Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії
Кафедра комп’ютерних наук

# Пояснювальна записка

до магістерської кваліфікаційної роботи
на тему:

## Інформаційна технологія розпізнавання спам-повідомлень

Виконав: студент 2 курсу групи 2КН-19м
спеціальності 122 «Комп'ютерні науки»

Бондарчук В.Ю.
(прізвище та ініціали)

Керівник к.т.н., доц. Арсенюк І. Р.
(прізвище та ініціали)

Рецензент к.т.н., доц. Коваленко О. О.

Вінниця, 2020 рік

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра комп’ютерних наук

ЗАТВЕРДЖЕНО
Завідувач кафедри КН
проф., д.т.н. Яровий А. А.
«____»_____2020 р.

**З А В Д А Н Н Я**

на магістерську кваліфікаційну роботу на здобуття кваліфікації магістра наук зі спеціальності: 122 «Комп'ютерні науки»
(шифр – назва спеціальності)

08-22.МКР.018.19.000.ПЗ

Магістранта групи __2КН-19м__ _Бондарчука Віталія Юрійовича
(назва групи) (прізвище, ім’я і по батькові)

Тема магістерської кваліфікаційної роботи: «Інформаційна технологія розпізнавання спам-повідомлень»

Вхідні дані: векторизований текст спам-повідомлення, розмічена навчальна вибірка спам-повідомлень, словник слів на базі розміченої навчальної вибірки, кросплатформна мова реалізації.

Короткий зміст частин магістерської кваліфікаційної роботи:

1. Графічна: Схема алгоритму інформаційної технології розпізнавання спам-повідомлень, структура інформаційної технології, метод *k*-найближчих сусідів, Метод опорних векторів, дерево ухвалення рішень, середовище VS Code.

2. Текстова (пояснювальна записка): вступ, аналіз предметної області розпізнавання спам-повідомлень, розробка інформаційної технології розпізнавання спам-повідомлень, програмна реалізація інформаційної технології розпізнавання спам-повідомлень, економічна частина, висновки, перелік використаних джерел, додатки.

## КАЛЕНДАРНИЙ ПЛАН ВИКОНАННЯ МКР

| № етапу | Назва етапу | Термін виконання | | Очікувані результати |
|---|---|---|---|---|
| | | початок | кінець | |
| 1 | Аналіз предметної області розпізнавання спам-повідомлень | | | Аналітичний огляд літературних джерел, задачі досліджень, розділ 1 ПЗ |
| 2 | Розробка методу та інформаційної технології розпізнавання спам-повідомлень | | | Метод, інформаційна технологія, розділ 2 |
| 3 | Програмна реалізація розробленої інформаційної технології, тестування | | | Програмне забезпечення, розділ 3 |
| 4 | Підготовка економічної частини | | | розділ 4 |
| 5 | Апробація та/або впровадження результатів дослідження | | | тези доповідей, авторського права на твір |
| 6 | Оформлення пояснювальної записки, графічного матеріалу та презентації | | | Пояснювальна записка, графічний матеріал, презентація |

Консультанти з окремих розділів магістерської кваліфікаційної роботи

1. Науковий керівник _____ _____канд. техн. наук, доцент___
(підпис) наук. ступінь, вчене звання (посада)

_____І. Р. Арсенюк_____
ініціали та прізвище

“____” _____20___ р.

2. Економічна частина _____ канд. екон. наук, доцент кафедри ЕПВМ
(підпис) наук. ступінь, вчене звання (посада)

_____М. В. Бальзан_____
ініціали та прізвище

“____” _____20___ р.

Дата попереднього захисту роботи “____” _____20___ р.

Рецензент _____ __канд. техн. наук, доц., доц. кафедри ПЗ___
(підпис) наук. ступінь, вчене звання (посада)

_____Коваленко О. О._____
ініціали та прізвище

Завдання видав
науковий керівник _____ ____канд. техн. наук, доцент_____
(підпис) наук. ступінь, вчене звання (посада)
_____І. Р. Арсенюк_____
ініціали та прізвище

“____” _____20___ р.

Завдання отримав магістрант _____ _____В. Ю. Бондарчук_____
(підпис) ініціали та прізвище

**ЗМІСТ**

# РЕФЕРАТ

Магістерська кваліфікаційна робота присвячена розробці інформаційної технології розпізнаванння спаму на основі наївного баєсового класифікатора.

У ході роботи здійснено аналіз предметної області розпізнавання спам-повідомлень. Розглянуто аналоги програмних засобів розпізнавання спам-повідомлень, а також проаналізовано їхні переваги та недоліки. Удосконалено метод наївного баєсового класифікатора шляхом врахування додаткових ознак, дозволяє підвищити точність розпізнавання спам-повідомлень. Розроблено інформаційну технологію розпізнавання спам-повідомлень на базі вдосконаленого наївного баєсового класифікатората. Програмний засіб реалізовано на мові програмування Python. Здійснено тестування даного застосунку.

## ABSTRACT

The master's qualification work is devoted to the development of information technology for spam recognition based on a naive Bayesian classifier.

In the course of work the analysis of the subject area of recognition of spam messages is carried out. Analogues of spam recognition software are considered, as well as their advantages and disadvantages are analyzed. The method of naive Bayesian classifier has been improved by taking into account additional features, it allows to increase the accuracy of spam message recognition. The information technology of spam message recognition on the basis of the improved naive Bayesian classifier is developed. The software is implemented in the Python programming language. This application has been tested.

# ВСТУП

**Актуальність теми дослідження.** Довгий час спам-розсилка була головною зброєю в руках рекламних компаній, шахраїв, хоч і з розвитком технологій електронна пошта перестала бути основним способом обміну повідомлень тому і способи поширення спаму змінилися.

Спам — масове розсилання кореспонденції рекламного чи іншого характеру людям, які не висловили бажання її одержувати. Передусім термін «спам» стосується рекламних електронних листів.

Термін «спам» почав вживатися з 1993 року, коли рекламні компанії стали публікувати в групах новин Usenet, дискусійних листах, гостьових книгах повідомлення, що не мають відношення до заданої тематики, або повідомлення, які є прямою рекламою [3].

Спамерам розсилання практично нічого не коштують, проте дорого обходяться одержувачу спаму, якому доводиться оплачувати своєму провайдерові час, витрачений на одержання непрошеної кореспонденції з поштового сервера. А також через масовість поштових розсилань вони ускладнюють роботу інформаційних систем і ресурсів, створюючи для них непотрібне навантаження. Користувачі інтернету, крім того, змушені щодня витрачати час на фільтрацію рекламних повідомлень [4]. Для того щоб обмежити цей час, користувачі змушені використовувати протиспамові фільтри, які можуть випадково стерти й важливе повідомлення, вважаючи його спамом. Втім, і людина, яка змушена переглядати десятки рекламних повідомлень у день, теж легко може пропустити серед них потрібне.

Саме тому потреба у розробці інформаційної технології, яка вирішує задачу розпізнавання спаму є досить актуальною.

**Зв'язок роботи з науковими програмами, планами, темами.**

Магістерська робота виконана у відповідності з напрямком наукових досліджень кафедри комп’ютерних наук Вінницького національного технічного університету 22 К1 «Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

**Мета та завдання дослідження.**

Метою досліджень магістерської кваліфікаційної роботи є підвищення точності розпізнавання спаму.

Для досягнення поставленої вище мети слід розв'язати такі завдання:

- здійснити огляд та аналіз існуючих програмних реалізацій розв’язання задачі розпізнавання спаму;
- запропонувати математичну модель для інформаційної технології розпізнавання спаму;
- визначити стадії інформаційної технології та на їх основі розробити структуру та алгоритм роботи програмного засобу;
- здійснити програмну реалізацію спроектованої інформаційної технології розпізнавання спаму;
- здійснити тестування програмного засобу та виконати аналіз отриманих результатів.

**Об’єкт дослідження** – процес розпізнавання спаму.

**Предмет дослідження** – інформаційна технологія розпізнавання спам-повідомлень.

**Метою дослідження** магістерської кваліфікаційної роботи є підвищення точності автоматичного розпізнавання спаму.

**Область застосування –** захист від спаму у різних організаціях, закладах та компаніях.

**Методи дослідження.** У роботі використані такі методи наукових досліджень: методи обробки природної мови, теорія штучних нейронних мереж для

реалізації інформаційної технології розпізнавання спаму, методи математичної статистики та комбінаторики для обрахунків результатів отриманих під час роботи програмного засобу, програмування на мовах високого рівня.

**Наукова новизна одержаних результатів полягає у наступному:**

Удосконалено архітектуру наївного баєсового класифікатора, що відрізняється від відомих, врахуванням додаткових ознак.

**Практичне значення одержаних результатів:**

1. Розроблено програмний продукт розпізнавання спаму з використанням модифікованого наївного баєсового класифікатора.
2. Розроблено алгоритм розпізнавання спаму з використанням модифікованого наївного баєсового класифікатора.
3. Розроблено структура інформаційної технології розпізнавання спам-повідомлень.

**Достовірність теоретичних** положень магістерської кваліфікаційної роботи погоджується строгістю постановки задачі, коректними застосуваннями математичних методів під час підтвердження наукових положень, строгим доведенням аналітичних відношень, порівнянням результату з відомими, та відповідністю результатів математичного моделювання з результатами, що отдержано під час впровадження програмного засобу.

