакалавр з системного аналізу».

Керівник кваліфікаційної роботи,
к.т.н., доц., завідувач кафедри САУ Желдак Т.А.

Додаток В

Лістинг програми перекладу на англійську

```
import math
from typing import Union, Any

import numpy as np
from googletrans import Translator
import pandas as pd
from numpy import isnan
from pandas import Series, DataFrame

dataset = pd.read_csv('reviews.csv')
df = pd.DataFrame(dataset)
df.to_csv('reviews_translated.csv')

translator = Translator(service_urls=['translate.googleapis.com'])
array_title = df['review_comment_title'].values
array_message = df['review_comment_message'].values

print(len(array_title))
for i in range(len(array_title)):
    if(pd.isnull(array_title[i])==False):
        array_title[i]=translator.translate(array_title[i], dest='en').text
    if (pd.isnull(array_message[i]) == False):
        array_message[i] = translator.translate(array_message[i], dest='en').text

df['review_comment_title']=array_title
df['review_comment_message']=array_message
df.to_csv('reviews_translated.csv')
```

Додаток Г

Лістинг програми поєднання втрачених записів

```
import math
from typing import Union, Any
import numpy as np
import pandas as pd
from numpy import isnan
from pandas import Series, DataFrame

dataset = pd.read_csv('reviews.csv')
reviews= pd.DataFrame(dataset)
dataset=pd.read_csv('items.csv')
items=pd.DataFrame(dataset)
dataset=pd.read_csv('orders.csv')
orders=pd.DataFrame(dataset)
dataset=pd.read_csv('payments.csv')
payments=pd.DataFrame(dataset)
count_del_rows=0
print(count_del_rows)

def change_identificators(index,reviews,orders,items,payments,customer_id,order,temp_order,review):
    items_to_edit = items.loc[items['order_id'] == temp_order]
    for k in range(len(items_to_edit)):
        items_to_edit.loc[items_to_edit.index[k], 'order_id'] = order
    # у таблиці замовлень замінюємо номер покупц на потрібний
    orders.loc[orders.index[orders['order_id'] == temp_order], 'customer_id'] = customer_id
    # у таблиці замовлень замінюємо номер замовлення на потрібне
    orders.loc[orders.index[orders['order_id'] == temp_order], 'order_id'] = order
    # у таблиці оплат замінюємо номер замовлення на потрібне
    payments.loc[payments.index[payments['order_id'] == temp_order], 'order_id'] = order
    teeemp = index
    reviews.loc[index, 'order_id'] = order
    return reviews,orders,items,payments
for i in range(len(reviews['review_id'])):
      match=reviews['review_id'][i]
    b=reviews.loc[reviews['review_id']==match]
    p=b.index
```

```
  #якщо існує два і більше відгуків для різних замовлень, тоді перевірити поєднати два замовлення
   if(len(b)>1):
      # needed order_id щоб замінити у всіх відгуках, які будуть зустрічатися
      for k in range(len(b)):
         #індекс який потрібно використовувати як заміну для інших
         sub_index=b.index[k]
         order_id_nd = reviews['order_id'][b.index[k]]
         review_id_nd = reviews['review_id'][b.index[k]]
         # витаскуємо список товарів за замовленням
         list_items = items.loc[items['order_id'] == order_id_nd]
         customer_id = orders.loc[orders.index[orders['order_id'] == order_id_nd],'customer_id']
         if (list_items.empty == False):
               break
      if(customer_id.empty==True):
         continue
      for j in range(len(b)):
         temp_customer_id = ''
         temp_order_id = ''
