НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

«На правах рукопису»
УДК 004.422

«До захисту допущено»
Завідувач кафедри
_____ Едуард ЖАРІКОВ
«___»_____2022 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інженерія програмного забезпечення інформаційних систем»

зі спеціальності 121 «Інженерія програмного забезпечення»

на тему: «Інтелектуальна система аналізу та класифікації вхідних e-mail повідомлень на основі їх вмісту»

Виконав (-ла):
студент ІІ курсу, групи ІП-з11мп
Пиж Олександр Ігорович _____

Керівник:
доц., к.т.н., доц.,
Крамар Юлія Михайлівна _____

Рецензент:
доцент кафедри ІСТ, к.т.н., доц.,
Амонс Олександр Анатолійович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2022 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет інформатики та обчислювальної техніки

Кафедра інформатики та програмної інженерії

Рівень вищої освіти – другий (магістерський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення інформаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Едуард ЖАРІКОВ

«___»_____2022р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Пиж Олександр Ігорович

1. Тема дисертації «Інтелектуальна система аналізу та класифікації вхідних e-mail повідомлень на основі їх вмісту», науковий керівник дисертації Крамар Юлія Михайлівна, кандидат технічних наук, доцент, затверджені наказом по університету від «08» листопада 2022 р. № 4097-с

2. Термін подання студентом дисертації «9» грудня 2022 р.

3. Об’єкт дослідження – процес комунікації з клієнтами у технологічних компаніях та компаніях фінансового сектору.

4. Предмет дослідження – сучасні методи машинного навчання, алгоритми класифікації.

5. Перелік завдань, які потрібно розробити – аналіз існуючих рішень для класифікації e-mail повідомлень; аналіз методів обробки та класифікації текстових даних; розробка програмного забезпечення для класифікації вхідних повідомлень; дослідження ефективності розробленого рішення.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу – загальна структурна схема розгортання застосунку, схема структурна класів, схема структурна бази даних.

7. Орієнтовний перелік публікацій – стаття та тези доповіді на третій Всеукраїнській науково-практичній конференції молодих вчених та студентів

«ІНЖЕНЕРІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ І ПЕРЕДОВІ ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ» (SoftTech-2022 осінь).

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9.Дата видачі завдання «1» вересня 2022 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання	Примітка
1	Огляд літератури	1.09.2022	
2	Порівняльний аналіз існуючих методів класифікації	12.09.2022	
3	Формулювання постановки математичної моделі задачі	20.09.22	
4	Аналіз сучасних методів обробки текстів за допомогою машинного навчання	27.09.2022	
5	Розробка інформаційного та програмного забезпечення	10.10.2022	
6	Проведення експериментальних досліджень розробленого ПЗ	25.10.2022	
7	Виконання експериментальних досліджень	08.11.2022	
8	Оформлення пояснювальної записки	17.11.2022	
9	Подання дисертації на попередній захист	22.11.2022	
10	Подання дисертації на захист	9.12.2022	

Студент _____ Олександр ПИЖ

Науковий керівник _____ Юлія КРАМАР

РЕФЕРАТ

Магістерська дисертація: 104 с., 23 рис., 27 табл., 48 джерел, 1 додаток.

Актуальність. У сучасному світі, ведення бізнесу зазвичай передбачає обов'язковий контакт з клієнтами або користувачами в тому чи іншому вигляді. Більшість сучасних компаній, діяльність яких пов'язана з інформаційними технологіями, наданням послуг, банківською справою та іншим, мають у своєму складі відділи або департаменти, які виконують функції надання технічної підтримки, прийняття скарг на якість сервісу чи виконують будь-який інший вид контакту зі споживачами.

Саме тому, по мірі розвитку та зростання бізнесу, компанії змушені залучати все більше і більше працівників для обслуговування потреб клієнтів онлайн, що в свою чергу, веде до зростання видатків на утримання департаментів зв'язку з клієнтами та зниження рентабельності всього бізнесу.

Завданням цієї роботи є побудова інтелектуальної автоматизованої системи класифікації текстових повідомлень на основі машинного навчання, що дозволяє бізнесу оптимізувати та значно спростити процес ведення комунікації з клієнтами, тим самим зменшити кількість персоналу, необхідного для виконання цієї роботи вручну та зменшити видатки на ведення цієї діяльності.

Мета дослідження – оптимізація процесу обробки великих масивів вхідних повідомлень та зменшення видатків на утримання відділів контактування з клієнтами для підприємств.

Завдання дослідження:

– аналіз сучасних методів машинного навчання, які можуть бути використані для обробки текстових даних;

– проведення аналізу методів машинного навчання на предмет їх ефективності;

– побудова інтелектуальної системи класифікації вхідних повідомлень на основі методів машинного навчання для підприємств.

Об’єкт дослідження – процес комунікації з клієнтами у технологічних компаніях та компаніях фінансового сектору.

Предмет дослідження – сучасні методи машинного навчання, алгоритми класифікації.

Наукова новизна одержаних результатів полягає оптимізації існуючих процесів комунікації з клієнтами у компаніях за рахунок впровадження автоматизованої системи обробки та класифікації вхідних повідомлень та запитів, що дозволяє зменшити кошториси компаній на утримання контакт-центрів.

ОПТИМІЗАЦІЯ, КЛАСИФІКАЦІЯ, МАШИННЕ НАВЧАННЯ, ШТУЧНИЙ ІНТЕЛЕКТ, ПОВІДОМЛЕННЯ, КЛІЄНТ, ТЕХНІЧНА ПІДТРИМКА, КОНТАКТУВАННЯ З КЛІЄНТАМИ

ABSTRACT

Master's thesis: 104 pages, 23 figures, 27 tables, 48 sources, 1 appendix.

Relevance: Nowadays, doing business usually involves mandatory contact with customers or users in one form or another. Most modern companies, whose activities are related to information technologies, service provision, banking, etc., have divisions or departments in their composition that perform the functions of providing technical support, accepting complaints about the quality of service, or performing any other type of contact with consumers

That is why, as the business develops and grows, companies are forced to attract more and more employees to serve the needs of customers online, which in turn leads to an increase in the cost of maintaining customer contact departments and a decrease in the profitability of the entire business.

The purpose of this work is to build an intelligent automated text message classification system based on machine learning, which allows businesses to optimize and greatly simplify the process of communicating with customers, thereby reducing the number of personnel required to perform this work manually and reducing the costs of conducting this activity.

The purpose of the research is to optimize the process of processing large volumes of incoming messages and reduce the costs of maintaining customer contact departments for enterprises.

To achieve the goal of research the following steps must be taken:

– analysis of modern machine learning methods that can be used to process text data;

– analysis of machine learning methods for their effectiveness;

– building an intelligent system for classifying incoming messages based on machine learning methods for enterprises.

The object of research is the process of communication with clients in technology companies and companies of the financial sector.

The subject of research is modern methods of machine learning, classification algorithms.

The scientific novelty of the obtained results lies in the optimization of the existing processes of communication with customers in companies due to the introduction of an automated system for processing and classifying incoming messages and requests, which allows reducing the company's estimates for maintaining contact centers.

OPTIMIZATION, CLASSIFICATION, MACHINE LEARNING, ARTIFICIAL INTELLIGENCE, MESSAGES, CUSTOMER, TECHNICAL SUPPORT, CUSTOMER COMMUNICATIONS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення.

RF – алгоритм машинного навчання random forest.

NLP – natural language processing.

БД – база даних.

HTTP – протокол передачі даних (HyperText Transfer Protocol).

API – інтерфейс прикладного програмування.

SVC – класифікатор опорних векторів.

ВСТУП

У зв'язку з бурхливим розвитком мережі Інтернет проблема ефективної організації доступу до інформації стає все більш актуальною. На сьогоднішній день майже всі компанії переходять на безпаперову передачу інформації за допомогою корпоративних мереж, яка допомагає знижувати фінансові та часові витрати.

На сьогоднішній день багато компаній використовують як корпоративні повідомлення не тільки поштові сервери, але і Web-додатки, тому браузерам потрібно приділити особливу увагу. Варто зазначити, що друге місце в рейтингу вразливостей, що використовуються зловмисниками, зайняла категорія “Браузери" (42%), вона включає експлойти для Internet Explorer, Google Chrome, Mozilla Firefox та ін, які найчастіше використовуються для передачі електронних листів до компанії

На сьогоднішній розвиток методів класифікації інформації залишається актуальним завданням наукових досліджень, об'єктом яких стає інформаційне та програмне забезпечення захисту поштових сервісів від небажаної поштової кореспонденції, предмет-методи та алгоритми класифікації текстової інформації, межі дослідження-електронна пошта корпоративної мережі.

Метою цієї роботи є розробка автоматизованої системи класифікації листів.

До завдань роботи віднесено:

– аналіз методів класифікації листів;

– розробка вимог до програми автоматичної класифікації листів;

– розробка системи автоматизованої класифікації листів та її опис.

За останні кілька років використання Web-орієнтованих додатків при створенні розподілених гетерогенних інформаційних систем набуло широкого поширення та стало популярним. Однією із завдань підтримки поштових сервісів стає завдання класифікації електронних листів на категорії спам або не спам. Численні спам-розсилки порушують доступність інформаційних

ресурсів, які необхідні клієнтам, т.к. споживають частину ресурсів каналу даних. Також вони спричиняють порушення у разі вірусної загрози або видалення електронного повідомлення при його пошуку, а також подібні розсилки часто призводять до повного або часткового знищення даних або їх спотворення. Ряд вірусних загроз може використовуватися для крадіжки персональних даних: номерів пластикових карток приватних користувачів, імен користувачів та логінів доступу до систем управління особистими кабінетами організацій. Тому спам-фільтрація дуже актуальна та потребує пошуку ефективних рішень.

Так як спам зазвичай відрізняється від звичайної кореспонденції, відомим методом боротьби з ним стає відсіювання (фільтрація) його із вхідних листів. Сьогодні цей підхід застосовується для фільтрації непотрібних повідомлень. Завдання фільтрації спаму розглядають як завдання класифікації вхідних повідомлень на категорії спам або не спам.

У цій роботі розроблено систему автоматичної класифікації електронних листів. Її використання забезпечує:

– підвищення ефективності роботи користувачів (лист може бути віднесений до одного або іншого типу автоматично);

– додаткові сервісні можливості (наприклад, можливість автоматичного сортування листів);

– скорочення витрат на обробку пошти (програмне забезпечення автоматизує їхню класифікацію).

1 АНАЛІЗ ІСНУЮЧИХ СИСТЕМ КЛАСИФІКАЦІЇ

Розглянемо програмні рішення, що вже присутні на ринку.

– Mailytica:

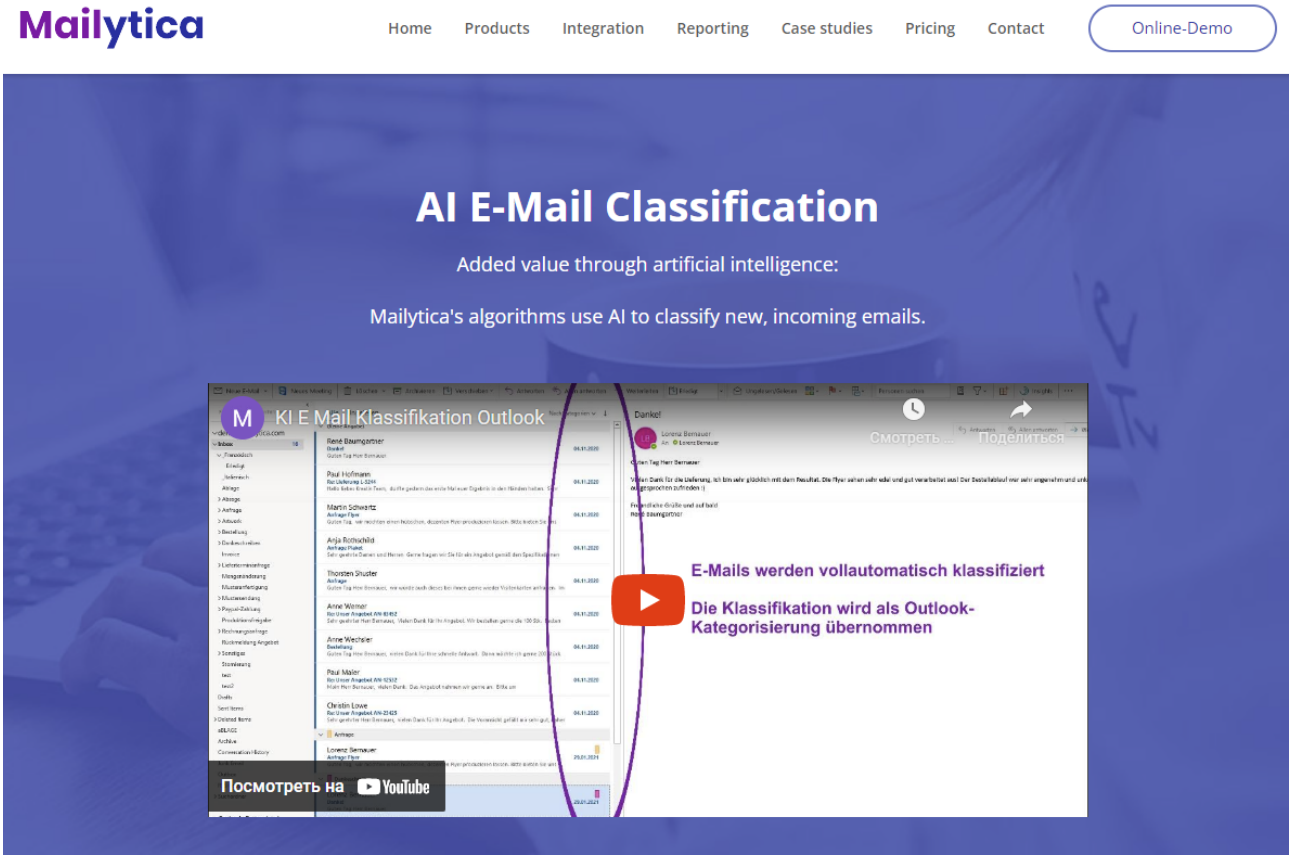


Рисунок 1.1 – Домашня сторінка сервісу Mailytica

Сервіс Mailytica надає розширення для його інтегрування в поштові клієнти Outlook, Gmail, Zendesk, а також сервій має власний API.

Штучний інтелект Mailytica здатний аналізувати вхідні електронні листи та розуміти їх за допомогою сучасних моделей мови NLP. Це означає не що інше, як те, що штучний інтелект може зрозуміти контекст електронних листів і, отже, наміри відправника.

Також сервіс дає можливість налаштувати класифікацію листів. Під класифікацією розуміють ключові слова електронного листа на основі наміру, який він містить. Наприклад, якщо клієнт запитує про поточний статус доставки, класифікація буде «запит на дату доставки». З іншого боку, якщо клієнт скаржиться на доставку, класифікація буде «скарга».

За допомогою автоматичної класифікації можна реалізувати такі випадки використання:

– автоматичне призначення категорій Outlook;

– автоматичне пересилання електронних листів відповідальному співробітнику;

– автоматичне виявлення та пріоритезація критичних електронних листів, таких як скарги;

– автоматичне архівування електронних листів на основі класифікації.

Серед переваг даного сервісу можна виділити його нативну інтеграцію з найпопулярнішими поштовими клієнтами, простоту його налаштування та широкий функціонал, зокрема автоматичне виявлення контексту вхідного листа та можливість автоматичного пересилання цього листа відповідальному працівникові.

Недоліками даного застосування є досить висока його вартість та обмежений функціонал при інтеграції через API з поштовими сервісами, які не є не є нативно інтегрованими з Mailytica;

– emailtree.ai:

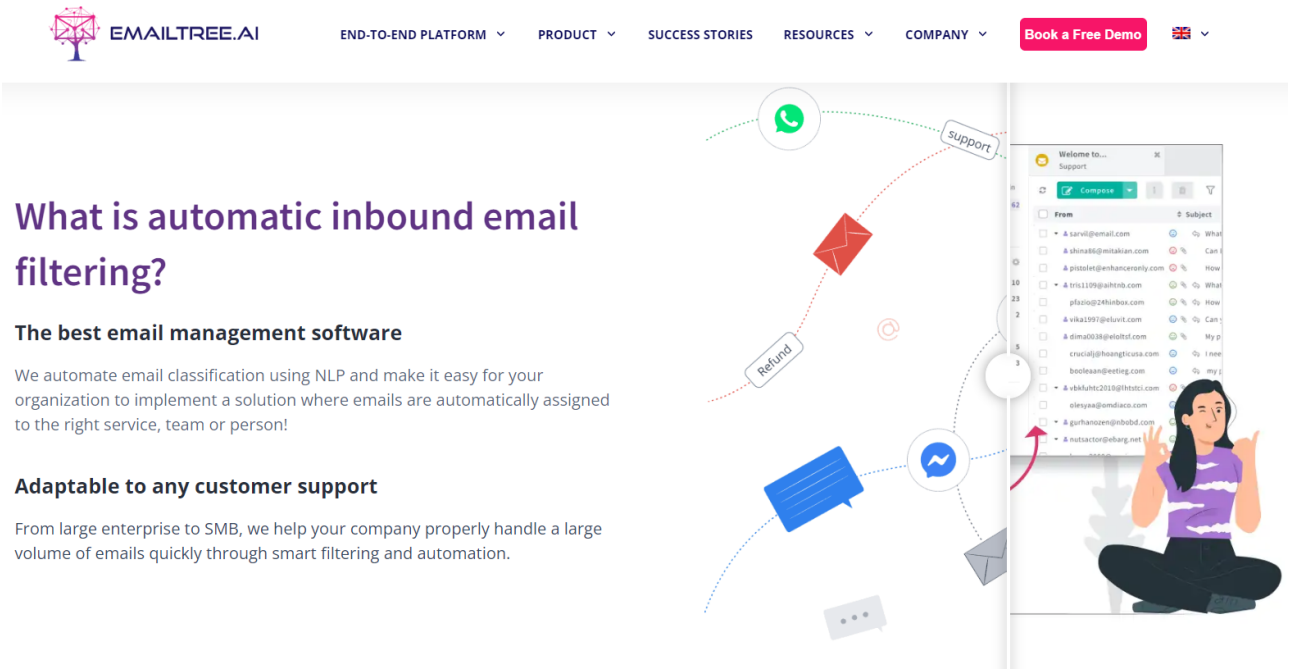


Рисунок 1.2 – Опис сервісу класифікації emailtree.ai

Завдяки штучному інтелекту, на основі якого працює сервіс emailtree, фільтрація електронної пошти та пріоритезація дозволяють заощадити час і підвищити продуктивність працівників.