**Особистий внесок здобувача.** Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно.

**Апробація результатів роботи.** Результати досліджень апробовані на науково-технічній конференції підрозділів ВНТУ.

**Публікації.** За результатами досліджень подано заяву про реєстрацію авторського права на твір.

# 1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗПІЗНАВАННЯ СПАМУ

## 1.1 Аналіз задачі обробки природної мови

Обро́бка приро́дної мо́ви (англ. *Natural-language processing, NLP*) — загальний напрям інформаціних технологій, штучного інтелекту і математичної лінгвістики [5]. Він розглядає проблеми комп'ютерного аналізу і синтезу природної мови. Згідно зі штучним інтелектом аналіз означає розуміння мови, в той час як синтез — за генерування тексту такого, який може написати людина. Розв'язання цих проблем надасть можливість створення зручнішої форми взаємодії комп'ютера та людини. Для кращого уявлення що являє собою NLP нижче на рисунку 1.1 зображена типова архітектура систем аналізу обробки природної мови з використанням сучасних технологій.

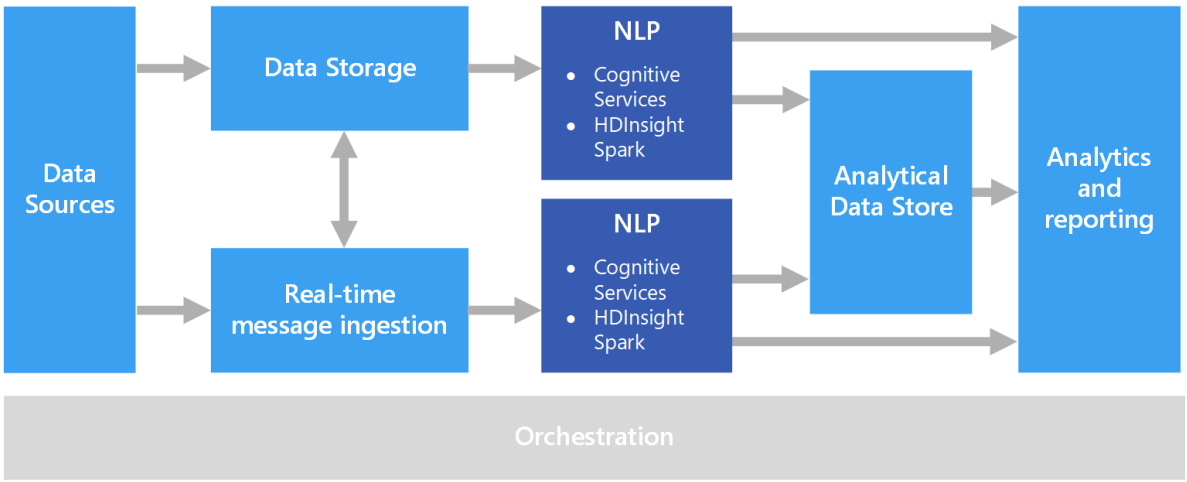


Рисунок 1.1 – Типова архітектура систем аналізу обробки природної мови з використанням сучасних технологій

Перша фаза (кінець 1940-х — кінець 1960-х)

Перша фаза розвитку обробки природної мови проходила з кінця 1940-х до кінця 1960-х років. У ці роки робота над на мащинним перекладом [6]. У 1952 році пройшла перша міжнародна конференція, де машинний переклад був нагальним питанням. На Джорджтаунському експерименті (реліз нової технології, 1954 р.) був представлений перший приклад машинного перекладу, а самеперекладу з англійської на російську. 1954-й рік запам'ятався не лише через перше демо машинного перекладу, а й через публікацію першого в світі випуску журналу «Механічний переклад». Завершенням данного етапу була Теддінгтонська міжнародна конференція з мовного аналізу та машинного перекладу, проведена у 1961 році, на ній організовано представленння звершень різних країн у галузях синтаксису, морфології, інтерпретації і семантики. Такий проміжок часу запам'ятався активністю та самовіддачею. Не беручи до уваги такі речі, як не високий технологічний розвиток, що у великій мірі ускладнював переробку даних, науковці приймали виклик складним завданням, які поставали перед ними у цей період.

Друга фаза (кінець 1960-х — кінець 1970-х)

Другий етап розвитку обробки природної мови відбувся в кінці 1960-х і завершився в кінці 1970-х років і пов'язаний був зі штучним інтелектом [7]. Розробники в межах цього періоду часу найбільшу частину своєї дорогоцінної уваги приділяли знаню про світ та формуванню певних значень у мовленні. Першою у світі аплікацією сприйняття природної мови був SHRDLU, реліз якої припав на 1972 рік. Ця програма могла розуміти велику кількість англомовних слів та могла робити певні висновки. Це було великим досягненням у галузі штучного інтелекту, однак додатки такого виду стикались із реальними ситуаціями, які подолати їм не вдавалось.

Третя фаза (кінець 1970-х — кінець 1980-х)

Третій етап розвитку обробки природної мови протікав з кінця 1970-х та до кінця 1980-х років. В цьому етапі велись роботи над такими областями як, штучний інтелект та семантика. З'явилась гостра необхідність уці оскільки вона використовується в штучному інтелекті під час обробки природної мови. В практичному сенсі лінгвісти розробили цілу низку граматичних типів, наприклад, функціональний та категоріальний, які були орієнтовані на обчислюваність. Даний період також характеризується стрімким розвитком логічного програмування з метою навчання програм обробки певних текстів.

Четверта фаза (1990-ті)

Четвертий етап почався у 1990-х, в 1990-му набрав популярності лексичний метод застосування до граматики [8]. Одне з авангардних місць в обробці природної мови очолював статистичний підхід, який давав змогу тепер не просто аналізувати данні, але й реально застосовувати цей алгоритм для обробки природної мови. Останніх десять років 20-го століття характеризувалось використанням методики спрощення текстів для виокремлення значущих одиниць з певного потоку інформації.

Розуміння природної мови деколи вважають AI-повною задачею, тому що розпізнавання живої мови потребує значних знань системи про довколишнє середовище і можливості взаємодіяти з ним. Саме означення змісту слова «розуміти» —головна задача штучного інтелекту. Сьогодні велику роль у розв'язанні задач з обробки природномовних даних займають онтології, наприклад, WordNet, UMN. У процесі дослідження обробки природної мови було досягнуто значних результатів, серед яких розробка потужних лексикографічних потужностей, додатків для машинного перекладу, електронних словників та ін. Але, є проблема, яка досі не була вирішена, вона закладається саме уприроді людської мови. Проблема розуміння людського мовлення є саме у його неоднозначності. Можна виділити такі види неоднозначностей:

Синтаксична неоднозначність*:* у прислів'ї «Час — не кінь, не підженеш і не зупиниш» для обробки мови буде абсолютно не зрозумілим те, про що саме йдеться у реченні, про коня чи про час.

Смислова неоднозначність*:* у реченні «Де знайти ключ до того замку?» слово *замо* може приймати два зовсім різні значення, залежно від того як падає наголос на слово [9].

Відмінкова неоднозначність: у фразах «Усі були схвильовані перед матчем» та «Не будемо загадувати на перед!» слово *перед* означає час або місце, що абсолютно змінює сенс речення.

Референційна неоднозначність: у фразі «Відкрий поличку та дістань мокру парасольку, я хочу її висушити» займенник *її* за смисловим значенням матиме відношення до мокрої парасолі, однак для машини, у якої повністю відсутнє розуміння реальності, даний займенник відноситиметься як до полиці, так і до парасолі [10] .

Одним із викликів, який виникає у процесі обробки природної мови, можна вважати проблему синонімії, в результаті якої одне поняття може бути вираженим декількома різними словами. Як наслідок, релевантні документи, в яких використано синоніми понять, що було вказано користувачем у запиті, може бути не визначено системою.

Вплив вищеперелічених явищ є особливо відчутним при створенні систем машинного перекладу. Проблема полягає у складності встановлення конкретного відображення дійсної семантико-синтаксичної структури речення у його внутрішнє логічне уявлення, яке автоматично генерується системою.[5]

Розв'язання таких типів неоднозначностей можливе за допомогою введення додаткових значень, які збільшать знання програми про ту чи іншу галузь. Сьогодні програм, які «розуміють» усі типи неоднозначностей у великому спектрі галузей,

не існує, проте є програми, що можуть коректно реагувати на неоднозначності у дуже вузьких сферах.

Статистичний підхід

В основі статистичного підходу до обробки природної мови лежить припущення, що зміст тексту може бути визначено за найуживанішими словами. Основним завданням даного підходу є визначення кількості повторень конкретного слова в тексті [11]. Латентно-семантичний підхід є різновидом статистичного методу та базується на ідеї, що сукупність усіх контекстів, у яких зустрічається або не зустрічається дане слово, визначає множину взаємних обмежень для виявлення схожостей у значеннях слів. Основна проблема, з якою стикаються статистичні підходи, полягає в розгляді тексту як набору слів без смислового зв'язку.

Лінгвістичний підхід

Лінгвістичний підхід до обробки природної мови складається з чотирьох рівнів [12]: графематичного, морфологічного, синтаксичного та семантичного. Перший рівень полягає у виділенні окремих компонентів тексту/документу, наприклад, розділів, абзаців, речень і т. д. Другий рівень полягає у визначенні морфологічних характеристик окремого слова. Третій рівень відповідає за визначення синтаксичної залежності слів у реченнях. Останній рівень пов'язаний зі смисловим розумінням тексту, що включає розробки у сфері штучного інтелекту. Дослідницькі досягнення у цій сфері є дуже обмеженими у зв'язку зі складністю людської мови.

Символічний підхід

Символічний підхід до обробки природної мови здійснює глибинний аналіз лінгвістичних явищ та базується на явному представленні знань, що здійснюється шляхом використання добре досліджених схем представлення знань та алгоритмів,

що працюють з ними [13]. Джерелом знання про мову можуть виступати словники, формули та правила, розроблені людьми.