         if(b.index[j]!=sub_index):
            temp_order_id=reviews.loc[b.index[j],'order_id']
            list_reviews = reviews.loc[reviews['order_id'] == temp_order_id]
            temp_customer_id = orders.loc[orders.index[orders['order_id'] == temp_order_id],'customer_id']
            if(len(list_reviews)>1):
               for k in range(len(list_reviews)):
                  reviews,orders,items,payments=change_identificators( list_reviews.index[k], reviews, orders, items, payments, customer_id[customer_id.index[0]], order_id_nd, temp_order_id,review_id_nd)
                  reviews.loc[list_reviews.index[k], 'review_id'] = review_id_nd
            else:
               reviews,orders,items,payments=change_identificators(b.index[j], reviews, orders, items, payments,customer_id[customer_id.index[0]], order_id_nd, temp_order_id, review_id_nd)
            # #у таблиці замовлень замінюємо номер покупц на потрібний
       orders.loc[orders.index[orders['order_id']==temp_order_id],'customer_id']=customer_id[customer_id.index[0]]


reviews.to_csv('reviews_cleared.csv') # назначаємо один ідентифікатор для ордерів, які стосувалися одного користувача з різними ідентифікаторами, але відносилися до одного відгуку

orders.to_csv('orders_cleared.csv') # ми позбулися ордерів, де один користувач мав декілька різних ідентифікаторів, а саме тепер для одного ордеру будуть однакові покупці

items.to_csv('items_cleared.csv') # товари які стосувалися одного відгуку, але мали різні ідентифікатори замовлення, були поєднані в одне замовлення

payments.to_csv('payments_cleared.csv')
```

Лістинг програми поєднання відгуків

```
import math
from typing import Union, Any
import numpy as np
import pandas as pd
from numpy import isnan
from pandas import Series, DataFrame

dataset = pd.read_csv('reviews.csv')
reviews= pd.DataFrame(dataset)
dataset=pd.read_csv('items.csv')
items=pd.DataFrame(dataset)
list=reviews['found_matches']
for i in range(len(reviews['found_matches'])):
    match=reviews['found_matches'][i]
    list_items=items.loc[items['order_id']==match]
    #якщо існує два і більше відгуків для одного замовлення, тоді перевірити поєднати два відгуки і взяти середній
бал
    b=reviews.loc[reviews['found_matches']==match]
    p=b.index
    if(len(b)>1):
        sum = 0
        count=0
        review_comment_message=''
        for j in range(len(b)):
            if(b.index[j]!=i):
                ss=b.index[j]
                sum=sum+reviews['review_score'][b.index[j]]
                count+=1
                reviews.loc[b.index[j],'found_matches']=None
                if (pd.isnull(reviews['review_comment_message'][b.index[j]])==False):
                    review_comment_message=review_comment_message+';'+reviews['review_comment_message'][b.index[j]]
        reviews.loc[i,'review_score']=(sum+reviews['review_score'][i])/(count+1)
        if (pd.isnull(reviews['review_comment_message'][i]) == False):
            review_comment_message = review_comment_message+';'+reviews['review_comment_message'][i]
        reviews.loc[i,'review_comment_message']=review_comment_message

reviews.to_csv('reviews_cleared.csv')
```

Додаток Д

Лістинг програми створення словника

```
import numpy as np

import pandas as pd

from numpy import savetxt


##tokenizer

import nltk

import nltk.data

from sklearn.metrics import rand_score


tokenizer = nltk.data.load('nltk:tokenizers/punkt/english.pickle')