Використовуючи Action Manager і штучний інтелект, можна за лічені хвилини розробити та автоматизувати маршрутизацію електронних листів до відповідної команди у компанії.

Даний сервіс є досить сильним конкурентом серед аналогічних рішень, оскільки він дозволяє класифікувати повідомлення базуючись на великій кількості різноманітних факторів, наприклад за за мовою, настроєм, наміром або розпізнаванням іменованих об'єктів, що однозначно є перевагою серед інших аналогічних продуктів. Також сервіс emailtree використовує різні алгоритми машинного навчання для обробки листів, що надає йому додаткової точності та є його беззаперечною перевагою.

Недоліком вищезгаданого продукту є те, що він являє собою самостійну систему та не інтегрується з існуючими поштовими аккаунтами та клієнтами, що може бути суттєвим мінусом при ухваленні рішення про його використанні на підприємствах, на яких уже існують налагоджені канали обміну електронною кореспонденцією;

– 360 Protective Marking:

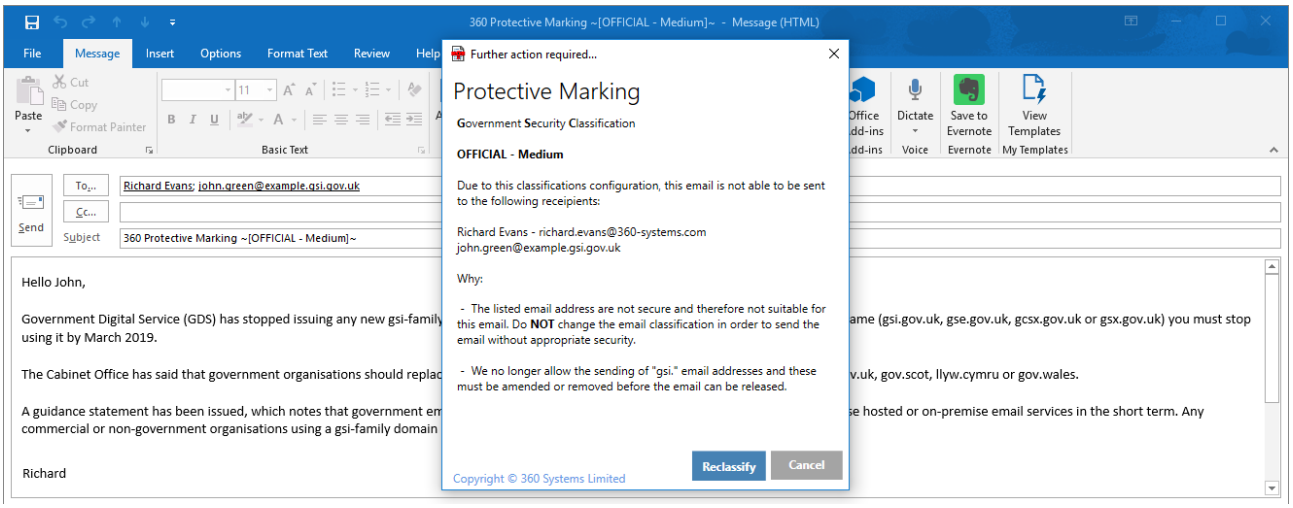


Рисунок 1.3 – Інтерфейс застосунку 360 Protective Marking

360 Protective Marking – це надбудова для Microsoft Office і IBM Notes, яка забезпечує позитивну класифікацію документів/електронних листів під

час збереження, друку та надсилання. Він доступний для електронної пошти Outlook, Word, Excel, PowerPoint, Visio та Notes.

Застосунок дозволяє користувачам класифікувати свої документи та електронні листи під час їх написання. Коли користувач переходить до збереження, друку або надсилання, йому пропонується вибрати класифікацію, яка застосовує метадані до документа чи електронного листа. Параметри класифікації можна пристосувати до потреб організації та можуть включати періоди зберігання поряд зі стандартною класифікацією, яка може використовуватися системою керування записами та документами.

360 Protective Marking можна налаштувати для застосування класифікацій EDRM до метаданих вмісту під час його створення та редагування. Застосовувана інформація часто зумовлена юридичними зобов'язаннями або потребами бізнесу та визначає:

– як використовуються активні документи;

– до якої бібліотеки надходять документи;

– які види документів повинні стати записами;

– класифікуйте записи, які потребують подібного поводження;

– хто є основним власником запису;

– як довго вони повинні зберігатися;

– що з ними відбувається після закінчення періоду зберігання.

Даний застосунок є вендор-специфічним, що означає, що він працює лише з однією програмною платформою, а саме продуктами компанії Microsoft: Outlook, Word, Excel, PowerPoint, Visio та Notes, що беззаперечно є його недоліком у порівнянні з іншими аналогічними продуктами. Також 360 Protective Marking не використовує жодних інтелектуальних систем аналізу вмісту повідомлень, тож можемо вважати дане рішення деякою мірою застарілим та не вартим уваги, адже існують більш сучасні та інтелектуальній його аналоги;

– Klassify:

Рисунок 1.4 – Опис пакету класифікації Classify

Електронна пошта є широко використовуваним інструментом внутрішньої та зовнішньої комунікації. Як і класифікація документів, класифікація електронних листів так само важлива для забезпечення безпеки даних і відповідності.

Застосувавши відповідну класифікацію до електронного листа, можна чітко визначити конфіденційність інформації та забезпечити належну обробку та захист критичних даних від несанкціонованого використання та випадкової втрати.

Модуль класифікації електронної пошти постачається в комплекті з пакетом класифікації даних Klassify. Це означає, що можна класифікувати електронні листи, вкладення та файли на робочому столі за допомогою одного інструменту.

Інструмент Klassify надає широкий перелік функціональних можливостей, серед яких:

– примусова класифікація електронної пошти, а саме керована користувачем, запропонована або автоматична класифікація;

– класифікація вкладень;

– класифікація на основі домену електронної пошти та відправника;

– класифікація вхідної пошти;

– білий/чорний список домену/відправника електронної пошти;

– обмеження на вкладення;

– використовуючи теги, додані до метаданих, можливо підвищити ефективність будь-якого рішення DLP або електронної пошти;

– автоматизований захист IRM файлів і електронних листів на основі класифікації завдяки інтеграції з провідним постачальником IRM.

Отже ми бачимо, що застосунок Klassify вирізняється серед інших наявністю великої кількості функціональних можливостей та можливістю навчання на основі коригувань користувача, що буде позитивно впливати на точність подальшої класифікації. Також, беззаперечною перевагою є можливість налаштування чорних та білих списків відправників.

Найочевиднішим недоліком даного рішення є те, що він є частиною пакету аналізу даних, що позбавляє користувача можливості придбати лише даний конкретний інструмент. Також, серед недоліків можна виділити відносну складність початкового налаштування інструменту.

Висновки до розділу

У даному розділі було розглянуто основні з існуючих аналогів програмного забезпечення, розглянуто їх переваги та недоліки.

Проаналізувавши дані, наведені в цьому розділі, можемо зробити висновок, що на ринку існує всього 4 активні гравці, але їх функціонал, як правило, є не надто широким та адаптованим під вирішення якогось одного типу задач.

На основі проведеного аналізу зроблено висновки про те, яким функціоналом має бути наділений майбутній програмний продукт, щоби бути конкурентноспроможним на сучасному ринку програмного забезпечення.

2 ОГЛЯД ТА ВИБІР ОПТИМАЛЬНОГО АЛГОРИТМУ МАШИННОГО НАВЧАННЯ ДЛЯ КЛАСИФІКАЦІЇ

2.1 Загальні засади машинного навчання

Машинне навчання – це галузь штучного інтелекту та інформатики, яка зосереджується на використанні даних і алгоритмів для імітації способу навчання людей, поступово покращуючи його точність.

За останні кілька десятиліть технологічний прогрес у сховищах і обчислювальних потужностях уможливив деякі інноваційні продукти на основі машинного навчання, такі як система рекомендацій Netflix і безпілотні автомобілі.

Машинне навчання є важливою складовою зростаючої галузі науки про дані. Завдяки використанню статистичних методів алгоритми навчаються робити класифікації або прогнози, а також виявляти ключові ідеї в проектах інтелектуального аналізу даних. Ця інформація згодом керує прийняттям рішень у програмах і бізнесі, ідеально впливаючи на ключові показники зростання. Оскільки великі дані продовжують розширюватися та зростати, ринковий попит на науковців з обробки даних зростатиме. Вони повинні будуть допомогти визначити найбільш релевантні бізнес-питання та дані для відповіді на них.

Алгоритми навчання працюють на основі того, що стратегії, алгоритми та умовиводи, які добре працювали в минулому, ймовірно, продовжуватимуть добре працювати в майбутньому. Ці висновки можуть бути очевидними, наприклад, «оскільки сонце сходило щоранку протягом останніх 10 000 днів, воно, ймовірно, зійде і завтра вранці». Вони можуть бути відтінками, наприклад «X% сімейств мають географічно окремі види з варіантами забарвлення, тому існує Y% шансів, що існують невідомі чорні лебеді ».

Програми машинного навчання можуть виконувати завдання без явного програмування на це. Він передбачає, що комп’ютери навчаються на основі наданих даних для виконання певних завдань. Для простих завдань,

призначених комп'ютерам, можна запрограмувати алгоритми, які вказують машині, як виконати всі кроки, необхідні для вирішення поточної проблеми; з боку комп'ютера навчання не потрібне. Для більш складних завдань людині може бути важко вручну створити необхідні алгоритми. На практиці може виявитися ефективнішим допомогти машині розробити власний алгоритм, а не доручати людям-програмістам вказувати кожен необхідний крок.

Дисципліна машинного навчання використовує різні підходи, щоб навчити комп'ютери виконувати завдання, де немає повністю задовільного алгоритму. У випадках, коли існує величезна кількість потенційних відповідей, один підхід полягає в тому, щоб позначити деякі з правильних відповідей як дійсні. Потім це можна використовувати як навчальні дані для комп'ютера, щоб покращити алгоритм(и), які він використовує для визначення правильних відповідей. Наприклад, для навчання системи задачі розпізнавання цифрових символів часто використовувався набір рукописних цифр MNIST.

2.2 Підходи до машинного навчання

Підходи до машинного навчання традиційно поділяються на три великі категорії, які відповідають парадигмам навчання, залежно від характеру «сигналу» або «зворотного зв'язку», доступного для системи навчання:

– **контрольоване навчання:** комп'ютеру надаються приклади вхідних даних і бажаних виходів, наданих «вчителем», і мета полягає в тому, щоб вивчити загальне правило, яке відображає вхідні дані та виходи;

– **неконтрольоване навчання:** алгоритму навчання не призначаються мітки, тому він сам знаходить структуру вхідних даних. Навчання без нагляду може бути самоціллю (виявлення прихованих шаблонів у даних) або засобом досягнення мети (навчання функцій);

– **навчання з підкріпленням:** комп'ютерна програма взаємодіє з динамічним середовищем, у якому вона повинна виконати певну мету (наприклад, керувати транспортним засобом або грати проти суперника).

Переміщаючись у просторі проблем, програмі надається зворотний зв’язок, аналогічний винагородам, які вона намагається максимізувати.

2.2.1 Контрольоване навчання

Алгоритми навчання під наглядом створюють математичну модель набору даних, яка містить як вхідні, так і бажані вихідні дані. Дані відомі як навчальні дані та складаються з набору навчальних прикладів. Кожен навчальний приклад має один або кілька входів і бажаний вихід, також відомий як контрольний сигнал. У математичній моделі кожен навчальний приклад представлений масивом або вектором, який іноді називають вектором ознак, а навчальні дані представлені матрицею. Завдяки ітераційній оптимізації цільової функції алгоритми керованого навчання вивчають функцію, яку можна використовувати для прогнозування результату, пов’язаного з новими входами. Оптимальна функція дозволить алгоритму правильно визначити вихід для вхідних даних, які не були частиною навчальних даних. Кажуть, що алгоритм, який з часом покращує точність своїх виходів або прогнозів, навчився виконувати це завдання.

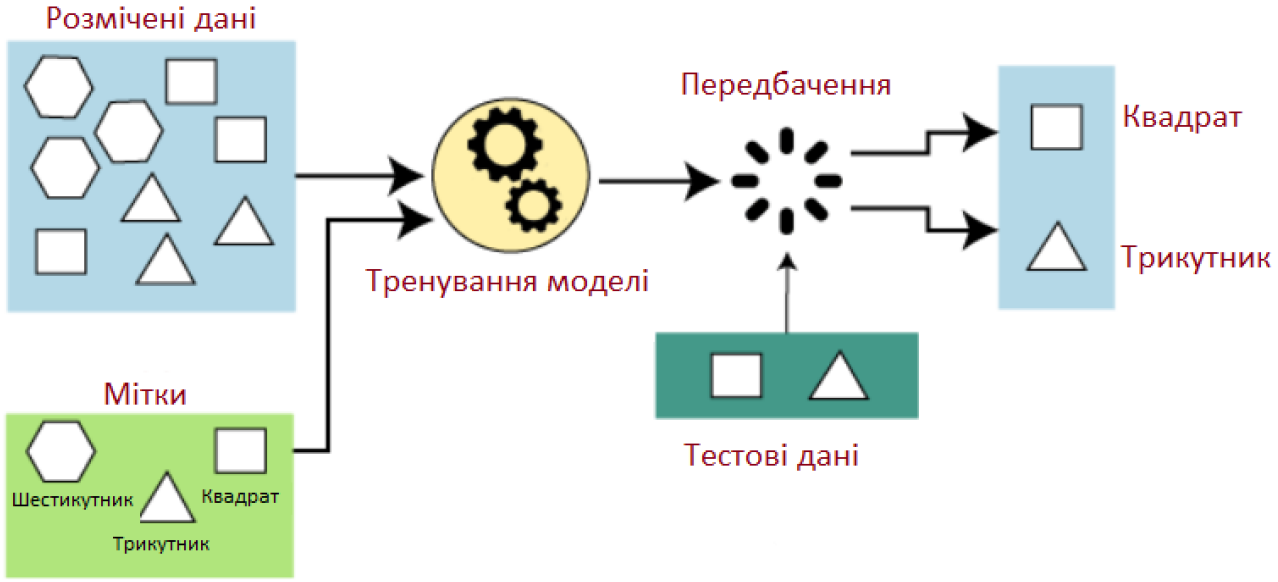


Рисунок 2.1 – Схема контрольованого навчання

Типи алгоритмів навчання під наглядом включають активне навчання, класифікацію та регресію. Алгоритми класифікації використовуються, коли

вихідні дані обмежені обмеженим набором значень, а алгоритми регресії використовуються, коли вихідні дані можуть мати будь-яке числове значення в межах діапазону. **Як приклад, для алгоритму класифікації, який фільтрує електронні листи, вхідними даними буде вхідний електронний лист, а вихідними – ім'я папки, у яку потрібно зберегти електронний лист.**

Навчання за подібністю – це сфера керованого машинного навчання, яка тісно пов'язана з регресією та класифікацією, але мета полягає в тому, щоб вчитися на прикладах за допомогою функції подібності, яка вимірює, наскільки схожі або пов'язані два об'єкти. Він має застосування для рейтингу, систем рекомендацій, відстеження візуальної ідентифікації, перевірки обличчя та перевірки мовця.

2.2.2 Неконтрольоване навчання

Алгоритми неконтрольованого навчання беруть набір даних, який містить лише вхідні дані, і знаходять у даних структуру, як-от групування чи кластеризацію точок даних. Таким чином, алгоритми вивчають тестові дані, які не були позначені, класифіковані чи класифіковані. Замість того, щоб реагувати на відгуки, алгоритми неконтрольованого навчання виявляють спільні риси в даних і реагують на наявність або відсутність таких спільних рис у кожній новій частині даних. Основним застосуванням неконтрольованого навчання є оцінка щільності в статистиці, наприклад, визначення функції щільності ймовірності. Хоча неконтрольоване навчання охоплює інші сфери, що включають узагальнення та пояснення характеристик даних.

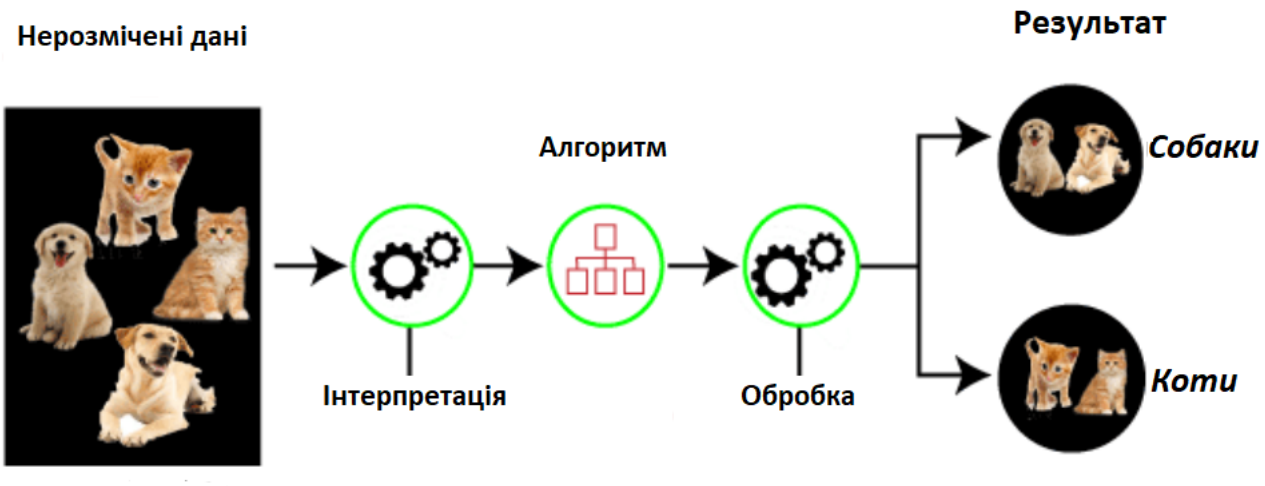


Рисунок 2.2 – Схема неконтрольованого навчання

Кластерний аналіз – це розподіл набору спостережень на підмножини (які називаються кластерами), щоб спостереження в межах одного кластера були подібними відповідно до одного чи кількох заздалегідь визначених критеріїв, тоді як спостереження, отримані з різних кластерів, були несхожими. Різні методи кластеризації роблять різні припущення щодо структури даних, які часто визначаються деякими показниками подібності та оцінюються, наприклад, внутрішньою компактністю, або подібністю між членами одного кластера, і розділенням, різницею між кластерами. Інші методи базуються на оцінці щільності та зв'язності графів.