Коннективістський підхід

Даний метод обробки природної мови відповідає за обробку загальних моделей з використанням конкретних прикладів мовних явищ. Найбільш значуща відмінність коннективістського підходу від інших статистичних методів полягає у поєднанні статистичних знань та різних теорій уявлень, що дозволяють працювати з логічними висновками та трансформацією логічних формул.

Метод допоміжних векторів

Диференційний метод машинного навчання, що допомагає провести класифікацію слів за категоріями. Даний метод побудований на певній множині властивостей [14].

Прихована марковська модель

Це така графічна система, у якій кожна вершина представляє собою випадкову змінну, що може набувати будь-якого значення (з певними ймовірностями) між декількома станами, породжуючи при цьому один з декількох можливих вихідних символів з кожним переходом. Множина всіх можливих станів та унікальних символів може бути великою. Ми можемо бачити вихідні дані, проте початкові стани системи є прихованими.

Умовні випадкові поля

Роздільна (диференційна) модель, яка формує логістичну регресію для послідовності даних. Використовується для передбачення стану змінної, що базується на спостереженій змінній [15].

N-грамні моделі

Модель побудована на послідовні з *n* елементів: речень, слів, букв, звуків і т. д. Модель дозволяє розрахувати ймовірність появи будь-якого елемента за відомих ймовірностей появи таких попередніх елементів. Така модель зводиться до скінченної множини ймовірностей, кожну з яких може бути оцінено після обчислення повторюваності відповідних *n*-грам [15].

## 1.2 Аналіз задачі розпізнавання спам-повідомлень

Найбільший потік спаму поширюється через електронну пошту (e-mail). Рекордним роком став 2016 рік, коли потік спаму в загальному трафіку електронної пошти становив 65 % (за даними Cisco Systems) [17].

Спамери збирають e-mail адреси за допомогою спеціального робота або вручну (рідко), використовуючи веб-сторінки, конференції Usenet, списки розсилок, електронні дошки оголошень, гостьові книги, чати тощо. Такі програми-роботи здатні зібрати за годину тисячі адрес і створити з них базу даних для подальшого розсилання на них спаму. Деякі компанії займаються тільки збором адрес, а бази потім продають. Деякі компанії продають спамерам e-mail адреси своїх клієнтів, що замовили в них товари чи послуги електронною поштою.

Для розсилання спаму використовуються підключені до інтернету погано захищені комп'ютери. Це можуть бути:

- Сервери, які помилково сконфігуровані так, що дозволяють вільне пересилання пошти (open relay, open proxy).
- Webmail сервіси, які дозволяють анонімний доступ чи доступ з простою реєстрацією нових користувачів (яку можуть виконати спеціальні програми-роботи).
- Комп'ютер-зомбі. Деякі спамери використовують відомі уразливості в програмному забезпеченні чи комп'ютерні віруси для того, щоб захопити

керування великою кількістю комп'ютерів, підключених до інтернету і використовувати їх для розсилання спаму.

Для ускладнення автоматичної фільтрації спаму повідомлення часто спотворюються — замість букв використовуються схожі цифри, латинські букви замість кирилиці, у випадкових місцях додаються пропуски тощо [18].

Застосовуються різні хитрощі для того, щоб переконатися, що повідомлення отримане й прочитане. Серед них:

– Підтвердження про доставку. Деякі поштові клієнти можуть відправляти його автоматично.
– Листи з зображеннями, які завантажуються із сайтів, контрольованих спамерами.
– Посилання на веб-сторінки, на яких пропонується одержати додаткову інформацію.
– Пропозиція відмінити підписку на цю розсилку, пославши листа на вказану адресу.

Якщо спамери одержують підтвердження, що поштова адреса дійсно використовується, потік спаму може багаторазово збільшитися.

## 1.2 Аналітичний огляд аналогів

### Proofpoint Essentials

Proofpoint пропонує потужні функції захисту від загроз [19]. Вони пропонують провідний фільтр спаму, домовившись про рівень обслуговування, що вони заблокують> 99% спаму. Proofpoint також фільтрує вміст у електронних листах, включаючи вміст для дорослих та маркетинговий вміст, який вони класифікують як сірий лист. SE Labs виявили, що Proofpoint мав найвищий загальний рейтинг точності (98%) серед усіх перевірених ними постачальників електронної пошти. Proofpoint економить час IT-відділів, дозволяючи кінцевим користувачам керувати власними карантинами електронної пошти. Користувачі

можуть переглядати архів усіх своїх минулих електронних листів, дозволяти доставляти електронні листи, які Proofpoint класифікує як збереження, та блокувати неприємних відправників. Proofpoint Essentials - ідеальна платформа для підприємств, які хочуть захистити свої електронні листи від загрози спаму та шкідливого програмного забезпечення[3].

Minecast

Mimecast забезпечує надійний багаторівневий захист від загроз для боротьби з вірусами та спамом, а також цілеспрямованими атаками електронної пошти, такими як фішинг [20]. Mimecast обіцяє забезпечити 100% захист від шкідливих програм з 99% розпізнавання анти-спаму. Mimecast пропонує інструменти для видалення сірої пошти та списків розсилки, як з адміністратором, так і з кінцевими користувачами. Вони також пропонують розширені функції, такі як Контроль вмісту та DLP, які захищають важливі дані компанії. Mimecast працює від їхньої групи розвідки загроз, глобальної мережі аналітиків, яка використовує інформацію з мільярдів електронних листів, що відстежуються щомісяця. Mimecast доступний як хмарне, так і локальне рішення, і доступний для бізнес-клієнтів.

Baracuda Essentials

Barracuda Essentials пропонує ряд функцій захисту як від вхідних, так і від вихідних загроз для бізнесу [21]. Використовуючи свої дані розвідувальної інформації про загрози, Barracuda ідентифікує домени електронної пошти для спаму, щоб заблокувати спам і сіру пошту. Вони також пропонують захист від фішингу, захист від шкідливих програм та захист посилань, щоб зупинити віруси, що передаються електронною поштою. Barracuda також фільтрує вихідні електронні листи, щоб зупинити використання ваших доменів для розсилки спаму. Barracuda Essentials - це повністю хмарна платформа захисту електронної пошти. Платформа Barracuda's Essentials також інтегрується з Sentinel, що є вдосконаленою платформою Barracuda для зупинки фішингу та атаки на фішинг в Office 365. Це

робить Barracuda надійним варіантом для організацій, які шукають захист від фішингу поряд із фільтрацією електронної пошти [22].

### 1.3 Постановка задачі

Необхідно розробити технологію розпізнавання спам-повідомлень. Для проектування інформіційної технології потрібно розв'язати настурні задачі:

1. Здійснити аналіз існуючих методів розпізнавання спаму, а також способів оцінки їх ефективності.
2. Визначити ознаки для використання в машинному навчанні з учителем.
3. Здійснити формування навчальної вибірки, та виконати її попередню обробку.
4. Розробити алгоритм розпізнавання спам-повідомлень.
5. Розробити структуру інформаційної технології розпізнавання спам-повідомлень
6. Здійснити вибір мови програмування.
7. Здійснити вибір середовища програмування
8. Здійснити реалізацію програмного засобу і проведення експериментальної оцінки результатів його роботи.

## 1.4 Висновок

У даному розділі виконано аналіз предметної області визначення спам-повідомлень. Проаналізовано основні способи поширення спаму серед яких можна відокремити: використання серверів, які помилково сконфігуровані; використання webmail-сервісів, які дозволяють анонімний доступ; використання комп'ютерів-зомбі тощо. Деякі спамери використовують відомі уразливості в програмному забезпеченні чи комп'ютерні віруси для того, щоб захопити керування великою кількістю комп'ютерів.

Виявлено, що для ускладнення автоматичної фільтрації спаму повідомлення часто спотворюються. Застосовуються різні хитрощі для того, щоб переконатися, що повідомлення отримане й прочитане, такі як лист із підтвердженням про доставку, листи з зображеннями, які завантажуються із сайтів, контрольованих спамерами, посилання на веб-сторінки, на яких пропонується одержати додаткову інформацію, пропозиція відмінити підписку на цю розсилку, пославши листа на вказану адресу тощо.

Розглянуто аналоги програмних засобів, зокрема Minecast, Baracuda Essentials. Mimecast забезпечує надійний багаторівневий захист від загроз для боротьби з вірусами та спамом, а також цілеспрямованими атаками електронної пошти, такими як фішинг. Barracuda Essentials пропонує ряд функцій захисту як від вхідних, так і від вихідних загроз для бізнесу. Ці програмні засоби мають такі основні недоліки: низька швидкість векторизації, неточність виявлення спаму.

Враховуючи недоліки існуючих програмних засобів здійснено постановку задачі, виконання якої дозволить підвищити точність визначення спаму.

# 2. РОЗРОБКА АЛГОРИТМУ РОЗПІЗНАВАННЯ СПАМ-ПОВІДОМЛЕНЬ

## 2.1 Дослідження основних методів визначення спам-повідомлень

Для класифікації текстів за допомогою машинного навчання із вчителем існують кілька відомих алгоритмів:

- наївний байєсовський класифікатор;
- метод k-найближчих сусідів;
- метод опорних векторів;
- метод логістичної регресії;
- дерево рішень;
- random forest.

### 2.1.1 Аналіз наївного баєсового класифікатора

Наївний баєсів метод – це метод класифікації, заснований на теоремі Баєса з припущенням про незалежність ознак. Він передбачає, що наявність якого-небудь ознаки в класі не пов'язано з наявністю будь-якого іншого ознаки [22].

$$P\ (c \mid d\ ) = \frac{P(c)P(d \mid c)}{P(d)},$$

де $P\ (c \mid d)$ – ймовірність, що документ d належить класу c; $P\ (d \mid c)$ – ймовірність зустріти документ d серед всіх документів класу c; $P\ (c)$ – безумовна ймовірність зустріти документ класу c в корпусі документів; $P\ (d)$ – безумовна ймовірність документа $d$ в корпусі документів.