from nltk.tokenize import sent_tokenize, word_tokenize

import re


def preproc(sentence):

    sent_text = re.sub(r'[^\w\s]', '', sentence)

    words = sent_text.lower().split()

    return (words)



def senttxt(sent, tokenizer, remove_stopwords=False):

    raw_sentences = tokenizer.tokenize(sent.strip())

    sentences = []

    for raw_sentence in raw_sentences:

                                sentences.append(preproc(raw_sentence))

    len(sentences)

    return sentences


dataset = pd.read_csv('reviews.csv')

reviews= pd.DataFrame(dataset)


review_ratio=reviews['sentiment'].value_counts()

#друкуємо кількість позитивних і негативних відгуків

print(review_ratio)


txt_snt = reviews['review_comment_message'].tolist()
```

```
sentences = []
for i in range(0, len(reviews)):
        sent = txt_snt[i].replace("/.", '')
        sentences += senttxt(sent, tokenizer)

##Word2Vec
from gensim.models.word2vec import Word2Vec
# model = Word2Vec(size=100, min_count=1,negative=10,window=10,sg=1,workers=6)
model=Word2Vec.load('model_sgram_hier_softmax_1to2.model')
# model.build_vocab(sentences)

# model.train(sentences, total_examples=model.corpus_count, epochs=model.epochs)
# tem=model.wv.index2word
# b=model.wv.vectors
w2v = dict(zip(model.wv.index2word, model.wv.vectors))
# model.save('word2vec_model_1_to_2_sg_10_window_10_negative.model')

#check the difference between antonyms

pairs = [
    ('good', 'bad'),
    ('fast', 'slow'),
    ('fast', 'long'),
    ('true', 'box'),
    ('bad', 'quality'),
    ('fast','delivery'),
    ('order','item'),
    ('box','delivery'),
    ('counterfeit', 'delivery'),
    ('counterfeit','fake'),
    ('fast','equipment'),
    ('item','equipment'),
    ('good','nice'),
    ('delivery','post')
]
for w1, w2 in pairs:
    print('%r\t%r\t%.2f' % (w1, w2, model.similarity(w1, w2)))
```

Додаток Е

Лістинг програми створення класифікаторів

```
import numpy as np

import pandas as pd

from numpy import savetxt

##tokenizer

import nltk

import nltk.data

from sklearn.metrics import rand_score


tokenizer = nltk.data.load('nltk:tokenizers/punkt/english.pickle')

from nltk.tokenize import sent_tokenize, word_tokenize

import re

def preproc(sentence):

    sent_text = re.sub(r'[^\w\s]', '', sentence)

    words = sent_text.lower().split()

    return (words)

def senttxt(sent, tokenizer, remove_stopwords=False):

    raw_sentences = tokenizer.tokenize(sent.strip())

    sentences = []

    for raw_sentence in raw_sentences:

                                sentences.append(preproc(raw_sentence))

    len(sentences)

    return sentences

dataset = pd.read_csv('reviews.csv')

reviews= pd.DataFrame(dataset)

review_ratio=reviews['sentiment'].value_counts()

#друкуємо кількість позитивних і негативних відгуків

print(review_ratio)

txt_snt = reviews['review_comment_message'].tolist()

sentences = []
```

```
for i in range(0, len(reviews)):
        sent = txt_snt[i].replace("/.", '')
        sentences += senttxt(sent, tokenizer)
##Word2Vec
from gensim.models.word2vec import Word2Vec
model=Word2Vec.load('model_sgram_hier_softmax_1to2.model')
# завантажуємо готову модель
w2v = dict(zip(model.wv.index2word, model.wv.vectors))
#check the difference between antonyms
pairs = [
    ('good', 'bad'),
    ('fast', 'slow'),
    ('fast', 'long'),
    ('true', 'box'),
    ('bad', 'quality'),
    ('fast','delivery'),