2.2.3 Навчання з підкріпленням

Навчання з підкріпленням – це область машинного навчання, яка займається тим, як розумні агенти повинні виконувати дії в середовищі, щоб максимізувати поняття сукупної винагороди.

Навчання з підкріпленням відрізняється від навчання під наглядом тим, що не потрібно подавати позначені пари введення/виходу, а також не потрібно явно виправляти субоптимальні дії. Натомість увага зосереджена на пошуку балансу між дослідженням (незвіданих територій) і експлуатацією (сучасних знань).

Середовище, як правило, формулюється у формі марківського процесу прийняття рішень (MDP), оскільки багато алгоритмів навчання з підкріпленням для цього контексту використовують методи динамічного програмування. Основна відмінність між класичними методами динамічного програмування та алгоритмами навчання з підкріпленням полягає в тому, що останні не передбачають знання точної математичної моделі MDP, і вони націлені на великі MDP, де точні методи стають неможливими.

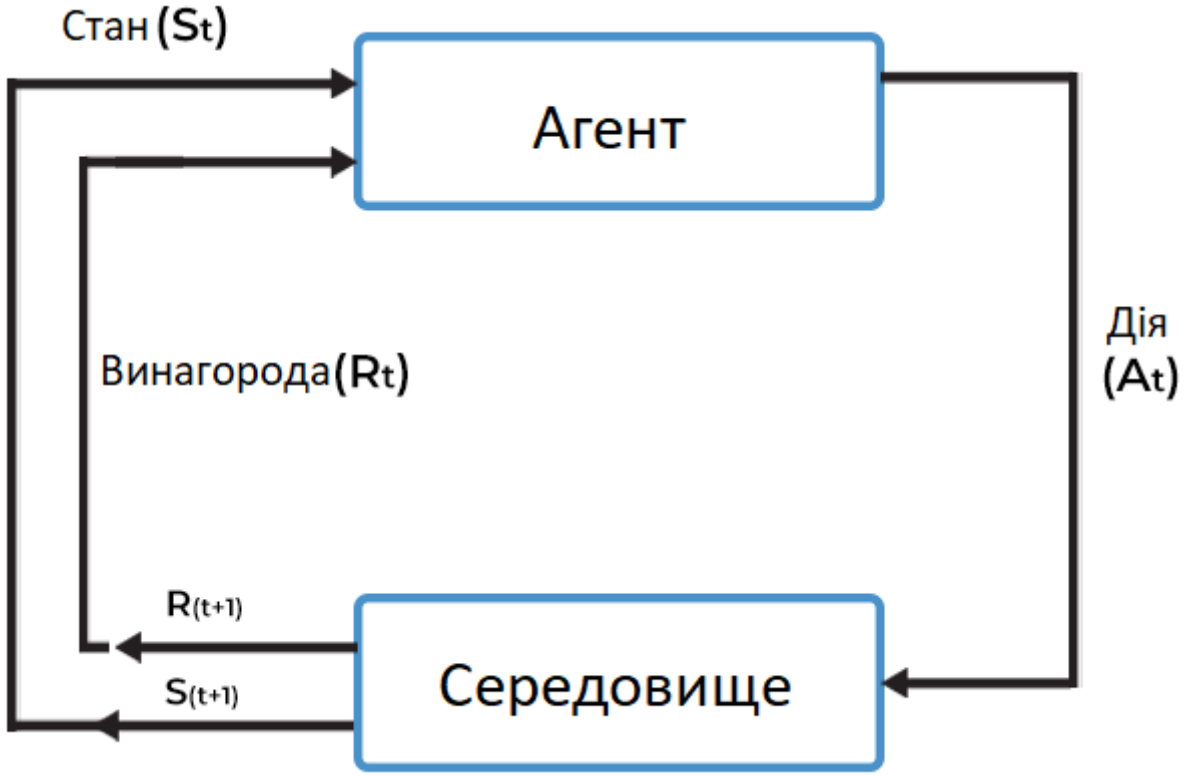


Рисунок 2.3 – Схема навчання з підкріпленням

2.3 Алгоритми, що можуть бути використані для вирішення задачі

Розглянемо деякі з популярних алгоритмів машинного навчання, які є придатними для розв’язання задачі, поставленої у даній роботі.

– логістична регресія: використовує сигмоїдну функцію, наведену вище, щоб повернути ймовірність мітки. Він широко використовується, коли проблема класифікації є двійковою – істинна чи хибна, виграш чи програш, позитивна чи негативна.

Сигмоїдна функція генерує ймовірнісний вихід. Порівнюючи ймовірність із заздалегідь визначеним порогом, об’єкту призначається відповідна мітка;

– дерево рішень: створює гілки дерева в ієрархічному підході, і кожну гілку можна розглядати як оператор if-else. Гілки розвиваються шляхом поділу набору даних на підмножини на основі найважливіших характеристик. Остаточна класифікація відбувається на листках дерева рішень;

– машина підтримки векторів (SVM): знаходить найкращий спосіб класифікувати дані на основі положення по відношенню до межі між позитивним і негативним класами. Ця межа відома як гіперплощина, яка максимізує відстань між точками даних з різних класів. Подібно до дерева рішень і випадкового лісу, машину опорних векторів можна використовувати як для класифікації, так і для регресії, SVC (класифікатор опорних векторів) призначений для задач класифікації.

– k-найближчий сусід (KNN): ми можемо уявити k алгоритм найближчих сусідів як представлення кожної точки даних у розмірному просторі, який визначається n характеристиками. Він обчислює відстань від однієї точки до іншої, а потім призначає мітку неспостережуваних даних на основі міток найближчих спостережуваних точок даних. KNN також можна використовувати для побудови системи рекомендацій.

– наївний Байєс: базується на теоремі Байєса – підході до обчислення умовної ймовірності на основі попередніх знань і наївного припущення, що кожна функція є незалежною одна від одної. Найбільшою перевагою Naive Bayes є те, що, хоча більшість алгоритмів машинного навчання покладаються на велику кількість навчальних даних, він працює відносно добре, навіть якщо розмір навчальних даних невеликий. Наївний байєсівський класифікатор Гауса – це тип наївного байєсовського класифікатора, який дотримується нормального розподілу.

– випадковий ліс (Random forest): як випливає з назви, випадковий ліс – це набір дерев рішень. Це поширений тип ансамблевих методів, які

агрегують результати кількох предикторів. Випадковий ліс додатково використовує техніку пакетування, яка дозволяє кожному дереву навчитися на випадковій вибірці вихідного набору даних і отримати більшість голосів від дерев.Порівняно з деревом рішень воно має краще узагальнення, але менш інтерпретоване, оскільки до моделі додано більше шарів.

У цій роботі, для розв’язання задач даної наукової роботи ми будемо використовувати саме метод random forest (RF), адже він має ряд переваг, порівняно з іншими алгоритмами машинного навчання. Детальне обгрунтування вибору алгоритму буде наведено у наступному розділі цієї наукової роботи.

Висновки до розділу

У даному розділі було детально розглянуто найпопулярніші з існуючих алгоритмів машинного навчання, які можуть бути використані для розв’язання задачі класифікації текстових повідомлень.

Було описано випадки, в яких є оптимальним використання того, чи іншого з них.

Обрано алгоритм машинного навчання, який буде використовуватися для розв’язання задачі сформульованої у даній магістерській роботі, а саме, алгоритм Random forest.

3 ОБҐРУНТУВАННЯ ВИБОРУ АЛГОРИТМУ

Для розв’язання поставленої задачі, а саме, класифікації e-mail повідомлень, було обрано алгоритм випадкових дерев (random forest), оскільки він має ряд переваг, у порівнянні з іншими алгоритмами:

– може виконувати як завдання регресії, так і класифікації;

– випадковий ліс дає гарні прогнози, які легко зрозуміти;

– може ефективно обробляти великі набори даних;

– алгоритм випадкового лісу забезпечує вищий рівень точності прогнозування результатів порівняно з алгоритмом дерева рішень.

Далі розглянемо обраний алгоритм більш детально.

3.1 Загальні відомості

Існує багато методів машинного навчання, які можуть бути використані для реалізації класифікації повідомлень. У даній науковій роботі ми будемо використовувати ансамблевий метод, а саме random forests (випадкові ліси або ліси випадкових рішень) який використовує побудову численних дерев прийняття рішень у процесі тренування моделі і будує моду для класів (класифікацій) або усереднений прогноз (регресія) побудованих дерев. Отож, розглянемо цей метод, його недоліки та переваги більш детально.

Випадкові ліси або ліси випадкових рішень – це метод ансамблевого навчання для класифікації, регресії та інших завдань, який працює шляхом побудови безлічі дерев рішень під час навчання. Для класифікаційних завдань результатом випадкового лісу є клас, обраний більшістю дерев. Для завдань регресії повертається середнє або середнє прогнозування окремих дерев. Випадкові ліси рішень виправляють звичку дерев рішень переобладнувати свій навчальний набір. Випадкові ліси загалом перевершують дерева рішень, але їхня точність нижча, ніж дерева з посиленням градієнта. Однак характеристики даних можуть впливати на їх ефективність.

Перший алгоритм для випадкових лісів рішень був створений у 1995 році Тін Кам Хо з використанням методу випадкового підпростору, який, у

формулюванні Хо, є способом реалізації підходу «стохастичної дискримінації» до класифікації, запропонованого Юджином Клейнбергом.

Розширення алгоритму було розроблено Лео Брейманом і Адель Катлер, які зареєстрували «Випадкові ліси» як торгову марку в 2006 році (станом на 2019 рік належить Minitab, Inc.). Розширення поєднує в собі ідею Бреймана «укладання в мішки» та випадковий вибір функцій, введених спочатку Хо, а пізніше незалежно Амітом і Джеманом для побудови колекції дерев рішень із контрольованою дисперсією.

Випадкові ліси часто використовуються як моделі «чорної скриньки» в бізнесі, оскільки вони генерують обґрунтовані прогнози для широкого діапазону даних, вимагаючи невеликої конфігурації.

Дерева рішень є популярним методом для різних завдань машинного навчання. «Навчання дерева «найближче відповідає вимогам щодо використання готової процедури для інтелектуального аналізу даних», кажуть Хасті та ін., "оскільки він є інваріантним щодо масштабування та різних інших перетворень значень ознак, стійкий до включення нерелевантних функцій і створює моделі, які можна перевірити. Однак вони рідко бувають точними". Зокрема, дерева, які вирощуються дуже глибоко, мають тенденцію вивчати дуже нерегулярні шаблони: вони переповнюють свої навчальні набори, тобто мають низький зсув, але дуже високу дисперсію. Випадкові ліси – це спосіб усереднення кількох глибоких дерев рішень, навчених на різних частинах одного навчального набору, з метою зменшення дисперсії. Це відбувається за рахунок невеликого збільшення зміщення та деякої втрати інтерпретації, але загалом значно підвищує продуктивність кінцевої моделі.

Ліси схожі на об’єднання зусиль алгоритму дерева рішень. Спільна робота багатьох дерев покращує продуктивність одного випадкового дерева. Хоча ліси не дуже схожі, вони дають ефект k-кратної перехресної перевірки.

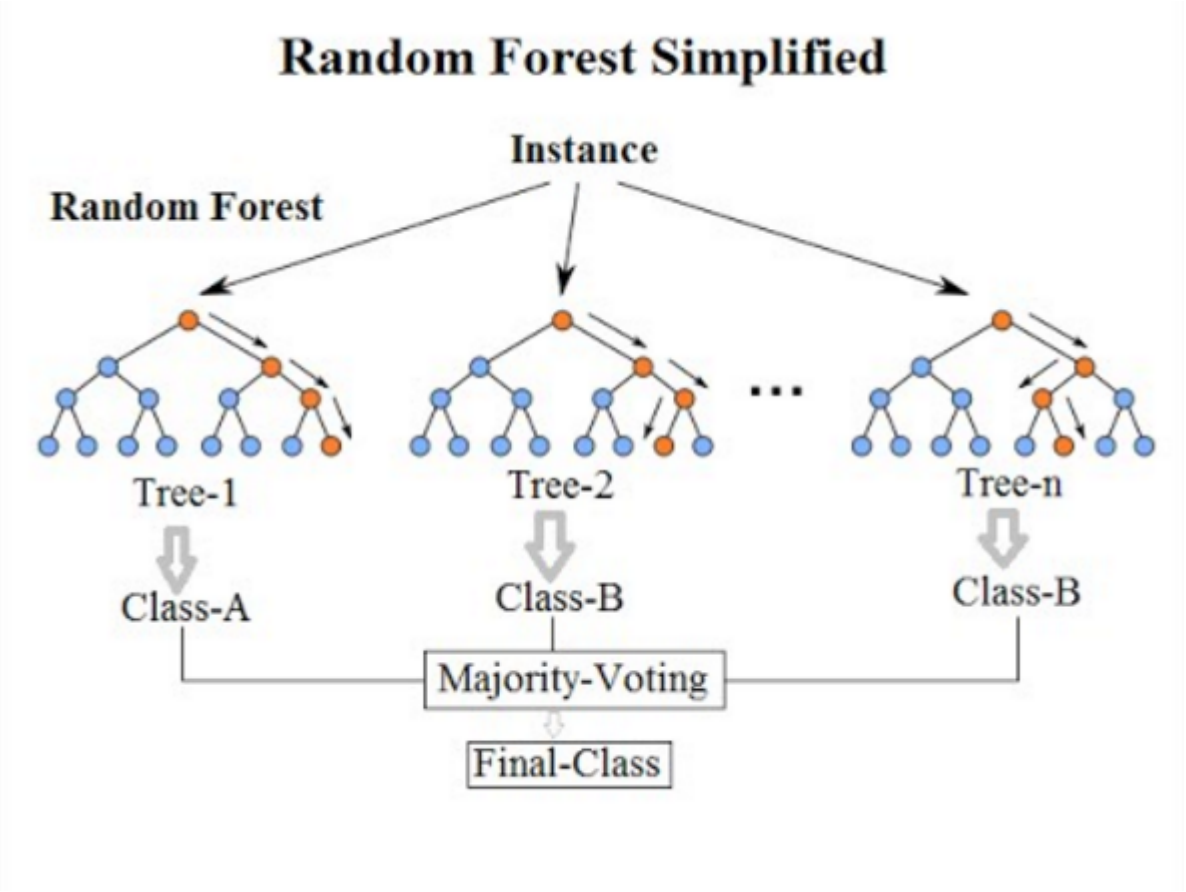


Рисунок 3.1 – Діаграма лісу випадкових рішень

Загальний метод випадкових лісів рішень був вперше запропонований Хо в 1995 році. Хо встановив, що ліси дерев, розбитих похилими гіперплощинами, можуть отримати точність, коли вони ростуть, не страждаючи від перенавчання, якщо ліси випадково обмежені, щоб бути чутливими лише для обраних розмірів функції. Даі працюючи в тому ж напрямку зробив висновок, що інші методи розбиття поводяться подібним чином, якщо вони випадковим чином змушені бути нечутливими до деяких розмірів ознак. Це спостереження більш складного класифікатора (більшого лісу), що стає більш точним майже монотонно, різко суперечить загальноприйнятій думці, що складність класифікатора може зрости лише до певного рівня точності, перш ніж постраждає від перенавчання. Пояснення стійкості методу лісу до перетренованості можна знайти в теорії стохастичного розрізнення Клейнберга. На ранній розвиток поняття випадкових лісів Бреймана вплинула робота Аміта та Джемана, які представили ідею пошуку випадкової підмножини доступних рішень під час

розбиття вузла в контексті вирощування одного дерева. Ідея випадкового вибору підпростору від Хо також мала вплив на проектування випадкових лісів. У цьому методі вирощується ліс дерев, і вводиться варіація серед дерев шляхом проектування навчальних даних у випадково вибраний підпростір перед підгонкою кожного дерева чи кожного вузла. Нарешті, ідея рандомізованої оптимізації вузла, де рішення на кожному вузлі вибирається за допомогою рандомізованої процедури, а не детермінованої оптимізації, була вперше представлена Томасом Г. Дітеріхом.

Правильне введення випадкових лісів було зроблено в статті Лео Бреймана. У цьому документі описано метод побудови лісу некорельованих дерев за допомогою процедури, подібної до CART, у поєднанні з рандомізованою оптимізацією вузлів і пакетуванням. Крім того, ця стаття поєднує в собі кілька компонентів, деякі з яких були раніше відомі, а деякі – нові, які складають основу сучасної практики випадкових лісів, зокрема:

– використання вихідної помилки як оцінки помилки узагальнення;

– вимірювання важливості змінної через перестановку.

Доповідь також пропонує перший теоретичний результат для випадкових лісів у формі обмеження помилки узагальнення, яка залежить від міцності дерев у лісі та їх кореляції.

3.2 Алгоритм

3.2.1 Навчання дерева рішень

Дерева рішень є популярним методом для різних завдань машинного навчання. Зокрема, дерева, які вирощуються дуже глибоко, мають тенденцію вивчати дуже нерегулярні шаблони: вони переповнюють свої навчальні набори, тобто мають низький зсув, але дуже високу дисперсію. Випадкові ліси – це спосіб усереднення кількох глибоких дерев рішень, навчених на різних частинах одного навчального набору, з метою зменшення дисперсії. Це відбувається за рахунок невеликого збільшення зміщення та деякої втрати інтерпретації, але загалом значно підвищує продуктивність кінцевої моделі.

Ліси схожі на об’єднання зусиль алгоритму дерева рішень. Спільна робота багатьох дерев покращує продуктивність одного випадкового дерева. Хоча ліси не дуже схожі, вони дають ефект k-кратної перехресної перевірки.