Теорема дозволяє розрахувати ймовірність того, що саме ця причина привела до спостережуваного події.

Для того, щоб визначити найбільш ймовірний клас, до якого належить документ, потрібно скористатися оцінкою апостеріорного максимуму. Іншими словами, потрібно розрахувати ймовірність для всіх класів і вибрати той клас, який володіє максимальною вірогідністю [23].

$$c_{map} = arg\ max_{c \in C} [P\ (c \mid d).\quad P(c)].\ (*)$$

Згідноз баєсівим класифікатором, документ - набір слів, ймовірності яких умовно не залежать одне від одного. В результаті чого, умовна ймовірність документа апроксимується твором умовних ймовірностей всіх слів, що входять в документ.

$$P(d|c) \approx P(w_1|c)P(w_2|c) \ldots P(w_n|c) = \prod\nolimits_{i=1}^{n} \cdot P(w_i|c).$$

Підставивши отримані вирази в формулу (*) отримаємо:

$$c_{map} = \arg max_{c\ \epsilon\ C} \left[P(c) \cdot \prod\nolimits_{i=1}^{n} P(w_1|c)\right].$$

При порівняно великому обсязі документа необхідно перемножати велику кількість маленьких чисел. Тому, щоб уникнути арифметичного переповнення, знизу можна скористатися властивістю логарифма добутку *log ab = log a + log b*. Так як логарифм є монотонною функцією, то його застосування до обох частин виразу змінить тільки чисельне значення, але не параметри, при яких досягається максимум. Формула (1.4) з використанням логарифма [24]:

$$c_{map} = \arg max_{c\ \epsilon\ C} \left[P(c) \cdot \sum\nolimits_{i=1}^{n} \log P(w_i|c)\right].$$

Ймовірність класу:

$$P(c) = \frac{D_c}{D},$$

де $D_c$ – кількість документів, що належать класу c, D – загальна кількість документів у вибірці.

Імовірність слова в класі:

$$P(w_i c) = \frac{W_{ic}}{\sum_{i` \epsilon V} W_{i` c}},$$

де $W_{ic}$ – кількість разів скільки і-те слово зустрічається в документах класу $c$, $V$ – кількість слів у всіх документах класу $c$.

Але якщо зустрілося слово, якого немає в документах класу, то $P\ (\ W_i|\ C)$ буде дорівнювати нулю. Вирішити дану проблему можна домовившись, що будь-яке слово в класі ми зустрічаємо мінімум один раз. Ось чому класифікатор називається наївним баїсовим. отримуємо підсумкову формулу [24]:

$$c_{map} = \arg max_{c\ \epsilon\ C} \left[ P(c) \cdot \sum_{i=1}^{n} \log \frac{W_{ic} + 1}{|V| + \sum_{i` \epsilon V} W_{i` c}} \right].$$

Основні переваги даного класифікатора:

– низька обчислювальна складність;

– здатність застосування на великих наборах даних.

### 2.1.2 Аналіз метод опорних векторів

Метод опорних векторів відноситься до сімейства лінійних класифікаторів. Завданням лінійної класифікації є пошук гіперплощини в просторі ознак, що розділяє всі об'єкти на класи.

Основна ідея методу опорних векторів полягає в пошуку розділяючої гіперплощини, максимально віддаленої від найближчих до неї точок в просторі ознак. Якщо класи не можна відразу лінійнорозділити, то алгоритм буде додавати новий вимір в прагненні подальшого поділу. Він буде продовжувати цей процес, поки не буде здатний розділити ознаки на два окремих класи.

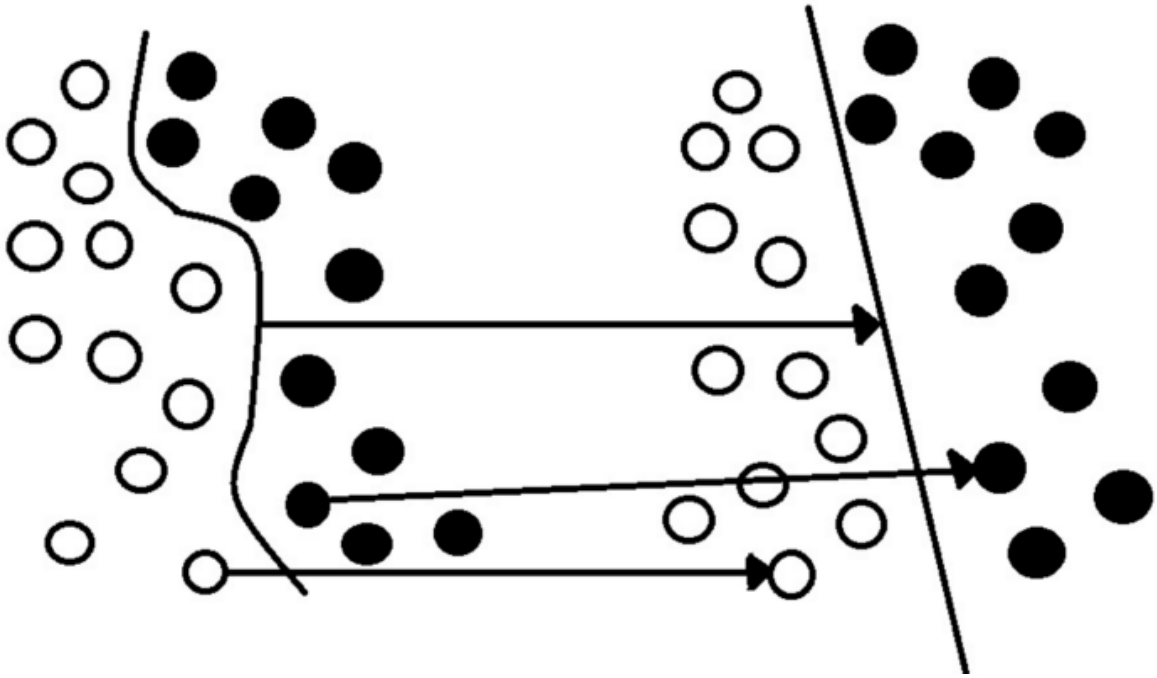

Рисунок 1.2 – Приклад поділу на класи методом опорних векторів

Після знаходження алгоритмом гіперплощини, що забезпечує максимальний рівень відмінності між класами, він може приступити до класифікації нових даних, які класифікуються в залежності від того, по який бік від гіперплощини вони знаходяться.

Однак у даного методу є істотний недолік: при збільшенні кількості вимірювань, вектор ознак збільшується в геометричній прогресії.

### 2.1.3 Аналіз метода *k*-найближчих сусідів

Метод k найближчих сусідів - найпростіший алгоритм класифікації. Основна ідея: об'єкт належить до того класу, до якого належить більшість найближчих його сусідів. Візуалізація метода k-найближчих сусідів зображена на рисунку 1.3

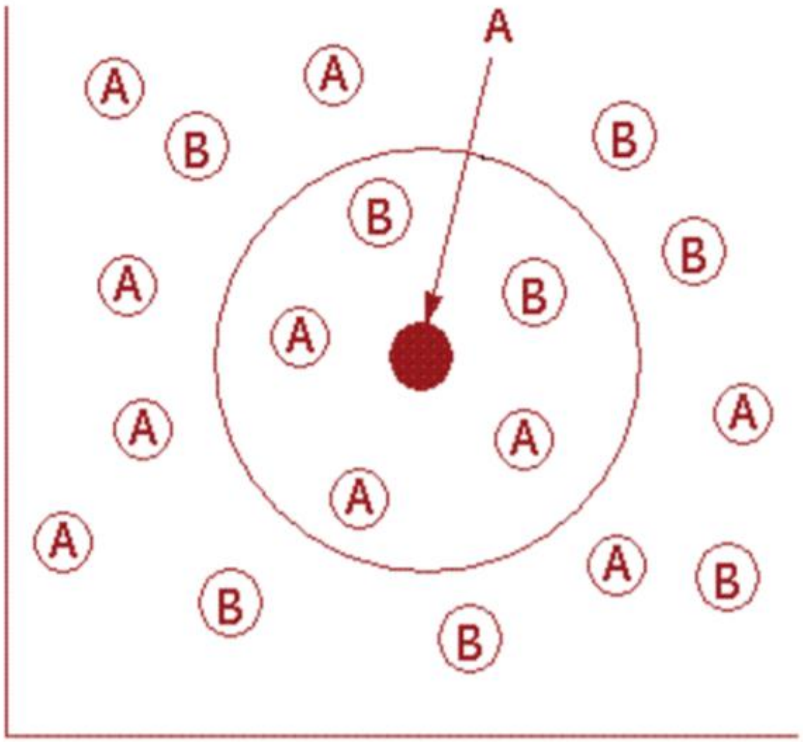


Рисунок 1.3 - Метод k найближчих сусідів

K – кількість сусідніх об'єктів, які порівнюються з класифікованих об'єктом. Якщо k = 5, то об'єкт буде порівнюватися з 5 сусідами.

Алгоритм:

1. Підготувати розмічену (навчальну) вибірку об'єктів, тобто вказати до якого класу належить кожен об'єкт.

2. Обчислити відстань до кожного з об'єктів навчальної вибірки. В більшості алгоритмів відстань розраховується за допомогою Евклідової відстані:

$$d(x_i, x_j) = \sqrt{\sum_{l=1}^{m}(x_i^{(l)} - x_j^{(l)})^2},$$

де $x_i = (x_i^{(1)}, \dots, x_i^m)$ – вектор m признаків і-го об`єкта, $x_j = (x_j^{(1)}, \dots, x_j^m)$ вектор m признаків j-го об`єкта.