    ('order','item'),
    ('box','delivery'),
    ('counterfeit', 'delivery'),
    ('counterfeit','fake'),
    ('fast','equipment'),
    ('item','equipment'),
    ('good','nice'),
    ('delivery','post')]
for w1, w2 in pairs:
    print('%r\t%r\t%.2f' % (w1, w2, model.similarity(w1, w2)))
class normword2vec():
    def transform(self, X, y=None, **fit_params):
        X['review_comment_message'] = X['review_comment_message'].str.strip()
        X['review_comment_message'] = X['review_comment_message'].str.lower()
        X['review_comment_message'] = X['review_comment_message'].astype(str)
```

```
        X['review_comment_message'] = [re.sub(r'[^\w\s]', '', e) for e in X['review_comment_message']]
        return X['review_comment_message']
    def fit_transform(self, X, y=None, **fit_params):
        self.fit(X, y, **fit_params)
        return self.transform(X)
    def fit(self, X, y=None, **fit_params):
        return self
class MeanVect(object):
    def __init__(self, word2vec):
        self.word2vec = word2vec
        # if a text is empty we should return a vector of zeros
        # with the same dimensionality as all the other vectors
        self.dim = len(word2vec.values())
    def fit(self, X, y):
        return self
    def transform(self, X):
        return np.array([
            np.mean([self.word2vec[w] for w in words if w in self.word2vec]
                    or [np.zeros(100)], axis=0)
            for words in X
        ])


import sklearn
import gensim.sklearn_api
from sklearn.pipeline import Pipeline
from gensim.sklearn_api import W2VTransformer
from sklearn.ensemble import RandomForestClassifier
rf=RandomForestClassifier(max_depth= 13, max_features='sqrt',bootstrap=False   ,n_estimators=500,
min_samples_leaf=18,min_samples_split=12)
# rf=RandomForestClassifier()
pipeline_rf = Pipeline([
```

```
    ('selectword2vec', normword2vec()),
    ("word2vec", MeanVect(w2v)),
    ('model_fitting', rf)])

from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
#default settings
pipeline_gnb = Pipeline([
    ('selectword2vec', normword2vec()),
    ("word2vec", MeanVect(w2v)),
    ('model_fitting', gnb)])
from sklearn.svm import SVC
svc=SVC(C=50.0,degree=3,gamma='scale')
pipeline_svc = Pipeline([
    ('selectword2vec', normword2vec()),
    ("word2vec", MeanVect(w2v)),
    ('model_fitting', svc)])

from sklearn.neural_network import MLPClassifier
nnc=MLPClassifier(activation='relu',alpha=0.001112,learning_rate_init=0.0105,solver='adam')
pipeline_nnc = Pipeline([
    ('selectword2vec', normword2vec()),
    ("word2vec", MeanVect(w2v)),
    ('model_fitting', nnc)])

from xgboost import XGBClassifier
param=[{'subsample': 0.30000000000000004, 'scale_pos_weight': 0.5,
        'reg_lambda': 0.1, 'reg_alpha': 0.30000000000000004, 'min_child_weight': 0.2, 'max_leaves': 15,
        'max_depth': 15, 'max_delta_step': 1.0, 'max_bin': 8, 'learning_rate': 0.05,
        'gamma': 0.1125, 'colsample_bytree': 0.30000000000000004,
        'colsample_bynode': 0.30000000000000004, 'colsample_bylevel': 0.5}]
```

```
xgb =
XGBClassifier(subsample=0.5,scale_pos_weight=0.5,reg_lambda=0.5,reg_alpha=0.1,min_child_weight=0.2,max_leaves=8,

pipeline_xgb = Pipeline([
    ('selectword2vec', normword2vec()),
    ("word2vec", MeanVect(w2v)),
    ('model_fitting', xgb)])


from sklearn.neighbors import KNeighborsClassifier