3.2.2 Пакування

Алгоритм навчання для випадкових лісів застосовує загальну техніку завантажувального агрегування або пакетування до тих, хто вивчає дерева. Дано навчальний набір $X = x_1,...,x_n$ з відповідями $Y = y_1,..., y_n$, багаторазове пакетування (B разів) вибирає випадкову вибірку із заміною навчального набору та підбирає дерева до цих зразків:

Для b = 1,..., B:

– зразок, із заміною, n навчальних прикладів з X, Y ; назвемо їх X_b, Y_b;

– тренування дерева класифікації або регресії f_b на X_b, Y_b.

Після навчання можна зробити прогнози для невидимих зразків x' шляхом усереднення прогнозів усіх окремих дерев регресії на x' :

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} f_b(x')$$

Ця процедура початкового завантаження призводить до кращої продуктивності моделі, оскільки вона зменшує дисперсію моделі, не збільшуючи зміщення. Це означає, що в той час як прогнози одного дерева дуже чутливі до шуму в його навчальному наборі, середнє значення багатьох дерев – ні, доки дерева не корельовані. Просте навчання багатьох дерев на одному навчальному наборі дасть сильно корельовані дерева (або навіть те саме дерево багато разів, якщо алгоритм навчання є детермінованим); Завантажувальна вибірка – це спосіб декореляції дерев, показуючи їм різні навчальні набори.

Крім того, оцінку невизначеності прогнозу можна зробити як стандартне відхилення прогнозів від усіх окремих дерев регресії на x' :

$$\sigma = \sqrt{\frac{\sum_{b=1}^{B}(f_b(x') - \hat{f})^2}{B-1}}.$$

Кількість зразків/дерев, B, є вільним параметром. Як правило, використовується від кількох сотень до кількох тисяч дерев залежно від розміру та характеру навчального набору. Оптимальну кількість дерев B можна знайти за допомогою перехресної перевірки або спостерігаючи за помилкою поза мішком: середня помилка передбачення для кожної навчальної вибірки x_i, використовуючи лише дерева, які не мали x_i у своїй початковій вибірці. Помилка навчання та тестування, як правило, нівелюється після підгонки певної кількості дерев.

Наведена вище процедура описує оригінальний алгоритм пакетування для дерев. Випадкові ліси також включають інший тип схеми пакетування: вони використовують модифікований алгоритм навчання дерева, який вибирає випадкову підмножину ознак у кожному кандидатському розділенні в процесі навчання. Цей процес іноді називають «розміщенням функцій». Причиною для цього є кореляція дерев у звичайній вибірці початкового завантаження: якщо одна або кілька ознак є дуже сильними предикторами для змінної відповіді (цільового виходу), ці функції будуть вибрані в багатьох деревах B, викликаючи їх стати корельованим. Аналіз того, як мішок і випадкова проекція підпростору сприяють підвищенню точності за різних умов, дає Хо.

Як правило, для задачі класифікації з p ознаками $\sqrt{p}$ (округлено вниз) ознаки використовуються в кожному розділенні. Для задач регресії винахідники рекомендують p/3 (з округленням до меншого) з мінімальним розміром вузла 5 як за замовчуванням. На практиці найкращі значення цих параметрів слід налаштовувати в кожному окремому випадку для кожної проблеми.

3.2.3 ExtraTrees

Додавання ще одного кроку рандомізації дає надзвичайно рандомізовані дерева, або ExtraTrees. Хоча вони схожі на звичайні випадкові ліси в тому, що вони являють собою ансамбль окремих дерев, є дві основні відмінності: по-перше, кожне дерево навчається за допомогою всієї навчальної вибірки (а не початкової вибірки), а по-друге, поділ зверху вниз учень дерева рандомізований. Замість обчислення локально оптимальної граничної точки для кожної функції, що розглядається (на основі, наприклад, приросту інформації або домішки Джіні), випадково вибрано точку зрізу. Це значення вибирається з рівномірного розподілу в межах емпіричного діапазону функції (у навчальному наборі дерева). Потім з усіх випадково згенерованих поділів для поділу вузла вибирається поділ, який дає найвищий бал. Подібно до звичайних випадкових лісів, можна вказати кількість випадково вибраних ознак, які слід враховувати на кожному вузлі. Значення за замовчуванням для цього параметра: $\sqrt{p}$ для класифікації i_p для регресії, де p - кількість функцій у моделі.

3.3 Властивості алгоритму

3.3.1 Змінна властивість

Випадкові ліси можна використовувати для ранжирування важливості змінних у регресії або проблемі класифікації природним способом. Наступна техніка була описана в оригінальній статті Бреймана і реалізована в пакеті R randomForest.

Перший крок у вимірюванні важливості змінної в наборі даних $\mathcal{D}_n = \{(X_i, Y_i)\}_{i=1}^n$ полягає у підгонці випадкового лісу до даних. Під час процесу підгонки помилка поза мішком для кожної точки даних записується та усереднюється по лісу (помилки незалежного тестового набору можна замінити, якщо під час навчання не використовується мішок).

Щоб виміряти важливість j-тої ознаки після навчання, значення в j-тій ознаці переставляється серед навчальних даних, і помилка поза пакетом знову

обчислюється на цьому збуреному наборі даних. Оцінка важливості для j-тої властивості обчислюється шляхом усереднення різниці помилок поза мішком до та після перестановки по всіх деревах. Оцінка нормалізується стандартним відхиленням цих різниць.

Функції, які дають великі значення для цієї оцінки, вважаються більш важливими, ніж функції, які дають малі значення.

Цей метод визначення важливості змінної має деякі недоліки. Для даних, що містять категоріальні змінні з різною кількістю рівнів, випадкові ліси зміщуються на користь атрибутів із більшою кількістю рівнів. Для вирішення проблеми можна використовувати такі методи, як часткові перестановки і вирощування незміщених. Якщо дані містять групи корельованих ознак, що мають однакову релевантність для результату, то меншим групам надається перевага перед більшими групами.

3.3.2 Відносини до найближчих сусідів

Взаємозв'язок між випадковими лісами та алгоритмом k - найближчого сусіда (k -NN) був відзначений Ліном і Джеоном у 2002 році. Виявляється, що обидва можна розглядати, як так звані схеми зважених околиць. Це моделі, побудовані з навчального набору $\{(x_i, y_i)\}_{i=1}^{n}$ які роблять прогнози $\hat{y}$ для нових точок x', розглядаючи "околиці" точки, формалізовані ваговою функцією W:

$$\hat{y} = \sum_{i=1}^{n} W(x_i, x')\, y_i.$$

Тут $W(x_i, x')$ є невід’ємною вагою і-тої точки навчання відносно нової точки x' у тому самому дереві. Для будь-якого конкретного x' ваги балів x_i сума має дорівнювати одиниці. Вагові функції задані таким чином:

– у k-NN ваги є $W(x_i, x') = \frac{1}{k}$, якщо x і є однією з k точок, найближчих до x', і нуль в іншому випадку;

– на дереві, $W(x_i, x') = \frac{1}{k'}$, якщо x і є однією з k' точок у тому самому аркуші, що й x', і нуль в іншому випадку;

Оскільки ліс усереднює прогнози набору з m дерев з індивідуальними ваговими функціями W_j, його прогнози є

$$\hat{y} = \frac{1}{m}\sum_{j=1}^{m}\sum_{i=1}^{n} W_j(x_i, x')\, y_i = \sum_{i=1}^{n}\left(\frac{1}{m}\sum_{j=1}^{m} W_j(x_i, x')\right) y_i.$$

.

Це показує, що весь ліс знову є зваженою схемою сусідства з вагами, які усереднюють ваги окремих дерев. Сусідами x' у цій інтерпретації є точки x_i спільне використання одного листя на будь-якому дереві $_j$. Таким чином, околиці x' складним чином залежать від структури дерев і, таким чином, від структури навчального набору. Лін і Чон показують, що форма околиці, яку використовує випадковий ліс, адаптується до місцевої важливості кожної функції.

3.4 Неконтрольоване навчання з випадковими лісами

Як частина їх побудови, випадкові прогнози лісу природно призводять до вимірювання відмінності між спостереженнями. Можна також визначити випадкову міру відмінності лісу між немічеными даними: ідея полягає в тому, щоб побудувати випадковий предиктор лісу, який відрізняє «спостережувані» дані від відповідним чином згенерованих синтетичних даних. Спостережувані дані є вихідними немаркованими даними, а синтетичні дані взяті з еталонного розподілу. Випадкова відмінність лісу може бути привабливою, оскільки вона дуже добре обробляє змішані типи змінних, є інваріантною до монотонних перетворень вхідних змінних і стійка до віддалених спостережень. Неподібність випадкового лісу легко справляється з великою кількістю напівнеперервних змінних завдяки своєму внутрішньому вибору змінних; наприклад, несхожість випадкового лісу "Addcl 1" зважує внесок кожної змінної відповідно до того, наскільки вона залежить від інших змінних. Випадкову відмінність лісу використовували в різноманітних

програмах, наприклад, для пошуку кластерів пацієнтів на основі даних тканинних маркерів.

3.5 Недоліки

Хоча випадкові ліси часто досягають вищої точності, ніж одне дерево рішень, вони жертвують внутрішньою інтерпретабельністю, наявною в деревах рішень. Дерева рішень належать до досить невеликого сімейства моделей машинного навчання, які легко інтерпретувати разом із лінійними моделями, моделями на основі правил і моделями на основі уваги. Така можливість інтерпретації є однією з найбільш бажаних якостей дерев рішень. Це дозволяє розробникам підтвердити, що модель отримала реалістичну інформацію з даних, і дозволяє кінцевим користувачам мати довіру та впевненість у рішеннях, прийнятих моделлю. Наприклад, слідувати шляху, який проходить дерево рішень для прийняття рішення, досить тривіально, але слідувати шляхами десятків чи сотень дерев набагато важче. Щоб досягти як продуктивності, так і інтерпретації, деякі методи стиснення моделі дозволяють трансформувати випадковий ліс у мінімальне «відроджене» дерево рішень, яке точно відтворює ту саму функцію прийняття рішень. Якщо встановлено, що прогностичні атрибути лінійно корельовані з цільовою змінною, використання випадкового лісу може не підвищити точність базового учня. Крім того, у задачах із кількома категоріальними змінними випадковий ліс може не підвищити точність базового учня.

Висновки до розділу

В даному розділі було описано та проаналізовано теоретичні основи алгоритму машинного навчання Random forest, розглянуто його математичну базу та обгрунтовано його вибір для побудови системи класифікації повідомлень.

Проаналізувавши всі переваги та недоліки алгоритму, зроблено висновок, що саме він найбільш відповідає технічним вимогам майбутнього

програмного застосунку, адже забезпечує оптимальну швидкість роботи, є не надто вимогливим до апаратного забезпечення інфраструктури та дає високу точність класифікації(передбачення), адже даний алгоритм був розроблений саме для роботи з текстовими даними. Всі перелічені чинники дають можливість побудувати застосунок, який буде конкурентоспроможним на ринку програмного забезпечення.

4 ОПИС ПРОГРАМНОГО ТА АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до програмного продукту

Програмний продукт має лише одного актора - користувача. Отже, опишемо функціональні вимоги до програмного забезпечення з огляду на потреби потенціального користувача.

Таблиця 4.1 – Функціональні вимоги до ПЗ

Актор	**Функція**	**Функціональна вимога**	**Пріоритет**
Користувач	Вхід в систему	Користувач повинен мати можливість авторизуватися у системі за допомогою свого облікового запису електронної пошти, після чого система має отримати токен доступу до облікового запису користувача для подальшого використання існуючих листів дня навчання математичної моделі та класифікації вхідних листів.	Високий

Продовження таблиці 4.1

Актор	**Функція**	**Функціональна вимога**	**Пріоритет**
Користувач	Побудова моделі	Побудова математичної моделі відбувається автоматично, після авторизації користувача, шляхом машинного навчання за допомогою алгоритму Random forest. Передумовою до початку навчання є наявність мінімум 10-ти заздалегідь класифікованих користувачем листів у папці “Вхідні”.	Високий

Продовження таблиці 4.1

Актор	**Функція**	**Функціональна вимога**	**Пріоритет**
Користувач	Класифікація вхідного повідомлення	Класифікація вхідного повідомлення відбувається автоматично, використовуючи алгоритм машинного навчання Random forest та раніше побудованої моделі.	Високий
Користувач	Повторне тренування моделі	Користувач повинен мати можливість повторити тренування моделі використовуючи актуальні дані.	Високий
Користувач	Видалення всіх користувацьких даних	Користувач повинен мати можливість видалити всі персональні дані з системи, включаючи створені математичні моделі.	Високий

4.2 Засоби проектування та розробки

В рамках проектування та розробки інтелектуальної системи, загальні відомості про яку та її опис міститься у попередніх розділах, було використано ряд технологій та фреймворків, за допомогою яких було реалізовано роботу з даними, користувачем та механізм тренування моделей машинного навчання.

Перелік інструментів та технологій, що були використані під час розробки, наведено у таблиці 3.2.

Таблиця 4.2 – Використанні технологій та засоби

Мови програмування	JavaScript(JS), TypeScript(TS), Python
Серверні технології	NodeJS, Flask
Середовища розробки	WebStorm, PyCharm
Фреймворки	NestJS, scikit-learn
Хмарні технологій	Azure Cloud Platform
Системи контролю версій	Git
Інші засоби	Github, Postman, draw.io

Розглянемо деякі з основних технологій більш детально.

4.2.1 Мова програмування Python

Для реалізації машинного навчання було використано мову програмування Python.

Python – це мова програмування високого рівня загального призначення. Її філософія дизайну наголошує на читабельності коду з використанням значних відступів.

Python динамічно типізується та збирає сміття. Він підтримує кілька парадигм програмування, включаючи структуроване (зокрема процедурне), об’єктно-орієнтоване та функціональне програмування.

Гвідо ван Россум почав працювати над Python наприкінці 1980-х як наступником мови програмування ABC і вперше випустив її в 1991 році як Python 0.9.0. Python 2.0 був випущений у 2000 році та представив нові функції, такі як поняття списку, збирання сміття з визначенням циклу, підрахунок посилань і підтримка Unicode. Python 3.0, випущений у 2008 році, був основною версією, яка не повністю сумісна з попередніми версіями. Python 2 було припинено з версією 2.7.18 у 2020 році.

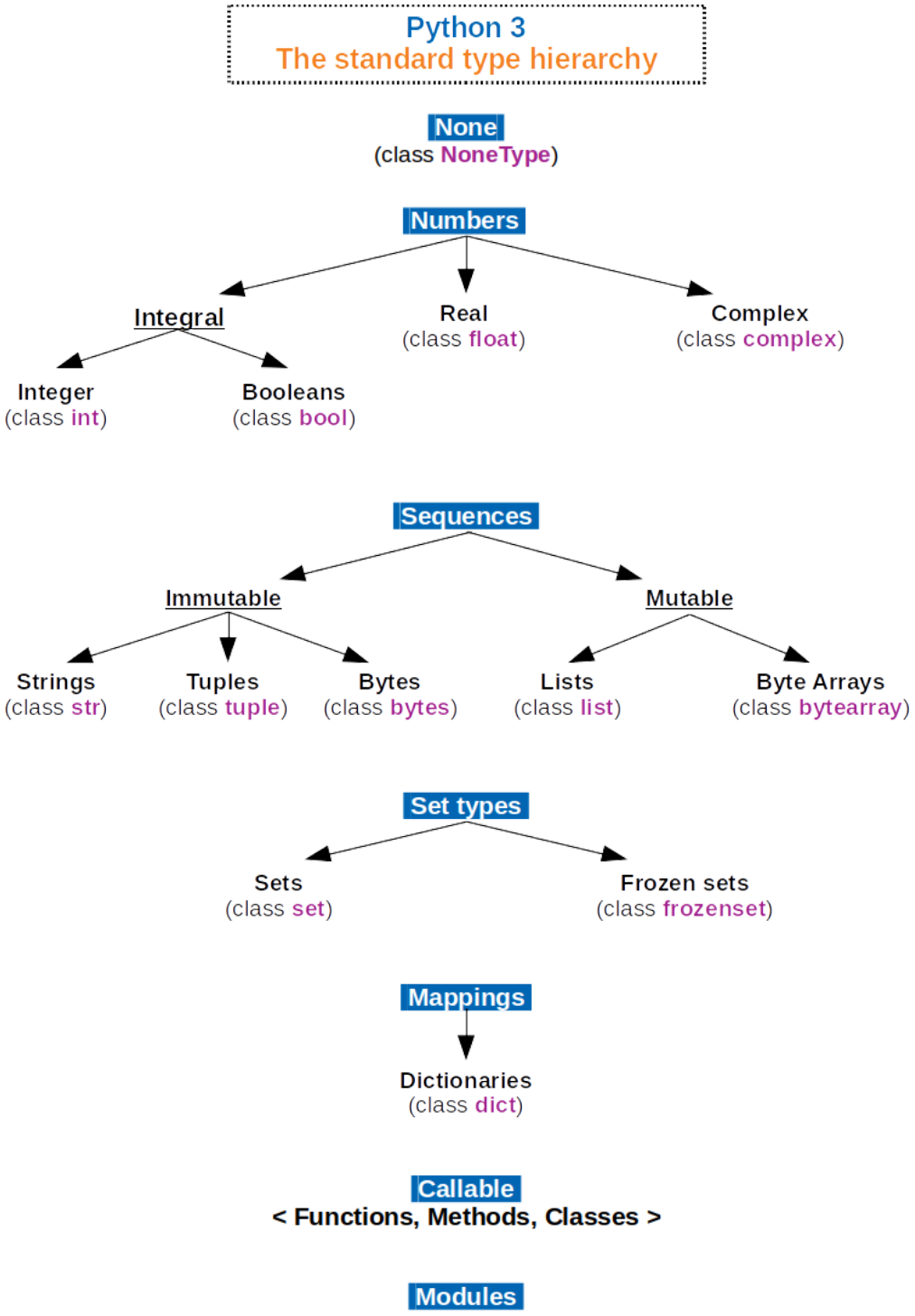


Рисунок 4.1 – Стандартна ієрархія типів у Python 3

Python є мультипарадигмальною мовою програмування. Об’єктно-орієнтоване програмування та структуроване програмування повністю підтримуються, і багато їхніх функцій підтримують функціональне та аспектно-орієнтоване програмування (включаючи метапрограмування та

метаоб’єкти). Багато інших парадигм підтримуються через розширення, включаючи проектування за контрактом і логічне програмування.