3. Відібрати *k* об'єктів навчальної вибірки, відстань до яких мінімальна.
4. Клас класифікуємого об'єкта - це клас, який найбільш часто зустрічається серед *k* найближчих сусідів.

Якщо значення *k* буде маленьким, то може виявитися, що єдиним найближчим об'єктом буде об'єкт з неправильно певним класом, який дасть невірне рішення, такі випадки називають «викид». якщо *k* матиме велике значення (N), то тоді «переможе» найпопулярніший клас. В такому випадку, відстань до классифицируемого об'єкта не має ролі. Компромісом вважають, коли *k = 4*. При впевненості, що все об'єкти вибірки коректно класифіковані, можна брати k меншим [6].

Недоліки алгоритму:

– якщо серед навчальних об'єктів є об'єкт невірно класифікований, то і найближчі до нього об'єкти, також будуть класифіковані невірно;
– алгоритм повністю залежить від того, наскільки вдало вибрані ознаки.

### 2.1.4 Аналіз метода логістичної регресії

Логістична регресія застосовується для передбачення ймовірності виникнення деякої події за значеннями безлічі ознак. При цьому ознаки можуть як числовими, так і категоріальними. наприклад, ймовірність, того, що у людини

станеться серцевий напад в певний період часу, може бути передбачена в залежності від віку людини, статі та індексу маси тіла.

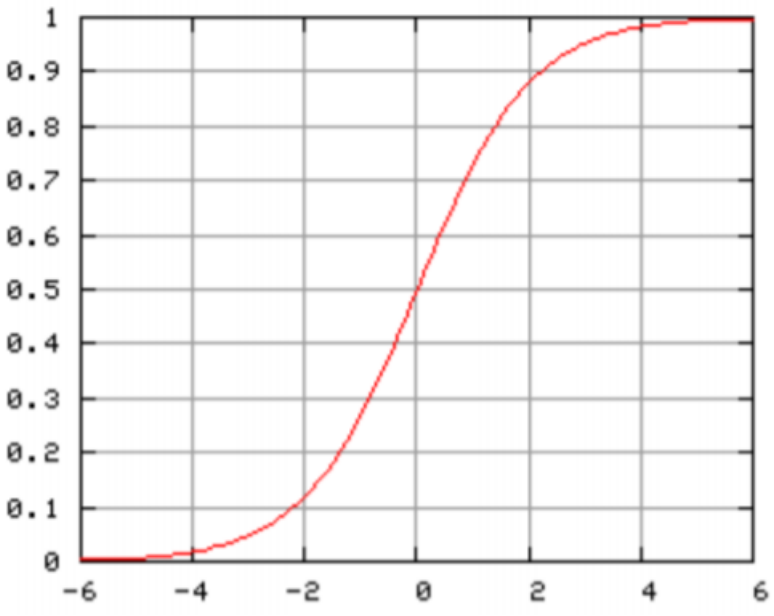


Рисунок 1.4 – Логічна функція з z на горизонтальній осі і f (z) на вертикальної осі

Логістична функція:

$$f(x) = \frac{1}{1+e^{-z}},$$

де змінна z включає в себе набір факторів ризику, а f(x) ймовірність конкретного результату, при заданому наборі ризиків.

$$z = \beta_0 + \beta_1 x_1 + \cdots + \beta_k x_k,$$

де $\beta_0$- «точка перетину», тобто величина z, при нульових значеннях факторів ризику; $\beta_0, \beta_1, \ldots, \beta_k$ - «коефіцієнт регресії» для факторів ризику $x_1, x_2, x_3$

Якщо коефіцієнт регресії позитивний, то це означає, що збільшується ймовірність аналізованого результату. якщо коефіцієнт негативний, то ймовірність

аналізованого результату зменшується. Великий коефіцієнт регресії означає, що даний фактор значно впливає на сукупний ризик. Якщо коефіцієнт регресії близький до нуля, то цей фактор має невеликий вплив на ймовірність результату [26].

### 2.1.5 Аналіз метода random forest

Випадкові ліси (Random Forest) або ліси випадкових рішень - алгоритм машинного навчання, запропонований Лео Брейманом [2] і Адель Катлер (англ.) рос., що полягає у використанні комітету (ансамблю) вирішальних дерев. Алгоритм поєднує в собі дві основні ідеї: метод беггінга Брейман, і метод випадкових підпросторів (англ.) Рос., Запропонований Тін Кам Хо (англ.) Рос .. Алгоритм застосовується для задач класифікації, регресії і кластеризації. Основна ідея полягає в використанні великого ансамблю вирішальних дерев, кожне з яких саме по собі дає дуже невисока якість класифікації, але за рахунок їх великої кількості результат виходить хорошим. Рішення дерев є популярним методом для різних завдань машинного навчання. Вузьке вивчення дерева "наближається до відповідності вимогам, щоб виступати як процедура незавершеного виявлення даних", - стверджує Хасти та співавтори, - "оскільки вона є інваріантною при масштабуванні та інших різноманітних перетвореннях значень ознак, є надійною до включення невідповідних функцій, а також створює моделі для перевірки, однак вони рідко точні."

Нехай навчальна вибірка складається з N зразків, розмірність простору ознак дорівнює M, і заданий параметр m (в задачах класифікації зазвичай {\ displaystyle m \ approx {\ sqrt {M}}} m \ approx {\ sqrt {M}}) як неповне кількість ознак для навчання.

Найбільш поширений спосіб побудови дерев комітету наступний (називається беггінг (англ. Bagging, скорочення від англ. Bootstrap aggregation)):

1. Згенеруємо випадкову подвиборку з повтореннями розміром {\ displaystyle N} N з навчальної вибірки. Таким чином, деякі зразки потраплять в неї два або більше разів, а в середньому {\ displaystyle N (1-1 / N) ^ {N}} {\ displaystyle N (1-1 / N) ^ {N}} (при великих {\ displaystyle N} N приблизно {\ displaystyle N / e} {\ displaystyle N / e}, де {\ displaystyle e} e - основа натурального логарифма) зразків не увійдуть в неї взагалі. Ті зразки, які не були в вибірку, називаються out-of-bag (невідібрані).
2. Побудуємо вирішальне дерево, що класифікує зразки даної підвибірки, причому в ході створення чергового вузла дерева будемо вибирати набір ознак, на основі яких проводиться розбиття (не з усіх M ознак, а лише з m випадково обраних). Вибір найкращого з цих m ознак може здійснюватися різними способами. В оригінальному коді Брейман використовується критерій Джині, що застосовується також в алгоритмі побудови вирішальних дерев CART. У деяких реалізаціях алгоритму замість нього використовується критерій приросту інформації. [26]
3. Дерево будується до повного вичерпання підвибірки і не піддається процедурі прунінга (англ. Pruning - відсікання гілок) (на відміну від вирішальних дерев, побудованих за таким алгоритмам, як CART або C4.5).

Класифікація об'єктів проводиться шляхом голосування: кожне дерево комітету відносить класифікується об'єкт до одного з класів, і перемагає клас, за який проголосувала найбільша кількість дерев.

Оптимальне число дерев підбирається таким чином, щоб мінімізувати помилку класифікатора на тестовій вибірці. У разі її відсутності, мінімізується оцінка помилки out-of-bag: тих зразків, які не були в навчальну подвиборку за рахунок повторень (їх приблизно N / e).

### 2.1.6 Аналіз метода дерево ухвалення рішень

Дерево рішень - це інструмент підтримки прийняття рішень, який використовує деревоподібну модель рішень та їх можливі наслідки, включаючи випадкові результати, витрати ресурсів та корисність. Це один з способів відображення алгоритму, який містить лише твердження з умовним керуванням. Дерева рішень зазвичай використовуються в дослідженнях операцій, зокрема в аналізі рішень, щоб допомогти визначити стратегію, яка, найімовірніше, досягне мети, але також є популярним інструментом машинного навчання [26].

Дерево рішень є схемою, подібною до структури, в якій кожен внутрішній вузол представляє "тест" на атрибуті (наприклад, при підкиданні монети випав орел чи решка), кожна гілка являє собою результат тесту, а кожен вузол представляє собою розділення гілки (рішення приймається після обчислення всіх атрибутів). Шляхи від кореня до листа являють собою правила класифікації. У процесі аналізу рішень дерево рішень тісно пов'язане з діаграмою впливу і використовується як візуальний та аналітичний інструмент підтримки прийняття рішень, де розраховуються очікувані значення (або очікувана корисність) конкуруючих альтернатив.

Дерево рішень складається з трьох типів вузлів:

- Рішення вузлів - як правило, представлені квадратами.
- Шаблонні вузли - зазвичай представлені колами.
- Кінцеві вузли - зазвичай представлені трикутниками.

Рішення дерев зазвичай використовуються в операціях дослідження та управління операціями. Якщо на практиці рішення повинні бути прийняті в Інтернеті без відкликання за додатковими заннями, то дерево рішень має бути представлено як алгоритм моделювання в режимі он-лайн вибору. Інше використання дерев рішень є як описовий засіб для обчислення умовних ймовірностей.

Приклад дерева рішень зображено на рисунку 1.5.