knn=KNeighborsClassifier(weights='distance',n_neighbors=38,leaf_size=55,algorithm='brute')
pipeline_knn = Pipeline([
    ('selectword2vec', normword2vec()),
    ("word2vec", MeanVect(w2v)),
    ('model_fitting', knn)])


from sklearn import model_selection
from sklearn.model_selection import train_test_split
y = reviews['sentiment']
X = reviews
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, test_size=0.3, random_state=2)


from sklearn.model_selection import RandomizedSearchCV
#
import time
start_time=time.time()


s_gnb=pipeline_gnb.fit(X_train, y_train)
pred_gnb = pipeline_gnb.predict(X_test)
gnb_time=time.time()-start_time


s_xgb=pipeline_xgb.fit(X_train, y_train)
pred_xgb = pipeline_xgb.predict(X_test)
```

```
xgb_time=time.time()-start_time-gnb_time


s_knn=pipeline_knn.fit(X_train, y_train)

pred_knn = pipeline_knn.predict(X_test)

knn_time=time.time()-start_time-xgb_time-gnb_time


s_svc=pipeline_svc.fit(X_train, y_train)

pred_svc = pipeline_svc.predict(X_test)

svc_time=time.time()-start_time-knn_time-xgb_time-gnb_time


s_nnc=pipeline_nnc.fit(X_train, y_train)

pred_nnc = pipeline_nnc.predict(X_test)

nnc_time=time.time()-start_time-svc_time-knn_time-xgb_time-gnb_time


s_rf=pipeline_rf.fit(X_train, y_train)

pred_rf = pipeline_rf.predict(X_test)

rf_time=time.time()-start_time-nnc_time-svc_time-knn_time-xgb_time-gnb_time


import seaborn as sns

import matplotlib.pylab as plt

#matplotlib inline

from matplotlib.pylab import rcParams

from sklearn import metrics

# import f1_score, log_loss, auc, roc_curve, roc_auc_score, homogeneity_score

gnb_f1=metrics.f1_score(y_test,pred_gnb)

xgb_f1=metrics.f1_score(y_test,pred_xgb)

knn_f1=metrics.f1_score(y_test,pred_knn)

nnc_f1=metrics.f1_score(y_test,pred_nnc)

svc_f1=metrics.f1_score(y_test,pred_svc)

rf_f1=metrics.f1_score(y_test,pred_rf)

plt.figure(0).clf()
```

```
preds={'gnb': pred_gnb,'xgb':pred_xgb,'knn':pred_knn,'nnc':pred_nnc,'svc':pred_svc,'rf':pred_rf}

s=len(preds)

sss=preds.keys()

display = []

i=0

for key in (preds.keys()):

    fpr, tpr, thresholds = metrics.roc_curve(y_test, preds[key])

    roc_auc = metrics.auc(fpr, tpr)

    text=str(key)+" estimator, F1-score: " + str(round(metrics.f1_score(y_test,preds[key]),2))+ ", ROC-score: "
+str(round(metrics.roc_auc_score(y_test,preds[key]),2))

    # display.append(metrics.RocCurveDisplay(fpr=fpr, tpr=tpr, roc_auc=roc_auc,estimator_name=text))

    if(i>1):

        plt.plot(fpr, tpr, '--',label=text)

    else:

        plt.plot(fpr, tpr, '-', label=text)

    i+=1

plt.ylabel("True positive rate")

plt.xlabel("False positive rate")

plt.legend()

plt.show()


gnb_score=metrics.rand_score(y_test,pred_gnb)

nnc_score=metrics.rand_score(y_test,pred_nnc)

knn_score=metrics.rand_score(y_test,pred_knn)

rf_score=metrics.rand_score(y_test,pred_rf)

svc_score=metrics.rand_score(y_test,pred_svc)

xgb_score=metrics.rand_score(y_test,pred_xgb)
```

```
y_test_df=pd.DataFrame(y_test)

print(y_test_df)

y_test_df.to_csv('y_test_cleared.csv')

dataset=pd.read_csv('y_test_cleared.csv')


y_test_df=pd.DataFrame(dataset)

y_test_df.rename(columns = {'Unnamed: 0':'index'}, inplace = True)


y_test_df['review_message']=''

for i in range(len(y_test_df)):

    y_test_df['review_message'][i]=reviews['review_comment_message'][y_test_df['index'][i]]


pred_gnb_df=pd.DataFrame(pred_gnb)

pred_xgb_df=pd.DataFrame(pred_xgb)


y_test_df.to_csv('y_test_cleared_1.csv')