Python використовує динамічну типізацію та комбінацію підрахунку посилань і збирача сміття, що визначає цикл, для керування пам’яттю. Він використовує динамічне розпізнавання імен (late binding), яке зв’язує імена методів і змінних під час виконання програми.

Його дизайн пропонує певну підтримку функціонального програмування в традиції Lisp. Він має filter, map і reduce функції; списки, словники, набори та вирази генератора. Стандартна бібліотека має два модулі (itertools і functools), які реалізують функціональні інструменти, запозичені з Haskell і Standard ML.

Замість того, щоб побудувати всю свою функціональність у своєму ядрі, Python був розроблений таким чином, щоб бути розширюваним за допомогою модулів. Ця компактна модульність зробила його особливо популярним як засіб додавання програмованих інтерфейсів до існуючих програм. Бачення Ван Россума невеликої базової мови з великою стандартною бібліотекою та інтерпретатором, що легко розширюється, виникло внаслідок його розчарування ABC, який підтримував протилежний підхід.

Python прагне до простішого, менш захараценого синтаксису та граматики, надаючи розробникам можливість вибору методології кодування. На відміну від девізу Perl «є більше ніж один спосіб зробити це», Python підтримує філософію «повинен бути один – і бажано лише один – очевидний спосіб зробити це». Алекс Мартеллі, співробітник Python Software Foundation і автор книги про Python, написав: «Описати щось як «розумне» не вважається компліментом у культурі Python».

Розробники Python прагнуть уникати передчасної оптимізації та відкидають виправлення некритичних частин еталонної реалізації CPython, які забезпечуватимуть незначне збільшення швидкості за рахунок ясності. Коли швидкість важлива, програміст на Python може перенести критичні за

часом функції до модулів розширення, написаних такими мовами, як C; або використовуйте PyPy, оперативний компілятор. Також доступний Cython, який перекладає сценарій Python на C і здійснює прямі виклики API на рівні C до інтерпретатора Python.

Назва мови програмування «Python» походить від комедійного серіалу BBC «Летючий цирк Монті Пайтона». Гвідо ван Россум подумав, що йому потрібна коротка, унікальна та трохи загадкова назва, тому він вирішив назвати мову програмування «Python».

Поширеним неологізмом у спільноті Python є pythonic, який має широкий діапазон значень, пов'язаних зі стилем програми. «Pythonic» код може добре використовувати ідіоми Python, бути природним або демонструвати вільне володіння мовою, або відповідати мінімалістичній філософії Python і наголосу на читабельності. Код, який важко зрозуміти або читається як груба транскрипція з іншої мови програмування, називається непітонічним.

Користувачів і шанувальників Python, особливо обізнаних і досвідчених, часто називають Pythonistas.

Розробка Python здійснюється в основному за допомогою процесу Python Enhancement Proposal (PEP), основного механізму для пропонування основних нових функцій, збору коментарів спільноти щодо проблем і документування проектних рішень Python. Стиль кодування на Python розглядається в PEP 8. Виняткові PEP переглядаються та коментуються спільнотою Python та керівною радою.

Удосконалення мови відповідає розробці еталонної реалізації CPython. Список розсилки python-dev є основним форумом для розробки мови. Конкретні питання спочатку обговорювалися в інструменті відстеження помилок Roundup, розміщеному фондом. У 2022 році всі проблеми та обговорення перенесено на GitHub. Спочатку розробка велася на власному сховищі вихідного коду під керуванням Mercurial, поки Python не перемістився на GitHub у січні 2017 року.

Публічні випуски CPython бувають трьох типів, які розрізняються за тим, яка частина номера версії збільшується:

– Зворотно несумісні версії, де очікується, що код зламаний і його потрібно перенести вручну. Перша частина номера версії збільшується. Такі випуски трапляються не часто – версія 3.0 була випущена через 8 років після 2.0. За словами Гвідо ван Россума, версія 4.0 навряд чи коли-небудь з'явиться.

– Основні або «функціональні» випуски в основному сумісні з попередньою версією, але містять нові функції. Друга частина номера версії збільшується. Починаючи з Python 3.9, очікується, що ці випуски випускатимуться щорічно. Кожна основна версія підтримується виправленнями помилок протягом кількох років після випуску.

– Випуски виправлень помилок, які не вводять нових функцій, відбуваються приблизно кожні 3 місяці та створюються, коли достатню кількість помилок було виправлено на початковому етапі з моменту останнього випуску. У цих випусках також виправлено вразливості системи безпеки. Третя й остання частина номера версії збільшується.

Багато альфа-, бета-версій і релізів-кандидатів також випускаються як попередній перегляд і для тестування перед остаточними випусками. Хоча для кожного випуску існує приблизний графік, вони часто затримуються, якщо код не готовий. Команда розробників Python відстежує стан коду, запускаючи великий набір модульних тестів під час розробки.

Основна наукова конференція з Python – PyCon. Існують також спеціальні наставницькі програми Python, наприклад Pyladies.

4.2.2 Бібліотека scikit-learn

Scikit-learn – це безкоштовна бібліотека машинного навчання для мови програмування Python. Він містить різноманітні алгоритми класифікації, регресії та кластеризації, включаючи машини опорних векторів, випадкові ліси, посилення градієнта, k - середні та DBSCAN, і розроблений для

взаємодії з чисельними та науковими бібліотеками Python NumPy та SciPy. Scikit-learn – це проект, який фінансується NumFOCUS.

Проект scikit-learn розпочався як scikits.learn, проект Google Summer of Code французького дослідника даних Девіда Курнапо. Його назва походить від уявлення про те, що це «SciKit» (SciPy Toolkit), окремо розроблене та розповсюджене стороннє розширення для SciPy. Оригінальна кодова база була пізніше переписана іншими розробниками. У 2010 році Фабіан Педрегоза, Гаель Варокво, Александр Грамфор і Вінсент Мішель, усі з Французького інституту досліджень комп'ютерних наук і автоматизації в Сакле, Франція, взяли на себе керівництво проектом і зробили перший публічний випуск 1 лютого 2010 року. У листопаді 2012 року scikit-learn і scikit-image з різних scikit були описані як «добре підтримувані та популярні». Scikit-learn – одна з найпопулярніших бібліотек машинного навчання на GitHub станом на 2022 рік.

Scikit-learn здебільшого написаний на Python і широко використовує NumPy для високопродуктивної лінійної алгебри та операцій з масивами. Крім того, деякі основні алгоритми написані на Cython для підвищення продуктивності. Машини опорних векторів реалізовані оболонкою Cython навколо LIBSVM; логістичної регресії та лінійних опорних векторних машин за допомогою подібної оболонки навколо LIBLINEAR. У таких випадках розширення цих методів за допомогою Python може бути неможливим.

Scikit-learn добре інтегрується з багатьма іншими бібліотеками Python, такими як Matplotlib і plotly для побудови графіків, NumPy для векторизації масивів, фрейми даних Pandas, SciPy та багато інших.

4.2.3 Платформа NodeJS

Node.js – це крос-платформне середовище виконання з відкритим вихідним кодом на стороні сервера (але не обмежуючись цим) на основі мови програмування JavaScript, асинхронне, із введенням/виведенням даних у керованій подіями архітектурі та на основі механізму Google V8. Він був

створений, щоб бути корисним у створенні високомасштабованих мережевих програм, таких як веб-сервери. Він був створений Раяном Далем у 2009 році, а його розвиток спонсорує компанія Joyent, у штаті якої також є Дал.

Node.js подібний за призначенням до Python Twisted або Tornado, Perl Object Environment від Perl, libevent або libev від C, EventMachine від Ruby, vibe.d від D і Java EE від Java. На відміну від більшості коду JavaScript, запускається не в браузері, а на сервері. Node.js реалізує деякі специфікації CommonJS. Node.js містить середовище REPL для інтерактивного налагодження.

Node.js працює з однопоточною моделлю оцінки, використовуючи асинхронні входи та виходи, які можуть виконуватися одночасно в кількості до сотень тисяч без витрат, пов’язаних із перемиканням контексту. Цей дизайн спільного використання єдиного потоку виконання між усіма запитами відповідає потребам висококонкурентних програм, у яких кожна операція, яка виконує введення та виведення, повинна мати функцію зворотного виклику. Одним із недоліків цього однопотокового підходу є те, що Node.js потребує додаткових модулів, таких як кластер, щоб масштабувати програму з кількістю процесорних ядер машини, на якій він працює.

V8 – це середовище виконання для JavaScript, створене для Google Chrome. Це безкоштовне програмне забезпечення з 2008 року, воно написано мовою C++ і компілює вихідний код JavaScript у машинний код замість того, щоб інтерпретувати його в реальному часі.

Node.js містить libuv для обробки асинхронних подій. Libuv – це рівень абстракції для роботи в мережі та файлової системи в системах Windows і системах на основі POSIX, таких як Linux, Mac OS X і Unix.

Тіло базових операцій Node.js написано на JavaScript із допоміжними методами, написаними на C++.

Node.js містить кілька «базових модулів», скомпільованих у двійковий файл, наприклад мережевий модуль, який надає рівень для асинхронного мережевого програмування, та інші основні модулі, такі як шлях, файлова

система, буфер, таймери та більш загальні модулі. Можна використовувати модулі, розроблені третіми сторонами, як попередньо скомпільовані файли «.node» або як звичайні файли Javascript. Модулі Javascript реалізовано відповідно до специфікації CommonJS для модулів, використовуючи змінну експорту, щоб надати цим сценаріям доступ до функцій і змінних, реалізованих модулями.

Сторонні модулі можуть розширювати node.js або додавати рівень абстракції, реалізуючи різні утиліти проміжного програмного забезпечення для використання у веб-додатках, таких як фреймворки підключення та експрес. Хоча модулі можна встановлювати як прості файли, вони зазвичай встановлюються за допомогою диспетчера пакетів (npm), який спрощує компіляцію, установку та оновлення модулів, а також керування залежностями. Крім того, модулі, які не встановлено в каталозі модулів Node за замовчуванням, потребуватимуть використання відносного шляху для їх пошуку.

Node.js можна поєднувати з базою даних документів (наприклад, MongoDB або CouchDB) і реляційними базами даних, такими як MySQL, PostgreSQL, серед іншого, дозволяючи вам розробляти в єдиному середовищі розробки JavaScript. З адаптацією шаблонів для розробки на стороні сервера, таких як MVC та його варіанти MVP, MVVM тощо. Node.js дозволяє легко повторно використовувати код з тієї самої моделі інтерфейсу між стороною клієнта та стороною сервера.

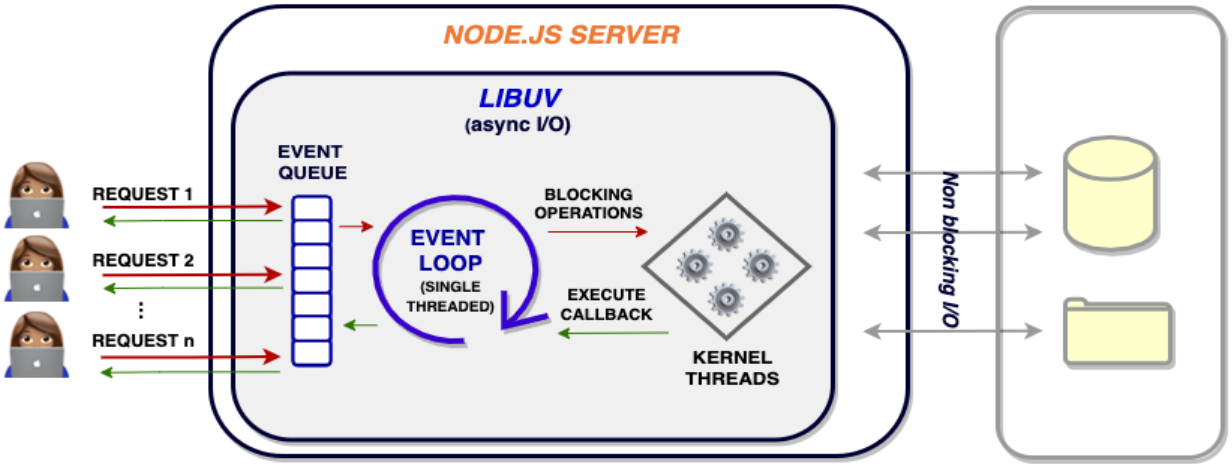


Рисунок 4.2 – Загальна архітектура Node.js сервера

Node.js реєструється в операційній системі, і кожного разу, коли клієнт встановлює з'єднання, виконується зворотний виклик. У середовищі виконання Node.js кожне з'єднання отримує невеликий розподіл простору динамічної пам'яті без необхідності створення потоку. На відміну від інших керованих подіями серверів, цикл обробки подій Node.js не викликається явно, а запускається в кінці кожного виконання функції зворотного виклику. Цикл обробки подій припиняється, коли більше немає подій, до яких потрібно звернути увагу.

4.3 Вимоги до технічного забезпечення

Оскільки дане програмне рішення має багаторівневу архітектуру, то різні його компоненти мають різні вимоги до апаратно-технічного забезпечення.

Загальна структурна схема розгортання застосунку наведена у розділі графічних матеріалів.

Мінімальні технічні характеристики обслуговуючого серверу:

– мінімальна частота процессора - 2,4 ГГц;

– оперативна пам'ять(RAM) мінімальним об'ємом 4 ГБ;

– 5 ГБ вільного простору на накопичувачі;

– Unix/Linux платформа.

Мінімальні технічні характеристики AI-серверу:

– мінімальна частота процессора - 3,6 ГГц;

– оперативна пам’ять(RAM) мінімальним об’ємом 16 ГБ;

– 10 ГБ вільного простору на накопичувачі;

– Unix/Linux платформа.

Технічні вимоги до клієнта:

Оскільки клієнтська частина застосунку повністю працює через веб-браузер, то для клієнта це рішення є кросплатформним. Це означає, що клієнт може використовувати будь-який із сучасних десктопних або мобільних web-браузерів, наприклад Chrome, Opera, Firefox або Safari.

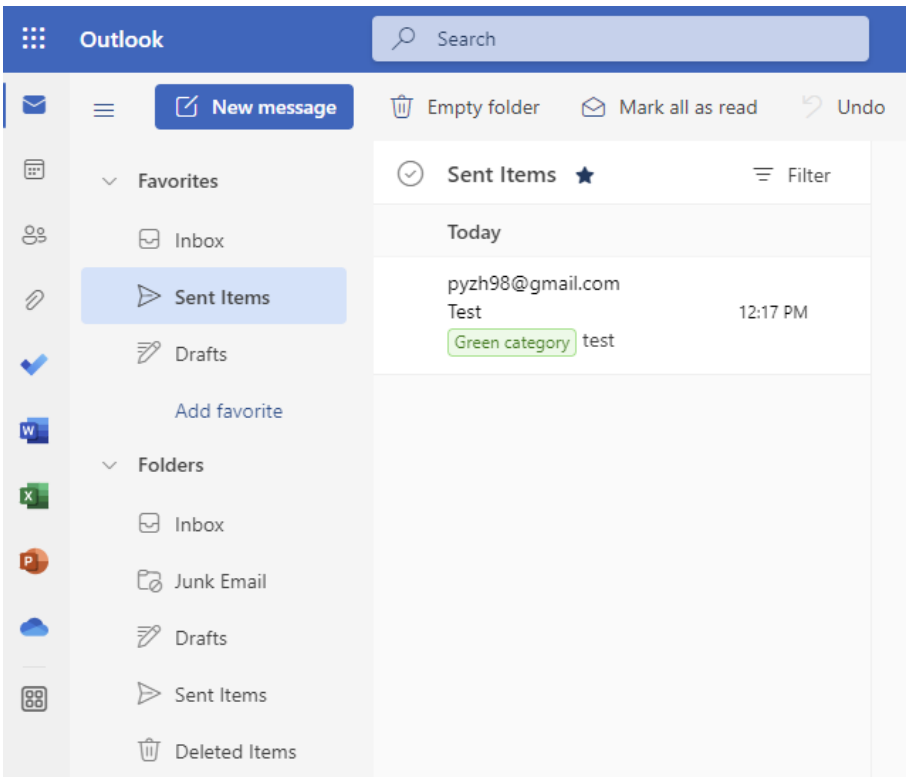


Рисунок 4.4 – Вигляд поштового клієнта Outlook у браузері

4.4 Архітектура програмного забезпечення

При розробці програмного забезпечення було використано шаблон проектування Repository. Використання даного популярного шаблону проектування дозволяє абстрагуватися від явних підключень до баз даних. Шаблон Repository є посередником між бізнес-логікою та базами даних.

Специфікації функцій представлені в наступних таблицях.

Таблиця 4.3 – Функції класу User

Назва	Примітка
public getLoginUrl()	Повертає URL для авторизації користувача через форму логіну поштового сервісу
public handleLogin()	Приймає зворотній виклик після логіну користувача та зберігає дані користувача
public removeUser()	Видаляє дані користувача з системи, зупиняє отримання користувацьких листів системою

Таблиця 4.4 – Функції класу MailApiClient

Назва	Примітка
public getEmailsForTraining()	Надсилає запит на API поштового сервісу, повертає датасет для тренування моделі
public getUserLabels()	Надсилає запит на API поштового сервісу, повертає існуючі мітки користувача
public setLabel()	Встановлює мітку на лист
public handleIncomingMail()	Отримує колбек з поштового клієнта в разі отримання нового листа та відправляє його на класифікацію

Таблиця 4.4 – Функції класу ClassifierClient

Назва	Примітка
public train()	Приймає датасет для тренування, відправляє його до модулю класифікації для побудови моделі
public classify()	Отримує вхідний лист, відправляє його на класифікацію, повертає отриманий клас

Схема структурна класів наведена в графічному матеріалі.

У базі даних містяться дані про наступні сутності:

– користувач;

– побудована математична модель.

Структурна схема бази даних наведена у графічному матеріалі.