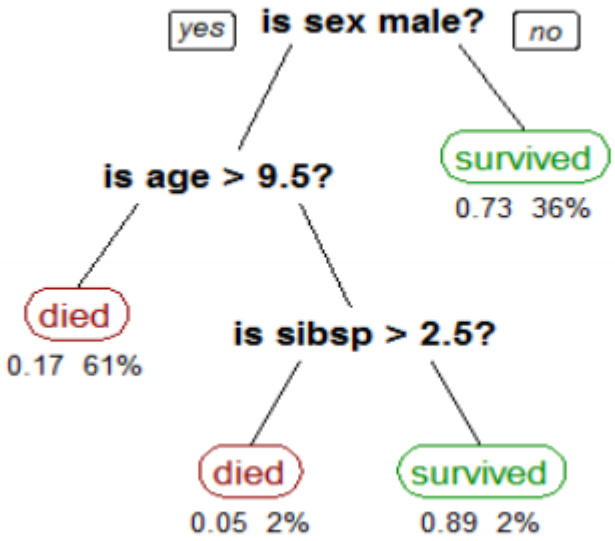


Рисунок 1.5 – Візуалізація класифікатора

Типологія дерев

Дерева рішень, які беруть участь в обробці даних, бувають двох основних видів:

- Дерево для класифікації, коли результат, що передбачається є класом, до якого належать дані.
- Дерево для регресії, коли результат, що передбачається можна розглядати як дійсне число (наприклад, ціна на будинок, або тривалість перебування пацієнта в лікарні).

Згадані вище терміни вперше були введені Брейманом. Перераховані типи мають деякі подібності (рекурсивний алгоритми побудови), а також деякі відмінності, такі, як критерії вибору розбиття в кожному вузлі. Деякі методи дозволяють побудувати більш одного дерева рішень (ансамблі дерев рішень):

- Беггінг над деревами рішень, найбільш ранній підхід. Будує кілька дерев рішень, неодноразово інтерполює дані з заміною (бутстреп), і в якості консенсусного відповіді видає результат голосування дерев (їх середній прогноз).
- Класифікатор «Випадковий ліс» заснований на беггінзі, проте як додаток до нього випадковим чином вибирає підмножину ознак в кожному вузлі, з метою зробити дерева більш незалежними.
- Бустінг над деревами може бути використаний для задач як регресії, так і класифікації. Одна з реалізацій бустінга над деревами, алгоритм XGBoost, неодноразово використовувався переможцями змагань з аналізу даних.
- «Обертання лісу» - дерева, в яких кожне дерево рішень аналізується першим застосуванням методу головних компонент (PCA) на випадкові підмножини вхідних функцій.

Алгоритми побудови дерева

Ми можемо описати загальну схему побудови дерев рішень, виділивши наступні аспекти:

- Обираємо якийсь атрибут Q, він стає нашим коренем дерева.
- Якщо атрибут Q має і значень, то : о В дереві ми залишаємо ті дані, у яких значення атрибута Q дорівнює і о Застосовуємо рекурсію і будуємо гілки головного дерева Є різні способи вибирати черговий атрибут.
- Алгоритм ID3, де вибір атрибута відбувається на підставі ентропії, або на підставі критерію Джині.
- Алгоритм C4.5 (поліпшена версія ID3), де вибір атрибута відбувається на підставі нормалізованого приросту інформації (англ. Gain Ratio).
- Алгоритм регресійних та класифікаційних дерев CART і їх модифікації різні модифікації.

– Автоматичний детектор взаємодії Хі-квадрат (CHAID). Виконує багаторівневе поділ при розрахунку класифікації дерев.
– MARS: розширює дерева рішень для поліпшення обробки цифрових даних.

На практиці в результаті роботи цих алгоритмів часто виходять занадто деталізовані дерева, які при їх подальшому застосуванні дають багато помилок. Це пов'язано з явищем перенавчання. Для скорочення дерев використовується відсікання гілок (англ. Pruning) [28].

## 2.2 Ознаки для використання в машинному навчанні з учителем

Ознаки поділяються на такі категорії: «графові» (що відображають характер і кількість зв'язків користувача з сусідами), «профільні» (що описують дані, відображені в профілі користувача), взаємодії (що відображають активність користувача) і «Твітові» (описують властивості змісту статусів користувача). Легко бачити, що більшість з них мають відображати використання користувачем будь-яких технік, описаних в попередньому розділі. Нижче наводяться описи ознак, що раніше використовувалися в роботах по визначенню спаму в Twitter, з поділом за категоріями, а також посиланням на джерело.

Графові ознаки:

1. Кількість друзів.
2. Кількість фоловерів.
3. Відношення кількості фоловерів до кількості друзів.
4. Репутація $R = \frac{N_{followers}}{N_{followers}+N_{friends}}$.

Також в якості признаків можуть використовуватисьтакі графові характеристики як ‘взаємність’ – відношення кількості взаємних до кількості вихідних ссилок для вершини користувача в соціальному графі «локальний коефіцієнт кластирізації»

$$LC = \frac{2|e^x|}{K_v(K_v-1)},$$

де $K_v$ – кількість вхідних і вихідних вершин для v, $e^v$ – кількість ребер на сусідах v «проміжне центрування»

$$BC = \frac{1}{(n-1)(n-2)} \sum_{s=v=t=V} \frac{\partial(v)}{\partial},$$

де $\partial$ – кількість коротших шляхів від $s$, $v$, $t$, $\partial$(м) – кількість тих із них що проходять через $v$.

Профільні ознаки:

1. Кількість статусів користувача.
2. Вік аккаунта в днях.
3. Ім’я акаунта містить спам-слова.

Ознаки на основі взаємодії [27]:

1. Кількість статусів-відповідей від користувача.
2. Кількість згадок на користувача.
3. Кількість унікальних згаданих користувачів.
4. Кількість статусів, проретвіченних іншими користувачами.
5. Показник взаємодії $IR = \frac{N_{tweets\ in\ friends}+N_{retweets}}{N_{total}}$.

Твітові ознаки:

1. Відсоток статусів, які містять посилання, серед останніх двадцяти.
2. Кількість хештегів по відношенню до кількості слів (мінімальне, медіана, максимальне, середнє арифметичне).
3. Кількість посилань по відношенню до кількості слів (мінімальне, медіана, максимальне, середнє арифметичне) [27].

4. Кількість символів в статусі (мінімальне, медіана, максимальне, середнє арифметичне) .
5. Кількість хештегів в статусі (мінімальне, медіана, максимальне, середнєарифметичне).
6. Кількість згадок в статусі (мінімальне, медіана, максимальне, середнє арифметичне) .
7. Кількість цифр в статусі (мінімальне, медіана, максимальне, середнє арифметичне) .
8. Кількість посилань в статусі (мінімальне, медіана, максимальне, середнє арифметичне) .
9. Кількість слів в статусі (мінімальне, медіана, максимальне, середнє арифметичне) .
10. Кількість ретвітів статуса (мінімальне, медіана, максимальне, середнє арифметичне) .
11. Проміжок часу між статусами (мінімальний, медіана, максимальний, середнє арифметичне) .
12. Кількість унікальних статусів серед останніх двадцяти.
13. Кількість статусів, що містять згадки, серед останніх двадцяти.
14. Кількість статусів, що містять хештеги, серед останніх двадцяти.
15. Кількість використаних в статусах хештегів.
16. Кількість використаних в статусах унікальних хештегов.
17. Кількість використаних в статусах посилань.
18. Кількість використаних в статусах унікальних посилань.
19. Середня частота повторення посилань.
20. Відсоток статусів, що містять спам-слова.

У роботі значимість ознак оцінювалася з використанням приросту інформації, найбільш значущими (в порядку зменшення) виявилися ознаки 17,13, 3, 16, 2. Також проводилася оцінка значущості по погіршення результатів класифікації для

класифікаторів Naive Bayesian, Jrip і J48 при видаленні оцінюваної ознаки. Таким чином було визначено, що для Twitter важливі ознаки, що характеризують використання користувачем посилань і згадок.

У роботі оцінка за методом хі-квадрат показала найбільшу значимість ознак

## 2.3 Формування навчальнї вибірки

Якість майбутнього класифікатора залежить від векторної моделі (моделі представлення текстів). Іншими словами, обраний набір ознак вектора має величезний вплив на точність майбутнього класифікатора.

Одна з найбільш часто використовуваних моделей - модель n-грам. В рамках моделі, вектори ознак являють собою послідовності з n слів. n може бути будь-яким цілим числом більше нуля; якщо n дорівнює одиниці, то така модель називається «уніграмм», якщо n дорівнює двом - «биграмм», трьом - «триграм», і так далі. У найпростішому випадку моделі уніграмм, вектор ознак являє собою набір всіх слів їх тексту, - тобто кожне слово є термін. У разі биграмм термін утворює пари слів з тексту. Це означає, що пара слів розглядає як ознака вектора. Для прикладу розглянемо пропозицію 'Jane prefers coffee to tea'. Воно може бути представлене як вектор уніграмм: [Jane; prefers; coffee; to; tea]. Крім цього, воно може бути виражено як набір биграмм: [[Jane prefers]; [Prefers coffee]; [Coffee to]; [To tea]]. Також може розглядатися комбінація уніграмм і биграмм окремо: [Jane; prefers; coffee; to; tea; [Jane prefers]; [Prefers coffee]; [Coffee to]; [To tea]].

Символьні n-грами повинні бути розглянуті окремо. Символьна n-грами означає n йдуть підряд символів з тексту. Наприклад, символьні біграми слова 'word' будуть виглядати наступним чином: '_w', 'wo', 'or', 'rd' and " d_. Символьні триграми того ж слова будуть: '_wo', 'wor', 'ord' and 'rd_'. Це може здаватися

примітивним способом, однак, подібна векторна модель мови показує непогані результати в деяких роботах [28].

Насамперед, необхідно представити колекцію документів у вигляді матриці термін-документ. Рядки будуть позначати окремі документи (тексти), а колонки - словник вибірки, що містить в собі всі слова в колекції документів.

Проілюструємо прикладом: дано два коротких документа; вони є прикладом найпростішої навчальної вибірки.