pred_gnb_df.to_csv('pred_gnb_cleared.csv')

pred_xgb_df.to_csv('pred_xgb_cleared.csv')


fig, axs = plt.subplots(2, 3)

ax = sns.heatmap((pd.crosstab(y_test, pred_gnb).apply(lambda r: r/y_test.size*100, axis=0)), fmt='.1f',cbar=None, annot=True, cmap="Blues",ax=axs[0,0],yticklabels=['Negative','Positive'],xticklabels=['Negative','Positive'])

axs[0,0].set_title("Naive Bayes")

axs[0,0].set_xlabel("Predict label")

axs[0,0].set_ylabel("Real label")

ax = sns.heatmap((pd.crosstab(y_test, pred_xgb).apply(lambda r: r/y_test.size*100, axis=0)), fmt='.1f',cbar=None, annot=True, cmap="Blues",ax=axs[0,1],yticklabels=['Negative','Positive'],xticklabels=['Negative','Positive'])


axs[0,1].set_title("Extreme Gradient Boosting")

axs[0,1].set_xlabel("Predict label")

axs[0,1].set_ylabel("Real label")
```

```
ax = sns.heatmap((pd.crosstab(y_test, pred_knn).apply(lambda r: r/y_test.size*100, axis=0)), fmt='.1f',cbar=None, annot=True, cmap="Blues",ax=axs[1,1],yticklabels=['Negative','Positive'],xticklabels=['Negative','Positive'])

# axs[1,1]=ax

axs[1,1].set_title("k-nearest neighbour")

axs[1,1].set_xlabel("Predict label")

axs[1,1].set_ylabel("Real label")

ax = sns.heatmap((pd.crosstab(y_test, pred_nnc).apply(lambda r: r/y_test.size*100, axis=0)), fmt='.1f',cbar=None, annot=True, cmap="Blues",ax=axs[1,0],yticklabels=['Negative','Positive'],xticklabels=['Negative','Positive'])

# axs[1,0]=ax


axs[1,0].set_title("Neural Network Classification")

axs[1,0].set_xlabel("Predict label")

axs[1,0].set_ylabel("Real label")

ax = sns.heatmap((pd.crosstab(y_test, pred_svc).apply(lambda r: r/y_test.size*100, axis=0)), fmt='.1f',cbar=None, annot=True, cmap="Blues",ax=axs[1,2],yticklabels=['Negative','Positive'],xticklabels=['Negative','Positive'])


axs[1,2].set_title("Support Vector  Classification")

axs[1,2].set_xlabel("Predict label")

axs[1,2].set_ylabel("Real label")

ax = sns.heatmap((pd.crosstab(y_test, pred_rf).apply(lambda r: r/y_test.size*100, axis=0)), fmt='.1f',cbar=None, annot=True, cmap="Blues",ax=axs[0,2],yticklabels=['Negative','Positive'],xticklabels=['Negative','Positive'])

# axs[0,2]=ax


axs[0,2].set_title("Random Forest Classification")

axs[0,2].set_xlabel("Predict label")

axs[0,2].set_ylabel("Real label")

plt.show()

end_time=time.time()-start_time


from wordcloud import WordCloud

#Most used words in negative reviews

negative_words=' '.join([text for text in reviews['review_comment_message'][reviews['sentiment']==0]])
```

```
wordcloud=WordCloud(width=800, height=500, random_state=21, max_font_size=110).generate(negative_words)

plt.figure(figsize=(10,7))

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis('off')

plt.title('Negative words')

plt.show()

negative_words_top=pd.DataFrame(wordcloud.layout_)

neg_words_top=negative_words_top.iloc[:, 0:2]

neg_words_top.sort_values(1,ascending=False)

#Most used words in positive reviews

positive_words=' '.join([text for text in reviews['review_comment_message'][reviews['sentiment']==1]])

wordcloud=WordCloud(width=800, height=500, random_state=21, max_font_size=110).generate(positive_words)

plt.figure(figsize=(10,7))

plt.imshow(wordcloud, interpolation='bilinear')

plt.axis('off')

plt.title('Positive words')

plt.show()

positive_words_top=pd.DataFrame(wordcloud.layout_)

pos_words_top=positive_words_top.iloc[:, 0:2]

pos_words_top.sort_values(1,ascending=False)
```