Оскільки математичні моделі мають досить великий об’єм, то для їх зберігання використовується Azure Blob Storage, який дозволяє зберігати файли великих об’ємів та мати до них швидкий доступ.

Висновки до розділу

У даному розділі було описано сценарії використання застосунку користувачем, визначено основні технічні та функціональні вимоги.

Також було перераховано та описано засоби та технології розробки, описано класи та методи.

Наведено вимоги до апаратного та технічного забезпечення.

5 РЕЗУЛЬТАТИ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

5.1 Побудова експерименту

Random forest є потужним інструментом для вирішення проблем класифікації, але, як і у випадку з багатьма алгоритмами машинного навчання, може знадобитися трохи зусиль, щоб зрозуміти, що саме передбачається і що це означає в контексті. Scikit-Learn дозволяє досить легко запускати випадковий ліс та інтерпретувати результати. Розглянемо процес навчання простої моделі випадкового лісу та оцінимо її продуктивність за допомогою матриць плутанини та класифікаційних звітів.

Набір даних для експерименту візьмемо з відкритих джерел. Це буде набір даних створений Дж. А. Блекардом у 1998 році, він містить понад півмільйона спостережень із 54 функціями. Кожне спостереження представляє ділянку землі розміром 30 на 30 метрів у дикій місцевості штату Колорадо. Об’єкти записують картографічні дані про кожну територію: висоту, аспект, відстань до води, доріг, точок загоряння лісових пожеж, кількість тіні в різний час доби та який із 40 типів ґрунту він містить.

Для проведення експерименту нам знадобляться пакети Numpy і Pandas, для маніпуляцій з даними. Також імпортуємо Matplotlib і Seaborn для побудови візуалізації, яка буде створена пізніше.

Нам також потрібні деякі функції з пакету Scikit-Learn. З sklearn.model_selection нам потрібен train-test-split, щоб підганяти та оцінювати модель на окремих фрагментах набору даних. Також, нам потрібен RandomForestClassifier, а з пакету sklearn.metrics нам потрібні функції accuracy_score, confusion_matrix і classification_report.

```
# Імпортуємо необхідні пакети
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

Рисунок 5.1 – Імпорт пакетів для експерименту

Далі імпортуємо дані для експерименту. Розмір датасету становить близько 75 МБ.

```
# Імпортуємо датасет
df = pd.read_table("covtype.data", sep=',', header=None)
```

Рисунок 5.2 – Імпорт датасету

DataFrame *df* містить 54 стовпці, що представляють функції, і один стовпець, що містить ціль (мітки) під назвою Cover_Type. Розділимо дані на функції та цілі.

```
# Розділити набір даних на функції та цілі
y = df['Cover_Type']
X = df.drop('Cover_Type', axis=1)
```

Рисунок 5.3 – Розділення даних

Перевіримо кількість записів у кожному класі.

```
2    283301
1    211840
3     35754
7     20510
6     17367
5      9493
4      2747
Name: Cover_Type, dtype: int64
```

Рисунок 5.4 – Розподіл даних по класам

Бачимо, що класи досить незбалансовані. Об’єм найменшого класу складає приблизно 1% від об’єму найбільшого. Моделювання даних із таким дисбалансом класів є дещо ризикованим, оскільки моделі не бачать загальної картини. Вони хочуть знайти спосіб максимізувати будь-який показник оцінки, який ми використовуємо, і для цього вони можуть знайти швидкі шляхи. Наприклад, якби у нас було два класи, один з яких мав 99 прикладів, а інший лише 1, модель завжди могла передбачити перший клас, і це було б правильно в 99% випадк. Модель матиме високі оцінки за точністю, але насправді це не допоможе визначити приклади меншого класу.

Оскільки це швидка і брудна модель для огляду деяких інструментів оцінювання, ми не будемо турбуватися про балансування класів.

На наступному етапі, нам необхідно провести розподіл даних та тренування моделі.

```
# Розділити дані на набори та здіснити тренування
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=1, stratify=y)
```

Рисунок 5.5 – Розділення даних та тренування моделі

У наведеному вище коді необхідно звернути увагу на дві важливі речі. По-перше, встановлено random_state=1; це гарантує, що якщо доведеться повторно запустити код, то ми отримаємо точно такий же поділ даних, тому результати не зміняться.

Друге, на що потрібно звернути увагу, це stratify=y. Це змушує функцію train_test_split переконатися, що навчальні та тестові набори даних містять приклади кожного класу в тих самих пропорціях, що й у вихідному наборі даних. Це особливо важливо робити через незбалансованість даних. Випадкове розбиття може легко закінчитися тим, що всі приклади найменшого класу опиняться в тестовій множині й жодного в навчальному наборі, і тоді модель не може ідентифікувати цей клас.

Отже, створимо екземляр класу RandomForestClassifier.

```
RandomForestClassifier(bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto', max_leaf_nodes=None,
            min_impurity_decrease=0.0, min_impurity_split=None,
            min_samples_leaf=1, min_samples_split=2,
            min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=None,
            oob_score=False, random_state=None, verbose=0,
            warm_start=False)
```

Рисунок 5.6 – Створений екземпляр класу

Тепер ми можемо отримати прогнозовані мітки для тестових даних:

```
# Зробити прогнози для набору тестів
y_pred_test = forest.predict(X_test)
```

Рисунок 5.7 – Виклик функції прогнозування

Першою оцінкою продуктивності моделі є оцінка її точності. Цей показник вимірює, скільки міток отримала модель із загальної кількості передбачень. Можна розглядати це як відсоток правильних прогнозів. Це надзвичайно легко обчислити за допомогою Scikit-Learn, використовуючи

справжні мітки з тестового набору та прогнозовані мітки для тестового набору.

```
1  # Переглянути оцінку точності
2  accuracy_score(y_test, y_pred_test)
```

Рисунок 5.8 – Виклик функції, яка повертає оцінку точності моделі

Ця модель має точність 94% за даними тестування. Це виглядає досить вражаюче, але потрібно пам’ятати, що точність не є гарним показником продуктивності класифікатора, коли класи незбалансовані. Нам потрібно більше інформації, щоб зрозуміти, наскільки добре працює модель. Чи є вона однаково точною для кожного класу? Чи були якісь пари класів, які було особливо важко розрізнити? Цю інформацію дізнаємось за допомогою матриці плутанини.

5.2 Матриця плутанини

Матриця плутанини – це спосіб виразити, скільки прогнозів класифікатора були правильними, а якщо були неправильними, де класифікатор заплутався (звідси і назва). У наведених нижче матрицях плутанини рядки представляють справжні мітки, а стовпці – передбачені мітки. Значення по діагоналі представляють кількість (або відсоток, у нормалізованій матриці плутанини) випадків, коли передбачена мітка збігається з істинною міткою. Значення в інших клітинках представляють випадки, коли класифікатор неправильно позначив спостереження; стовпець повідомляє нам, що передбачив класифікатор, а рядок говорить нам, якою була правильна мітка. Це зручний спосіб визначити місця, де моделі може знадобитися додаткове навчання.

Scikit-Learn confusion_matrix() приймає справжні мітки та передбачення та повертає матрицю плутанини у вигляді масиву.

```
array([[50020,  2803,     1,     0,    13,    12,   111],
       [ 2932, 67515,   163,     1,    98,    91,    25],
       [    2,   210,  8508,    34,     5,   179,     0],
       [    0,     0,    91,   583,     0,    13,     0],
       [   54,   591,    31,     0,  1690,     7,     0],
       [   11,   200,   439,    22,     5,  3665,     0],
       [  324,    32,     0,     0,     1,     0,  4771]])
```

Рисунок 5.9 – Матриця плутанини

Ми бачимо, що в недіагональних клітинках у першому та другому стовпцях є кілька великих значень, а це означає, що класифікатор передбачав класи 1 та 2 багато разів, коли цього не мав робити. Це не дивно; це два найбільші класи, і класифікатор може зробити багато своїх передбачень правильними, просто вгадуючи один із цих двох класів знову і знову.

Цю матрицю плутанини було б набагато легше прочитати, якби вона мала деякі мітки та навіть кольорову шкалу, щоб допомогти нам визначити найбільші та найменші значення. Для цього використаємо Seaborn heatmap(). Також нормалізуємо значення в матриці плутанини, оскільки відсотки легше зрозуміти, ніж абсолютні підрахунки, коли класи мають такий різний розмір.

```
# Отримати та змінити форму даних
matrix = confusion_matrix(y_test, y_pred_test)
matrix = matrix.astype('float') / matrix.sum(axis=1)[:, np.newaxis]

# Побудувати графік
plt.figure(figsize=(16,7))
sns.set(font_scale=1.4)
sns.heatmap(matrix, annot=True, annot_kws={'size':10},
            cmap=plt.cm.Greens, linewidths=0.2)