1. 'Jane likes coffee and tea'
2. 'Jane also likes cookies'

На даному етапі може бути сформований словник вибірки; він буде містити всі слова з текстів навчальної вибірки. Далі, можна буде сформувати матрицю термін-документ: значення в кожному осередку показує, чи зустрічається певний термін в опредленія документі (1) чи ні (0).

Таблиця 2 – Матриця термін-документ

| | *Jane* | *likes* | *coffee* | *and* | *tea* | *also* | *cookies* |
|---|---|---|---|---|---|---|---|
| 1$^{ый}$ текст | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| 2$^{ой}$ текст | 1 | 1 | 0 | 0 | 0 | 1 | 1 |

Таким чином, виходять бінарні вектори: $d_1$ = [1,1,1,1,1,0,0] і $d_2$ = [1,1,0,0,0,1,1].

Навчальна вибірка представляє собою заздалегідь розмічену (з позначенням емоційного забарвлення) колекцію документів, на основі якої буде складено тональний словник.

Як навчальної вибірки був обраний корпус російськомовних текстів, зібраних з соціальної мережі Twitter і розподілених на два класу: spam (114911 записів) і ham (111923 записів) [28].

Кожен текст в корпусі має наступні атрибути:

– текст твіти;

– дата публікації;
– ім'я автора;
– клас, до якого належить текст (позитивний, негативний);
– кількість копіювань цього повідомлення іншими користувачами;
– кількість друзів користувача;
– кількість користувачів, у яких даний автор в друзях.

Але в даній роботі використовуються тільки три атрибути: текст твіти, дата публікації і клас спаму тексту.'

## 2.4 Попередня обробка текстів

Повідомлення в соціальних мережах містять інформацію, що не є необхідною для аналізу спаму:

1. Хештег - являє собою слово або об'єднання слів, яким передує символ #, наприклад, # спорт, # Пітер. Використовується для полегшення пошуку повідомлень по темі або змісту.
2. Посилання - універсальний покажчик ресурсу.
3. Ім'я користувача - унікальне позначення користувача в соціальної мережі. У Twitter ім'я користувача починається з символу @, наприклад, @username.

Використовуючи регулярні вирази, відповідні дані були видалені з текстів.

Також були вилучені знаки пунктуації та усі великі букви приведені до рядковим.

### 2.4.1 Слова з максимальною частотою зустрічаваності

У даній роботі був складений частотний словник, в який спочатку увійшли 100 слів з максимальною частотою зустрічальності в російській мові згідно

«частотних словників сучасної російської мови» [29], який був побудований на основі «Національного корпусу англійської мови» .

Але так як більшість слів в складеному словнику виявилася займенниками, спілками, приводами, які видаляються за допомогою mystem-scala, то словник був скорочений до 29 слів. У таблиці 3 зображений частотний словник.

Таблиця 3 – частотний словник

| Слово | Частота зустрічаваності |
|---|---|
| the | 12160.7 |
| be | 3727.5 |
| to | 2912.3 |
| of | 2723.0 |
| and | 2396.5 |
| a | 2134.4 |
| in | 2039.9 |
| that | 1832.7 |
| have | 1712.4 |
| I | 1412.1 |
| it | 1389.6 |
| for | 1389.9 |
| not | 1233.4 |
| on | 1217.5 |

### 2.4.2 Обробка смайликів

Смайлик (смайл) - спосіб вираження емоцій у дописах. В тому числі, вони допомагають позбутися від неоднозначності [28].

Був складений словник смайликів, в який увійшли (рисунок 2.3):

- найпопулярніші текстові смайли [28];
- смайли з максимальною частотою зустрічальності в навчальній вибірці.

Для кожного з них був проставлений тип емоційного забарвлення (Позитивний, негативний)

### 2.4.3 Формування словника негативних слів

У соціальних мережах іноді вживають смайли відмінною забарвлення від тексту, що призводить до неоднозначності розпізнавання спаму. Наприклад, «Це якийсь жах!) Кошмар!)». В даному прикладі текст має явно негативне забарвлення, а смайлики - позитивну. Тому було вирішено створити словник негативних слів, в якому всі слова приведені до словникової формі. Тоді алгоритм буде наступний:

- Якщо текст негативний і містить слова з негативного словника, то незважаючи на смайлики, він буде віднесений до негативного класу спаму.
- Якщо текст негативний і не містить слів з негативного словника, але включає позитивні смайлики, то він буде віднесений до позитивного класу спаму.

В ході дослідження було з'ясовано: якщо наївний баєсів класифікатор відносить текст з імовірністю більшої або рівної 0.8 до позитивного класу спаму, то можна не враховувати до якого класу спаму належать смайлики, так як текст буде явно спамом. Наприклад: «Сьогодні чудовий день! (». В даному реченні користувач швидше за все помилився при вказівці смайлика; якщо наївний баєсів класифікатор

відносить текст з ймовірністю меншою або рівній 0.45 до позитивного або негативного класу, і текст не містить позитивних або негативних смайликів, то він має нейтральну забарвлення. Наприклад, твіт: «привеет. Як справи? »Не містить смайликів і слів з явною емоційним забарвленням, тому його можна віднести до нейтрального класу спаму.

## 2.5 Аналіз метрики ефективності інформаційної системи розпізнавання спам-повідомлень.

Як метрикики ефективності розпізнавання спам-повідомлень були обрані точність (precision) і повнота (recall). Точність в межах класу - це частка текстів, дійсно належать даному класу, щодо всіх текстів, зарахованих класифікатором до цього класу. Повнота системи - відношення числа знайдених класифікатором текстів, що належать класу, до числа всіх текстів цього класу в тестовій колекції.
В результаті класифікації текстів рецензій тестової вибірки, до класу позитивних (позитивних) рецензій правильно віднесені TP текстів, неправильно - FP, до класу негативних (негативних) рецензій правильно були віднесені TN текстів, неправильно - FN. Іншими словами:

- TP - істинно-позитивне рішення.
- TN - істинно-негативне рішення.
- FP - хибно-позитивне рішення.
- FN - хибно-негативне рішення.

У таблиці нижче наочно представлені результати класифікації.

Таблиця 1 – Результати класифікації

| Документи | | Експертна оцінка | |
|---|---|---|---|
| | | Позитивна | Негативна |
| Оцінка системи | Позитивна | TP | FP |
| | Негативна | FN | TN |

Тоді, щодо класу позитивних рецензій, точність Precision і повнота Recall визначаються наступним чином:

$$Precision = \frac{TP}{TP + FP};$$

$$Recall = \frac{TP}{TP + FN}.$$

2.6 Розробка алгоритму функціонування інтелектуального модуля аналізу спаму текстів

Варто зазначити, що текстові повідомлення доцільно поділити на класи: спам, не спам, оскільки навчальна вибірка містить лише два класи спаму. Здійснити оцінку ефективності алгоритму можна тільки в тому випадку, якщо кількість класів спаму на вході і виході буде однаково. Тому було вирішено, що віднесення до нейтрального класу буде проводитися тільки в додатку, з урахуванням описаних вище умов.

Текст розбивається на уніграмми і біграми в разі заперечення, наприклад, «не люблю».

Оскільки в якості алгоритму був обраний наївний байесовский класифікатор, то для розпізнавання спаму необхідно враховувати наступні параметри: кількість позитивних і негативних повідомлень в корпусі, кількість позитивних і негативних

слів (словосполучень) в корпусі. Нижче представлений алгоритм: «Навчання класифікатора»(рисунок 2.5)


Рисунок 2.5 – Алгоритм навчання класифікатора

Навчання класифікатора передбачає заповнення словника, який включає в себе: n-грами, спаму n-грами, скільки дана n-грами зустрічалася в текстах кожного класу спаму. Для цього необхідно попередньо обробити кожен текст: видалити посилання, хештеги, імена користувачів, розділові знаки, привести слова до початкової формі і т.д. Другий етап - розбиття тексту на n-грами та запис їх в словник з урахуванням спаму, до якої належить текст. Третій етап - підрахунок кількості n-грам в тексті. Четвертий - збільшення лічильника кількості n-грам відповідно до кількості, підрахованому в третьому етапі з урахуванням класу спаму, до якого належить текст, і в тому ж класі спаму збільшення лічильника кількості текстів на одиницю. Загальний алгоритм функіонування класифікатора на рисунку 2.6.

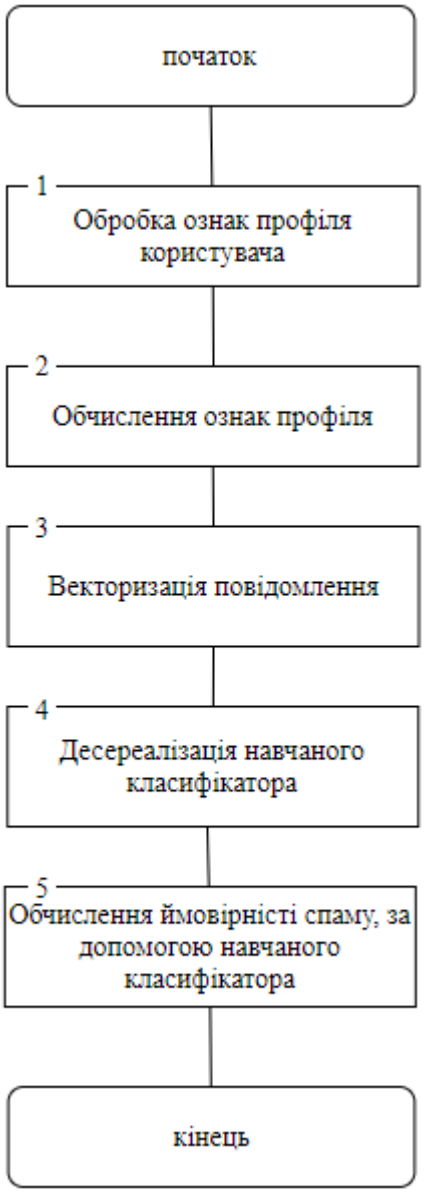


Рисунок 2.6 – Загальний алгоритм функціонування класифікатора.