# Додати мітки до поділу
class_names = ['Spruce/Fir', 'Lodgepole Pine', 'Ponderosa Pine',
               'Cottonwood/Willow', 'Aspen', 'Douglas-fir',
               'Krummholz']
tick_marks = np.arange(len(class_names))
tick_marks2 = tick_marks + 0.5
plt.xticks(tick_marks, class_names, rotation=25)
plt.yticks(tick_marks2, class_names, rotation=0)
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion Matrix for Random Forest Model')
plt.show()
```

Рисунок 5.10 – Нормалізація та візуалізація даних

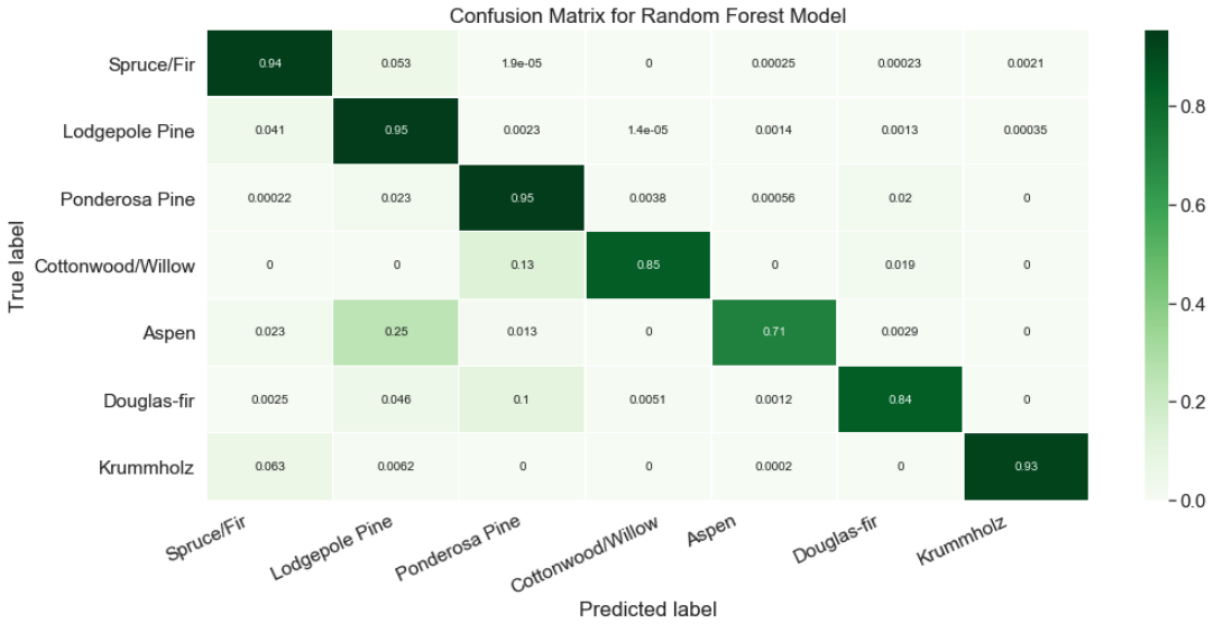


Рисунок 5.11 – Графічне представлення матриці плутанини

Тепер легко побачити, що нашому класифікатору було важко передбачити мітку Aspen. Приблизно у четверті випадків мітка Aspens була помилково класифікована як Lodgepole Pines.

5.3 Звіт про класифікацію

Щоб отримати ще більше розуміння продуктивності моделі, ми повинні вивчити інші показники, такі як точність, запам’ятовування та оцінка F1.

Точність – це кількість правильно ідентифікованих членів класу, поділена на всі випадки, коли модель передбачила цей клас. У випадку Aspens оцінка точності дорівнює кількості правильно ідентифікованих Aspens, поділеній на загальну кількість разів, коли класифікатор передбачив «Aspen», правильно чи неправильно.

Відкликання – це кількість членів класу, яку класифікатор правильно визначив, поділена на загальну кількість членів у цьому класі. Для Aspens це буде кількість фактичних Aspens, які класифікатор правильно визначив як такі.

Оцінка F1 трохи менш інтуїтивно зрозуміла, оскільки вона поєднує в собі точність і запам’ятовування в один показник. Якщо і точність, і запам’ятовування високі, F1 також буде високим. Якщо вони обидва низькі,

F1 буде низьким. Якщо один високий, а інший низький, F1 буде низьким. F1 – це швидкий спосіб визначити, чи справді класифікатор добре ідентифікує членів класу, чи він знаходить ярлики (наприклад, просто ідентифікує все як члена великого класу).

Використаємо classification_report() з пакету Scikit-Learn, щоб переглянути ці показники для нашої моделі.

	precision	recall	f1-score	support
1	0.94	0.94	0.94	52960
2	0.95	0.95	0.95	70825
3	0.92	0.95	0.94	8938
4	0.91	0.85	0.88	687
5	0.93	0.71	0.81	2373
6	0.92	0.84	0.88	4342
7	0.97	0.93	0.95	5128
micro avg	0.94	0.94	0.94	145253
macro avg	0.94	0.88	0.91	145253
weighted avg	0.94	0.94	0.94	145253

Рисунок 5.10 – Звіт про класифікацію

Проаналізуємо метрику для 5-го класу (Aspen). Точність висока, це означає, що модель була обережна, щоб уникнути маркування як «Aspen» речей, які не є Aspen. З іншого боку, запам’ятовування є відносно низьким, що означає, що класифікатор пропускає купу Aspens, оскільки він надто обережний. Оцінка F1 відображає цей дисбаланс.

Висновок до розділу

Під час проведення даного експерименту було розглянуто способи оцінки якості моделі Random forest.

Класифікаційний звіт повідомляє нам, які помилки припустила модель, але не дає нам конкретних даних.

Матриця плутанини повідомляє, де саме були зроблені помилки, але вона не дає нам підсумкових показників, таких як точність, запам’ятовування або результат F1. Використання обох цих параметрів може дати набагато

більш детальне розуміння того, як працює модель, виходячи далеко за межі того, що може сказати нам оцінка точності, і уникаючи деяких її пасток.

6 РОЗРОБКА СТАРТАП-ПРОЕКТУ

6.1 Опис ідеї проекту

6.1.1 Зміст ідеї

Класифікатор email-повідомлень призначений для оптимізації процесу обробки звернень клієнтів до корпоративних контакт-центрів, служб технічної підтримки, консьєрж-сервісів та інших подібних організацій, діяльність яких пов’язана з наданням послуг клієнтам та споживачам.

Розроблений програмний продукт також може бути впроваджений на комунальних та державних підприємствах, у яких є необхідність в обробці та відповіді багатьох однотипних повідомлень, наприклад скарг на відсутність електропостачання, інтернету або передачі показів лічильників. При отриманні компанією відповідного повідомлення, за допомогою класифікації, воно може бути одразу спрямоване до відповідального спеціаліста, що потенційно пришвидшить швидкість реакції на цей запит.

Тож, основним напрямком застосування програмного продукту, що був розроблений, є оптимізація роботи контакт-центрів та центрів технічної підтримки.

Потенційні вигоди для користувачів:

– зменшення часу реакції на повідомлення;

– здешевлення вартості обробки запиту;

– підвищення ефективності роботи всієї структури.

У таблиці 6.1 наведено концепцію ідеї стартап-проекту, можливі сфери його застосування та потенційні вигоди для установ, що впровадили у себе даний програмний продукт.

Таблиця 6.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Класифікатор email-повідомлень призначений для оптимізації процесу обробки звернень клієнтів до корпоративних контакт-центрів, служб технічної підтримки, консьєрж-сервісів та інших подібних організацій	1. Обробка повідомлень зі скаргами у службах технічної підтримки	1. Прискорення реакції на однотипні скарги за рахунок маршрутизації їх у відповідний відділ або на відповідального працівника. 2. Можливість формування автоматичної відповіді в залежності від виду скарги.
	2. Обробка запитів до консьєрж-сервісу	1. Збільшення швидкості відповіді на запит за рахунок направлення запиту працівникові, який є кваліфікований у даному питанні. 2. Формування автоматичних відповідей на типові запити.

Продовження таблиці 6.1

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Класифікатор email-повідомлень призначений для оптимізації процесу обробки звернень клієнтів до корпоративних контакт-центрів, служб технічної підтримки, консьєрж-сервісів та інших подібних організацій	3. Автоматична обробка показів лічильників провайдерами послуг(наприклад електро- або водопостачання)	1. Зменшення кількості необхідних людських ресурсів, потрібних для оброблення великих масивів повідомлень у ручному режимі та вивільнення ресурсів для інших задач. 2. Автоматичне інформування користувачів про помилки у наданій інформації.

Отже, в загальному, основну ідею стартап-проекту можна “застосунок для оптимізації та автоматизації обробки великих масивів однотипних email-повідомлень”.

6.1.2 Техніко-економічний аналіз проекту

Розроблена система класифікації повідомлень для контакт-центрів має декілька конкурентів:

– Mailytica;

– emailtree.ai;

– 360 Protective Marking;

– Klassify.

Розглянемо переваги та недоліки конкурентів у порівнянні з нашим продуктом у таблиці 5.2.

Таблиця 6.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко-економічні хар-ки стартап-ідеї	Конкуруючі програмні продукти					W (слабка сторона)га	N (нейтральна сторона)	S (сильна сторона)
		Мій продукт	Mailytica	emailtree.ai	360 Protective Marking	Klassify			
1	Ступінь автоматизації	Всі процеси автоматизовані, втручання адміністратора не потребується	Всі процеси автоматизовані, втручання адміністратора не потребується	Всі процеси автоматизовані, втручання адміністратора не потребується	Потрібна участь адміністратора	Всі процеси автоматизовані, втручання адміністратора не потребується			+

Продовження таблиці 6.2

№ п/п	Техніко-економічні хар-ки стартап-ідеї	Конкуруючі програмні продукти					W (слабка сторона)га	N (нейтральна сторона)	S (сильна сторона)
		Мій продукт	Mailytica	emailtree.ai	360 Protective Marking	Klassify			
2	Можливість модифікації в залежності від галузі застосування	Передбачена можливість простої модифікації під специфіку галузі	Не передбачено	Не передбачено	Частково передбачено	Не передбачено			+

Продовження таблиці 6.2

№ п/п	Техніко-економічні хар-ки стартап-ідеї	Конкуруючі програмні продукти					W (слабка сторона) га	N (нейтральна сторона)	S (сильна сторона)
		Мій продукт	Mailytica	emailtree.ai	360 Protective Marking	Klassify			
3	Можливість використання декількох джерел даних	Передбачено можливість роботи з різними поштовими сервісами за рахунок відсутності прив’язко до жодного з них	Не передбачено	Не передбачено	Не передбачено	Не передбачено			+

Продовження таблиці 6.2

№ п/п	Техніко-економічні хар-ки стартап-ідеї	Конкуруючі програмні продукти					W (слабка сторона)га	N (нейтральна сторона)	S (сильна сторона)
		Мій продукт	Mailytica	emailtree.ai	360 Protective Marking	Klassify			
4	Актуальність технологій розробки	Новітні	Сучасні	Сучасні	Застарілі	Сучасні			+
5	Використання інтелектуальних алгоритмів	Використання високоефективного і точного алгоритму машинного навчання Random forest	Не регламентовано	Так	Ні	Так			+

Продовження таблиці 6.2

№ п/п	Техніко-економічні хар-ки стартап-ідеї	Конкуруючі програмні продукти					W (слабка сторона)га	N (нейтральна сторона)	S (сильна сторона)
		Мій продукт	Mailytica	emailtree.ai	360 Protective Marking	Klassify			
6	Вартість впровадження та підтримки	Низька	Середня	Середня	Висока	Середня			+

Отже, як можна бачити з наведеної вище таблиці, розроблена нами класифікаційна система має значні переваги порівняно з усіма існуючими конкурентами.

6.2 Технологічний аудит ідеї проекту

Для реалізації програмного продукту було використано певний стек технологій. У таблиці 6.3 наведено технології, які було використано для розробки, а також, проаналізовано їх актуальність та доступність.

Таблиця 6.3 – Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Класифікатор email-повідомлень призначений для оптимізації процесу обробки звернень клієнтів до корпоративних контакт-центрів, служб технічної підтримки, консьєрж-сервісів та інших подібних організацій	Python – мова програмування високого рівня загального призначення	Так	Доступна безкоштовно
		Scikit-learn – безкоштовна бібліотека машинного навчання для мови програмування Python	Так	Доступна безкоштовно
		Flask – мікровеб-фреймворк, написаний мовою Python	Так	Доступна безкоштовно
		Node.js – серверне середовище з відкритим кодом.	Так	Доступна безкоштовно
		NestJS – це платформа для створення ефективних, масштабованих веб-додатків Node.js	Так	Доступна безкоштовно

Продовження таблиці 6.3

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Класифікатор email-повідомлень призначений для оптимізації процесу обробки звернень клієнтів до корпоративних контакт-центрів, служб технічної підтримки, консьєрж-сервісів та інших подібних організацій	MongoDB – кросплатформна документоорієнтована програма бази даних із доступним вихідним кодом	Так	Доступна безкоштовно
		Microsoft Azure – платформа хмарних обчислень, якою керує Microsoft для керування додатками через центри обробки даних, якими керує Microsoft.	Так	Доступна, стягується плата за користування

Отже, ідея стартап-проекту може бути реалізована за допомогою ряду технологій, перерахованих у таблиці 6.3, адже вони є доступними та можуть бути вільно використані.

6.3 Аналіз ринкових можливостей запуску стартап-проекту

У наступній таблиці(6.4) наведено попередній аналіз попиту продукту, обсяг, а також динаміка розвитку галузі.

Таблиця 6.4 – Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку (найменування)	Показник
1	Кількість головних гравців, од	4
2	Загальний обсяг продаж, грн/ум.од	4000 грн/ум.од
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу (вказати характер обмежень)	Відсутні
5	Специфічні вимоги до стандартизації та сертифікації	Дотримання вимог статті 12 Закону України “Про захист персональних даних”
6	Середня норма рентабельності в галузі (або по ринку), %	20

Середня норма ринкової рентабельності по галузі є досить високою, а саме, 20% на рік, що є набагато вигідніше за банківські вклади станом на кінець 2022 року, отже стартап є досить привабливим з точки зору цікавості інвесторів.

Із попередньої оцінки ринку можна зробити висновок, що наразі ринок є сприятливим для старту проекту.

Далі, у таблиці 6.5 проаналізуємо потенційних клієнтів продукту та їх характеристики.

Таблиця 6.5 – Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Програмне забезпечення для спрощення та оптимізації процесу обробки вхідних email-повідомлень	Працівники контакт-центрів	Спеціалісти контакт-центрів зазвичай вручну обробляють великі обсяги вхідних заявок	Швидкість роботи, точність класифікації
2	Програмне забезпечення для спрощення та оптимізації процесу обробки вхідних запитів	Працівники консьєрж-сервісів	Ручна обробка великих кількостей запитів	Збільшення швидкості реакції на запити, автоматична маршрутизація на відповідального працівника

Продовження таблиці 6.5

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
3	Програмне забезпечення для спрощення та оптимізації процесу обробки заявок	Працівники служб технічної підтримки	Обробка та реакція на велику кількість заявок та скарг	Полегшення процесу обробки заявок, формування автоматичних відповідей на типові заяви та скарги

У таблиці 6.6 проведемо аналіз та покажемо стан ринку, розглянемо фактори загроз, що можуть бути перешкодою при запуску стартап-проекту.

Таблиця 6.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Зростаюча конкуренція	Вихід на ринок подібних продуктів з переважаючими характеристиками та функціоналом	Пропрацювати можливості покращення продукту для отримання додаткових переваг над конкурентними продуктами

Продовження таблиці 6.6

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
2	Зміна запитів цільової аудиторії	Цільовим користувачам потрібен інший функціонал програмного забезпечення	Потрібно передбачити можливість зміни та розширення функціоналу в залежності від потреб цільових користувачів, тобто продукт має бути адаптивним
3	Відсутність зацікавленості цільових користувачів	Потенційні клієнти не зацікавлені в покупці продукту	Розвиток маркетингової програми проекту
4	Нестабільність ринку України	Нестабільна економічна ситуація, реформи індустрії	Вихід проекту на іноземні ринки

У таблиці 6.7 наведено позитивні фактори ринку, які сприятимуть впровадженню нового продукту на ринку.

Таблиця 6.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Інвестування	Знаходження інвесторів для проекту	Аналіз ринку та інвестиційної привабливості продукту, його покращення
2	Розвиток продукту	Збільшення привабливості продукту для цільовою аудиторії за рахунок його покращення	Додавання нового функціоналу, покращення точності класифікації та алгоритмів, поширення на суміжні галузі
3	Конкуренто-спроможність	Продукт є унікальним на ринку	Адаптація продукту до реальних потреб ринку конкретної галузі

Далі, у таблиці 6.8 проведемо аналіз пропозиції, а саме ступеневий аналіз ринкової конкуренції.

Таблиця 6.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Тип конкуренції: олігополія	Наразі на ринку присутні декілька гравців зі схожим продуктом, але їх продукти мають менший функціонал та програють у ціновому порівнянні	Постійне покращення продукту, розробка нового функціоналу, адаптація до потреб клієнтів
2. За рівнем конкурентної боротьби: інтернаціональний рівень	Конкуренти присутні в багатьох країнах світу	Адаптація програмного продукту для використання в різних країнах та галузях, переклад інтерфейсу на різні мови
3. За галузевою ознакою: внутрішньогалузева	Програмний продукт призначений для галузі послуг та інформаційного обслуговування користувачів	Додавання нового функціоналу, перманентний розвиток продукту
4. Конкуренція за видами товарів: товарно-видова	Конкуренція між різними видами програмних продуктів та компаніями, які їх розробили	Створення продукту з урахуванням недоліків конкурентів

Продовження таблиці 6.8

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
5. За характером конкурентних переваг: нецінова	Зниження собівартості розробки та життєвого циклу програмного продукту	Пошук способів здешевлення собівартосту розробки і підтримку продукту. Наприклад, використання дешевших або безкоштовних технологій, оптимізація витрат на інфраструктуру
6. За інтенсивністю: не марочна	Продукт не є брендом	Необхідно створити бренд

У таблиці 6.9 більш детально проаналізуємо умови конкуренції в галузі за М. Портером.

Таблиця 6.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	– Mailytica – emailtree.ai – 360 Protective Marking – Klassify	Наявність компаній, які розвивають схожі за функціоналом програмні продукти	Не релевантно до даної галузі	Потреба клієнтів у автоматизації процесів	Продукт є більш відомим та впізнаванім на ринку, має ширший функціонал та можливості, має кращу якість

Продовження таблиці 6.9

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Висновки	Інтенсивність конкурентної боротьби на ринку є середньою	Час необхідний для виходу на ринок - 5 місяців. Ринок достатньо відкритий для входу, але наявні конкуренти	Не релевантно до даної галузі	Потреби клієнтів є вагомим фактором при плануванні розвитку продукту та введенні нових функціональних можливостей в продукт	Необхідно завжди слідкувати за продуктами конкурентів, щоб бути в курсі їх нововведень

Отже, можемо зробити висновок, що ринок є достатньо відкритих для нових гравців, конкуренція на середньому рівні. Перелічені фактори позитивно впливатимуть на рішення про вихід продукту на ринок.

У таблиці 6.10 проаналізовано фактори конкурентоспроможності з огляду на аналіз конкуренції.

Таблиця 6.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Вартість	Вартість використання продукту для клієнтів є набагато нижчою у порівнянні з продуктами-конкурентами
2	Якість та швидкість класифікації повідомлень	Швидкість та якість робот продукту є вищою ніж у існуючих продуктів
3	Орієнтованість на цільову аудиторію	Взаємодія розробників продукту з кінцевими споживачами для виявлення їх потреб
4	Розвиток галузі	Орієнтація продукту на галузь, яка стрімко розвивається та постійно потребує нових рішень

У таблиці 6.11 проаналізуємо слабкі та сильні сторони стартап-проекту за факторами конкурентоспроможності, що були визначені у таблиці 6.10.

Таблиця 6.11 – Порівняльний аналіз сильних і слабких сторін рекомендаційної системи

№ п/п	Фактор конкуренто-спроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з даним проектом						
			-3	-2	-1	0	1	2	3
1	Вартість			*					
2	Якість та швидкість класифікації повідомлень					*			
3	Орієнтованість на цільову аудиторію						*		
4	Розвиток галузі							*	

У таблиці 6.12 проведемо аналіз слабких і сильних сторін продукту, можливостей та загроз з погляду на визначені ринкові загроз та можливості.

Таблиця 6.12 – SWOT-аналіз стартап-проекту

Сильні сторони: – Вартість експлуатації – Рівень автоматизації роботи – Адаптивність	**Слабкі сторони:** – Вузький функціонал
Можливості: – Вдосконалення алгоритмів класифікації та самого продукту	**Загрози:** – Конкуренція зі сторони інших гравців – Нестабільність галузі

У таблиці 6.13 розглянемо альтернативні способи виведення проекту на ринок.

Таблиця 6.13 – Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Розповсюдження програмного продукту на комерційній основі	85%	6 місяців
2	Таргетинг	60%	2 місяці
3	Залучення інвестицій	90%	1 рік
4	Безкоштовний пробний період	98%	1 місяць

6.4 Розроблення ринкової стратегії проекту

Для того, щоб розробити ринкову стратегію продукту, необхідно визначити стратегію охоплення ринку.

У таблиці 6.14 проведено опис груп цільових споживачів.

Таблиця 6.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Великі технологічні компанії, які мають у своєму складі підрозділи, які займаються контактами з клієнтами	Не висока	40%	Помірна	Середня
2	Середні технологічні ком-панії, які мають у своєму складі підрозділи, які займаються контактами з клієнтами	Висока	75%	Помірна	Висока

Продовження таблиці 6.14

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
3	Малі технологічні компанії, які мають у своєму складі підрозділи, які займаються контактами з клієнтами	Середня	60%	Низька	Висока
4	Банківські установи	Висока	80%	Висока	Висока
5	Державні установи	Низька	15%	Низька	Низька
6	Консьєрж-сервіси	Висока	65%	Середня	Помірна
Які цільові групи обрано: Середні технологічні компанії та банківські установи					

Після того, як було обрано цільові групи користувачів та потенційних клієнтів, наступним кроком є формульування базової стратегій розвитку.

У таблиці 6.15 описана базова стратегія розвитку.

Таблиця 6.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Альтернативний варіант, котрий передбачає розробку програмного забезпечення для автоматизованої класифікації вхідних повідомлень	Визначити потреби цільових користувачів, розробити стратегію розвитку відповідно до потреб цільової аудиторії	Вартість експлуатації, якість класифікації, швидкість роботи, відсоток працівників, яких можна вивільнити від здійснення роботи вручну	Стратегія диференціації

Далі потрібно обрати стратегії конкурентної поведінки. Зробимо це у таблиці 6.16.

Таблиця 6.16 –Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Так	Компанія буде залучати нових споживачів	Не буде	Стратегія диференціації

Проаналізувавши дані про вимоги споживачів до продукту та компанії, наведені в таблиці 6.5, враховуючи обрану стратегію розвитку (таблиця 6.15) та стратегію конкурентної поведінки (наведено у таблиці 6.16), у таблиці 6.17 розробити стратегію позиціонування.

Таблиця 6.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Точність класифікації, висока швидкість роботи, низька вартість експлуатації	Стратегія диверсифікації	Проект має відповідати ідеям та цілям компанії, бути безпечним та захищеним, бути конкурентоспроможним, мати низький поріг входу	Економія людських ресурсів, зручність використання, швидкість роботи, точність роботи

6.5 Розроблення маркетингової програми стартап-проекту

Для розроблення маркетингової програми проекту, визначимо ключові переваги концепції продукту у таблиці 6.18.

Таблиця 6.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або ті, що потрібно створити)
1	Висока швидкість класифікації	Алгоритм машинного навчання Random forest працює надзвичайно швидко	Відсутність помилок при класифікації
2	Робота з декількома джерелами даних	Програмний продукт може інтегруватися з багатьма платформами	Можливість інтеграції у сервіси, якими уже користується клієнт
3	Точність класифікації	Користувач отримує бажаний точний результат класифікації	Підвищена точність алгоритмів класифікації
4	Автоматизація формування відповідей на типові повідомлення	Працівники не втручаються у процес формування відповідей	Система автоматично формує відповіді на повідомлення

Далі необхідно розробити та описати трирівневу маркетингову модель продукту. Уточнимо ідею, складові та способи надання розробленого програмного продукту у таблиці 6.19.

Таблиця 6.19 – Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Швидкість та точність класифікації повідомлень</td></tr>
<tr><td rowspan="7">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Точність класифікації</td><td>М</td><td>Тл</td></tr>
<tr><td>2. Швидкість роботи</td><td>М</td><td>Тл</td></tr>
<tr><td>3. Безпека даних</td><td>М</td><td>Е</td></tr>
<tr><td>4. Адаптивність продукту</td><td>М</td><td>Ор</td></tr>
<tr><td colspan="3">Якість: засвідчена тестуванням</td></tr>
<tr><td colspan="3">Пакування: відсутнє</td></tr>
<tr><td colspan="3">Марка: відсутня</td></tr>
<tr><td rowspan="2">III. Товар із підкріпленням</td><td colspan="3">До продажу: гарантія якості</td></tr>
<tr><td colspan="3">Після продажу: технічна підтримка користувачів</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: ліцензування</td></tr>
</table>

Далі необхідно визначити межі цін з урахуванням рівня доходів потенційних користувачів. Дані про межі цін наведено у таблиці 6.20.

Таблиця 6.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
	150$	150$	Середній	20-50

У таблиці 6.21 розглянемо систему збуту продукту.

Таблиця 6.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Цільовими клієнтами продукту є технологічні компанії, які прагнуть оптимізувати та спростити процес обробки вхідних листів та заявок у своїх контакт-центрах	Контактування з потенційними клієнтами, дослідження ринку, удосконалення програмного продукту з урахуванням відгуків споживачів, розміщення реклами та розвиток маркетингових стратегій	Одноріневий (виробник-споживач)	Прямий збут споживачам

Останнім, але не менш важливим компонентом маркетингової стратегії є розробка програми маркетингової комунікації, яка базується на обраній раніше основі для позиціонування, а також специфіку поведінки клієнтів. Наведемо її в таблиці 6.22.

Таблиця 6.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Цільові клієнти - технологічні компанії, які прагнуть оптимізувати та спростити процес обробки вхідних листів та заявок у своїх контакт-центрах	Інтернет спілкування, живе спілкування з іншими компаніями, рекламні кампанії	Порівняння продукту з конкурентами, демонстрація його переваг, наголошення на його інноваційності та актуальності	Залучення нових клієнтів, збільшення обсягів продажу	Наголошення на можливостях оптимізації виробничих процесів, які отримує клієнт придбавши програмний продукт

Висновки до розділу

У даному розділі було розроблено стартап-проект для програмного продукту. Було сформовано ідею, яка полягає у створення інтелектуальної системи для класифікації повідомлень, що дозволяє компаніям значно оптимізувати свої процеси комунікацій з клієнтами, дозволяє зменшити кількість працівників, необхідну для нормальної роботи компанії, оптимізувати людські ресурси, за рахунок чого зменшуються видатки на утримання департаментів комунікації.

У цьому розділі також було проведено технічний та економічний аналіз ідеї, проаналізований ринок та розглянуто можливості для запуску даного стартап-проекту.

Також було розроблено маркетингову концепцію та ринкову стратегію розвитку проекту в майбутньому.

ВИСНОВКИ

За період роботи над даною магістерською дисертацією було виконано аналіз сучасних досліджень у сфері класифікації текстів методами машинного навчання. Насамперед, було проведено огляд сучасних алгоритмів класифікації, які використовуються у машинному навчанні, відомих методів роботи з даними та методів інтерпретації роботи алгоритмів. Базуючись на проведеному огляді методів машинного навчання, було обрано оптимальний алгоритм для розв'язання поставленої задачі, а саме, алгоритм Random forest (випадкові дерева). Виявлено, що обраний алгоритм краще за інші справляється з поставленим завданням.

На основі проведених досліджень було розроблено систему класифікації повідомлень, яка дозволяє оптимізувати роботу підприємств та департаментів, робота яких пов'язана з комунікаціями з клієнтами та передбачає ручну обробку великих масивів повідомлень, заявок та іншого. Застосування алгоритму машинного навчання дало можливість зробити систему гнучкою, покращити точність класифікації, що є беззаперечною перевагою розробленої системи над існуючими аналогами, а також, є фактором, який привертає увагу потенційних користувачів розробленого програмного продукту.

Протягом тестування розробленої системи, було перевірено правильність та точність її роботи, а також, відповідність очікуванням користувачів.

Було розроблено стартап-проект, в якому проаналізовано актуальність продукту на сучасному ринку, розроблено маркетингову концепцію та ринкову стратегію розвитку проекту, описано варіанти дистрибуції та варіанти виходу програмного продукту на ринок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Batura, Tatiana. (2017). Методи автоматичної класифікації текстів. Міжнародний журнал Програмні продукти і системи. 23. 85-99. 10.15827/0236-235X.117.085-099
2) Розроблення стартап-проєкту [Електронний ресурс] // Методичні рекомендації до виконання розділу магістерських дисертацій для студентів інженерних спеціальностей / За заг. ред. О.А. Гавриша. – Київ : НТУУ «КПІ», 2016. – 28 с.
3) Random Forests [Електронний ресурс] // Режим доступу: https://link.springer.com/article/10.1023/A:1010933404324
4) Breiman, Leo. (2001). RANDOM FORESTS [Електронний ресурс] // Режим доступу: https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf
5) Matthias Schonlau and Rosie Yuyan Zou. The random forest algorithm for statistical learning [Електронний ресурс] // Режим доступу: https://journals.sagepub.com/doi/full/10.1177/1536867X20909688
6) Alessia Sarica, Antonio Cerasa, Aldo Quattrone. Random Forest Algorithm for the Classification of Neuroimaging Data in Alzheimer's Disease: A Systematic Review [Електронний ресурс] // Режим доступу: https://www.frontiersin.org/articles/10.3389/fnagi.2017.00329/full
7) Vladimir Svetnik, Andy Liaw, Christopher Tong, J. Christopher Culberson, Robert P. Sheridan, and Bradley P. Feuston. Random Forest: A Classification and Regression Tool for Compound Classification and QSAR Modeling [Електронний ресурс] // Режим доступу: https://pubs.acs.org/doi/10.1021/ci034160g
8) Raphael Couronné, Philipp Probst, Anne-Laure Boulesteix. Random forest versus logistic regression: a large-scale benchmark experiment [Електронний ресурс] // Режим доступу: https://bmcbioinformatics.biomedcentral.com/articles/10.1186/s12859-018-2264-5

9) Jenny Yeon, Jared Wilber. The Random Forest Algorithm. How the majority vote and well-placed randomness can enhance the decision tree model. [Електронний ресурс] // Режим доступу: https://mlu-explain.github.io/random-forest/

10) Gerard Biau. (2012). Analysis of a Random Forests Model [Електронний ресурс] // Режим доступу: https://www.jmlr.org/papers/volume13/biau12a/biau12a.pdf

11) Song J , Gao Y, Yin P, Li Y, Zhang J, Su Q, Fu X, Pi H. (2021). The Random Forest Model Has the Best Accuracy Among the Four Pressure Ulcer Prediction Models Using Machine Learning Algorithms [Електронний ресурс] // Режим доступу: https://www.dovepress.com/the-random-forest-model-has-the-best-accuracy-among-the-four-pressure--peer-reviewed-fulltext-article-RMHP

12) Mark P. Little, Philip S. Rosenberg, Aryana Arsham. (2022). Alternative stopping rules to limit tree expansion for random forest models [Електронний ресурс] // Режим доступу: https://www.nature.com/articles/s41598-022-19281-7

13) Danielle Denisko and Michael M. Hoffman. (2018). Classification and interaction in random forests [Електронний ресурс] // Режим доступу: https://www.pnas.org/doi/10.1073/pnas.1800256115

14) Vrushali Y Kulkarni, PhD Student, COEP, Pune, India. (2013). Random Forest Classifiers :A Survey and Future Research Directions [Електронний ресурс] // Режим доступу: https://adiwijaya.staff.telkomuniversity.ac.id/files/2014/02/Random-Forest-Classifiers_A-Survey-and-Future.pdf

15) Breiman, L. (2001) Random Forests. Machine Learning, 45, 5-32.

16) Shi, Z. (2019) Cognitive Machine Learning. International Journal of Intelligence Science, 9, 111-121.

17) Grove, A. and Schuurmans, D. (1998). Boosting in the limit: Maximizing the margin of learned ensembles. In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98).

18) Domingos, P., The Master Algorithm: How the Quest for the Ultimate Learning Machine Will Remake Our World. Basic Books, 2015.

19) Hardt, Moritz, Price, Eric & Srebro, Nati. “Equality of opportunity in supervised learning.” Advances in neural information processing systems. 2016.

20) Henke, N., et. al., “The age of analytics: Competing in a data-driven world,” McKinsey Global Institute, December 2016. Accessed October 2020.

21) Lundberg, Scott M., & Lee, Su-in. “A Unified Approach to Interpreting Model Predictions.” Advances in Neural Information Processing Systems. 2017.

22) Altman RB, Bergman CM, Blake J, et al. Text mining for biology–the way forward: opinions from leading scientists. Genome Biology. 2008;9(Suppl 2):S7.

23) Aphinyanaphongs Y, Aliferis C. Text categorization models for identifying unproven cancer treatments on the web. Stud Health Technol Inform. 2007;129:968–72.

24) Comparative Effectiveness Review No. 32. Rockville, MD: Agency for Healthcare Research and Quality; 2011. (Prepared by Tufts Evidence-based Practice Center under Contract No. 290-2007-100551). AHRQ Publication No. 11-EHC052-EF [Електронний ресурс] // Режим доступу: www.effectivehealthcare.ahrq.gov/reports/final.cfm

25) Cohen AM, Ambert K, McDonagh M. Cross-topic learning for work prioritization in systematic review creation and update. J Am Med Inform Assoc. 2009;16(5):690–704.

26) Cohen AM, Ambert K, McDonagh M. A Prospective Evaluation of an Automated Classification System to Support Evidence-based Medicine and Systematic Review. AMIA Annual Symposium proceedings. 2010:121–5.

27) Facilitate Comparative Effectiveness Review Updating, Methods Research Report. Rockville, MD: Agency for Healthcare Research and Quality; Sep, 2012. Prepared by the Southern California Evidence-based Practice Center under Contract No. 290-2007-10062-I). AHRQ Publication No. 12-EHC069-EF.

28) Deerwester S, Dumais S, Furnas G, et al. Indexing by Latent Semantic Analysis. J Am Soc Inf Sci. 1990;41(6):391–407.

29) EPOC: Cochrane Effective Practice and Organisation of Care Group. Welcome to the EPOC Ottawa Website [Електронний ресурс] // Режим доступу: http://epoc.cochrane.org.

30) Friedman J, Hastie T, Tibshirani R. Regularization paths for generalized linear models via coordinate descent. J Stat Softw. 2010;33(1):1–22.

31) Genkin A, Lewis DD, Madigan D. Large-scale Bayesian logistic regression for text categorization. Techometric. 2007;49(3):201–304.

32) Hastie T, Tibshirani R, Friedman J. New York: Springer Verlag; The elements of statistical learning: data mining, inference, and prediction. 2009

33) O'Neil S, Rubenstein L, Danz M, et al. Identifying continuous quality improvement publications: what makes an improvement intervention 'CQI'? BMJ Qual Saf. 2011 Dec;20(12):1011–9. Epub 2011 Jul 4.

34) Vapnik VN. The Nature of Statistical Learning theory. New York: Springer-Verlag; 1995.

35) Wallace BC, Small K, Brodley C, et al. Active learning for biomedical citation screening. KDD '2010. Proceedings of the 16th ACM SIGKDD international conference on knowledge discovery and data mining; Washington, DC: ACM; 2010.

36) Ahmed, A., Aly, M., Gonzalez, J., Narayanamurthy, S., & Smola, A. J. (2012). Scalable inference in latent variable models. Proceedings of the fifth

ACM international conference on Web search and data mining (pp. 123–132).

37) Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., … others. (2022). Flamingo: a visual language model for few-shot learning. arXiv preprint arXiv:2204.14198.

38) Anil, R., Gupta, V., Koren, T., Regan, K., & Singer, Y. (2020). Scalable second order optimization for deep learning. arXiv preprint arXiv:2002.09018.

39) Baevski, A., & Auli, M. (2018). Adaptive input representations for neural language modeling. International Conference on Learning Representations.

40) Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

41) Beutel, A., Murray, K., Faloutsos, C., & Smola, A. J. (2014). Cobafi: collaborative bayesian filtering. Proceedings of the 23rd international conference on the World wide web (pp. 97–108).

42) Bojanowski, P., Grave, E., Joulin, A., & Mikolov, T. (2017). Enriching word vectors with subword information. Transactions of the Association for Computational Linguistics, 5, 135–146.

43) Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., … others. (2021). On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258.

44) Bottou, L., & Le Cun, Y. (1988). Sn: a simulator for connectionist models. Proceedings of NeuroNimes 88 (pp. 371–382). Nimes, France. [Електронний ресурс] // Режим доступу: http://leon.bottou.org/papers/bottou-lecun-88

45) Bowman, S. R., Angeli, G., Potts, C., & Manning, C. D. (2015). A large annotated corpus for learning natural language inference. arXiv preprint arXiv:1508.05326.

46) Bradley, R. A., & Terry, M. E. (1952). Rank analysis of incomplete block designs: i. the method of paired comparisons. Biometrika, 39(3/4), 324–345.

47) Brown, P. F., Cocke, J., Della Pietra, S. A., Della Pietra, V. J., Jelinek, F., Mercer, R. L., & Roossin, P. (1988). A statistical approach to language translation. Coling Budapest 1988 Volume 1: International Conference on Computational Linguistics.

48) Cer, D., Diab, M., Agirre, E., Lopez-Gazpio, I., & Specia, L. (2017). Semeval-2017 task 1: semantic textual similarity multilingual and crosslingual focused evaluation. Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017) (pp. 1–14).

ДОДАТОК А

Графічні матеріали

Загальна структурна схема розгортання застосунку

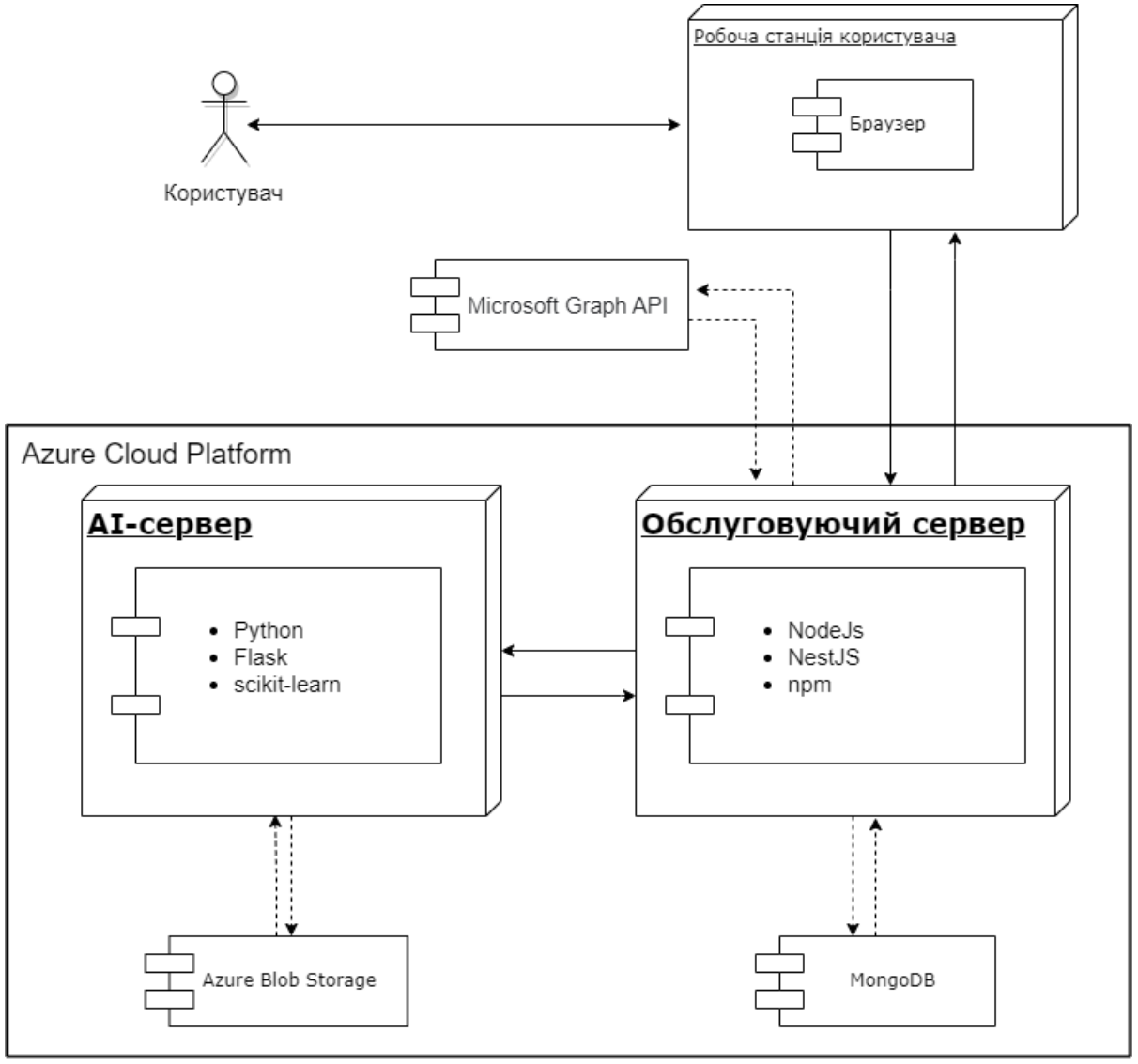
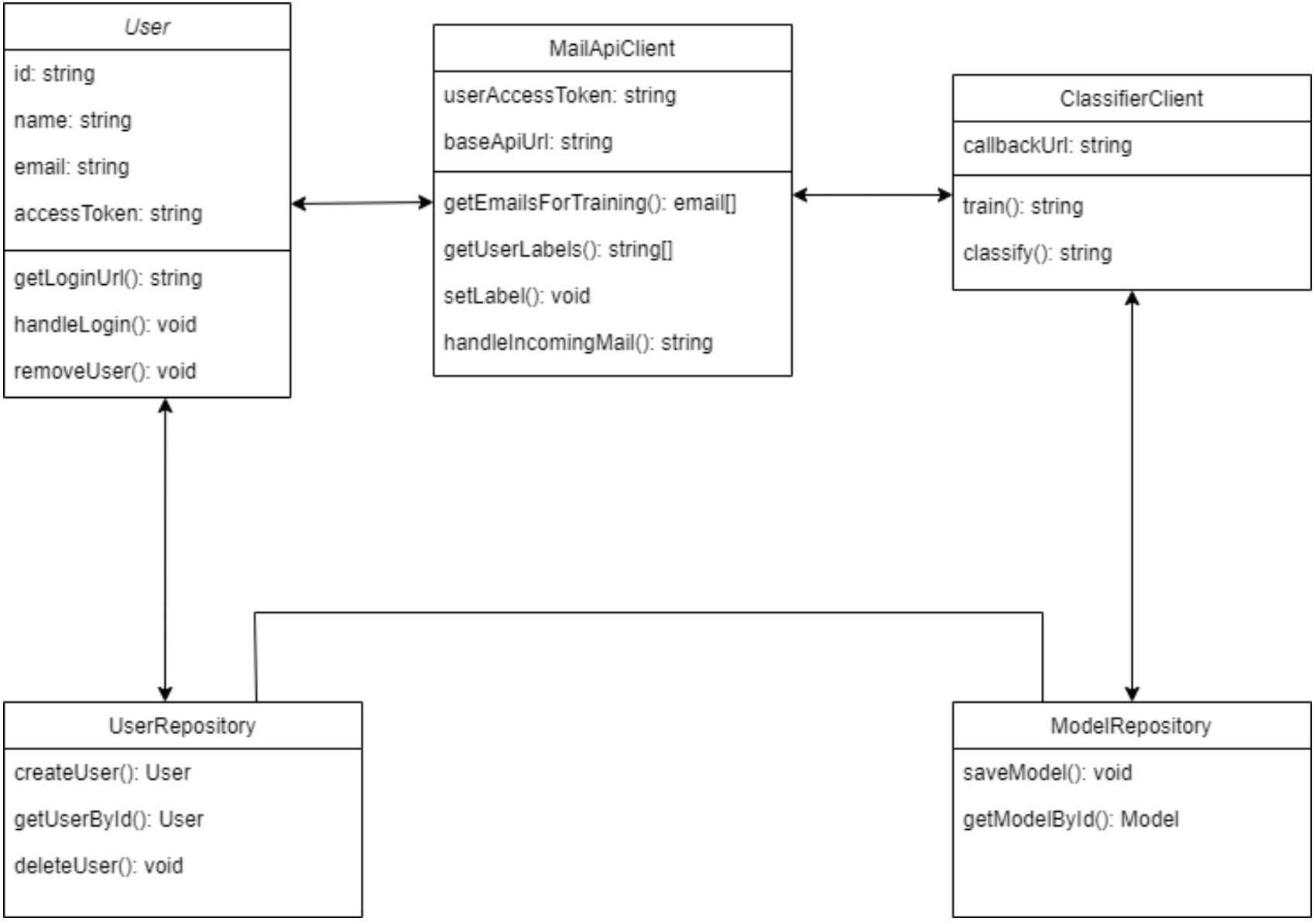


Схема структурна класів



Модель бази даних та файлового сховища системи