## 2.7 Розробка структури інформаційної технології розпізнавання спам-повідомлень

На вхід подається текст повідомлення у вигляді строки. Після цього відбувається векторизація даного тексту на основі словника слів, який свою чергу був утворений на базі навчальної вибірки. Після цього йде обробка та оптимізація

даного вектора. Після оптимізації даний вектор потрапляє на вхід до баєсового класифікатора. Наступним є процес обробки вихідних даних та виведення оптимального результату.

Структура інформаційної технології розпізнавання спам-повідомлень зображена на рисунку 2.7

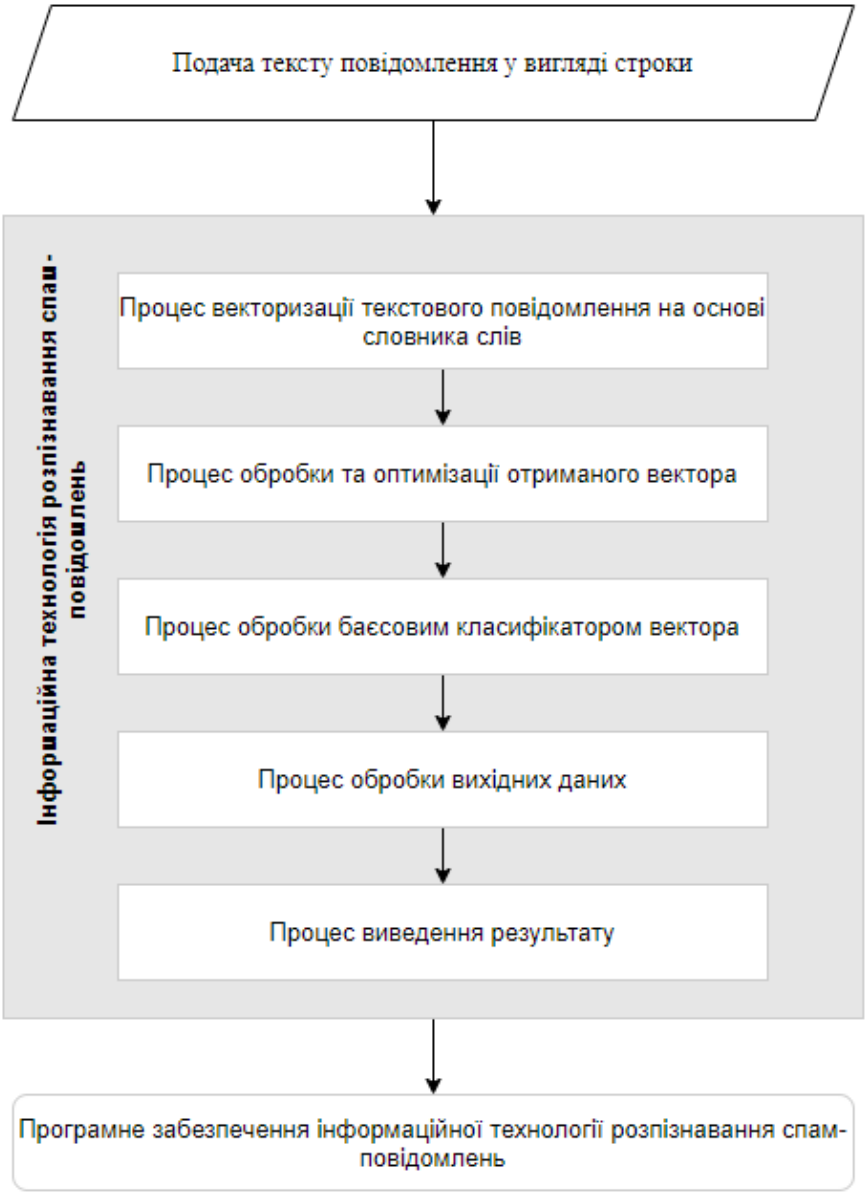


Рисунок 2.7 – Структура інформаційної технологій розпізнавання спам-повідомлень

## 2.8 Висновок

У даному розділі здійснено дослідження основних методів класифікації спаму зокрема: наївний баєсовий класифікатор, методи опорних векторів, метод k-найближчих сусідів, метод логістичної регресії, random forest, дерево ухваленння рішень. Надано перевагу баєсівому класифікатору через його точність.

Здійснено аналіз основних та допоміжних ознак, які вказують на спам. Здійснено аналіз векторної моделі мови що дозволило . Сформовано словник негативних слів.

Сформовано та розмічено навчальну вибірку. Здійснено попередню обробку навчальної вибірки з метою видалити символи і слова, які не впливають на результат розпізнавання, включаючи обробку смайликів, слів з максимальною частотою зустрічаваності.

Розроблено алгоритм функціонування інформаційної системи розпізнавання спам-повідомлень. Розглянуто способи поліпшення алгоритму розпізнавання спам-повідомлень на основі наївного байєсового класифікатора. Проаналізовано оцінки ефективності кількох варіантів реалізацій алгоритму, за результатам яких було запропоновано та розроблений модифікований алгоритм розпізнавання спам-повідомлень.

Розроблена структура інформаційної технології розпізнавання спам-повідомлень, яка складається з 5 етапів та дозволяє підвищити точність розпізнавання спам-повідомлень.

# 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ СПАМ-ПОВІДОМЛЕНЬ

## 3.1 Обґрунтування вибору мови програмування

При виборі засобів для розробки програмного проекту необхідно врахувати надзвичайно велику кількість різноманітних аспектів, найбільш важливим із яких ємова програмування, тому що вона в значній мірі визначає інші доступні засоби. Наприклад, для розробки користувацького графічного інтерфейсу розробникам необхідна сумісна з мовою програмування GUI-бібліотека, яка надає готові елементи інтерфейсу, такі, як кнопки и меню. При виборі мови програмування основними критеріями були продуктивність програмування, продуктивність роботи додатку та ефективність використання пам’яті. Важливим чинником для виборумови, в даному випадку є її універсальність.

Для розробки інтелектуальних модуля такого рівня існує три мови, які відповідають вимогам : Java, C# та Python.

Java – об'єктно-орієнтована мова програмування, випущена компанією SunMicrosystems у 1995 році як основний компонент платформи Java. Зараз мовою займається компанія Oracle, яка придбала Sun Microsystems у 2009 році. Синтаксисмови багато в чому походить від C та C++. У офіційній реалізації, Java програмикомпілюються у байткод, який при виконанні інтерпретується віртуальноюмашиною для конкретної платформи. Oracle надає компілятор Java та віртуальнумашину Java, які задовольняють специфікації Java Community Process, під ліцезієюGNU General Public License Мова значно запозичила синтаксис із C і C++. Зокрема, взято за основу об'єктну модель C++, проте її модифіковано. Усунуто можливість появи деяких конфліктних ситуацій, що могли виникнути через помилки програміста та полегшено сам процес розробки об'єктно-орієнтованих програм. Ряд дій, які в C/C++ повинні здійснювати програмісти, доручено

віртуальній машині. Передусім Java розроблялась як платформо-незалежна мова, тому вона має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм. За необхідності таких дій Java дозволяє викликати підпрограми, написані іншими мовами програмування. [29].

C# – об'єктно-орієнтована мова програмування з безпечною системоютипізації для платформи .NET. Розроблена Андерсом Гейлсбергом, СкотомВілтамутом та Пітером Гольде під егідою Microsoft Research (при фірмі Microsoft).

Синтаксис C# близький до C++ і Java. Мова має строгу статичну типізацію,підтримує поліморфізм, перевантаження операторів, вказівники на функції-членикласів, атрибути, події, властивості, винятки, коментарі у форматі XML.Перейнявши багато що від своїх попередників – мов C++, Delphi, Модула і Smalltalk– C#, спираючись на практику їхнього використання, виключає деякі моделі, щозарекомендували себе як проблематичні при розробці програмних систем,наприклад множинне спадкування класів (на відміну від C++).

JavaScript (часто просто JS) - це легка, що інтерпретується або JIT-компільована, об'єктно-орієнтована мова з функціями першого класу. Найбільш широке застосування знаходить як мова сценаріїв веб-сторінок, але також використовується і в інших програмних продуктах, наприклад, node.js або Apache CouchDB. JavaScript це прототипно-орієнтована, мультипарадигменна мова з динамічною типізацією, яка підтримує об'єктно-орієнтованийф, імперативний і декларативний (наприклад, функціональне програмування) стилі програмування.

Стандартом мови JavaScript є ECMAScript. Станом на 2012, всі сучасні браузери повністю підтримують ECMAScript 5.1. Більш ранні версії браузерів підтримують принаймні - ECMAScript 3. 17 червня 2015 року відбувся випуск шостої версії ECMAScript. Ця версія офіційно називається ECMAScript 2015 року, яку найчастіше називають ECMAScript 2015 або просто ES2015. З недавнього часу

стандарти ECMAScript випускаються щорічно. Ця документація відноситься до останньої версії чернетки, якої є ECMAScript 2018.

Mozilla надає дві реалізації JavaScript. Найперша реалізація JavaScript була створена Бренданом Ейхом (Brendan Eich) в компанії Netscape, і з тих пір оновлюється, щоб відповідати ECMA-262 Edition 5 і пізніших версій. Цей движок називається SpiderMonkey і реалізований на мові C / C ++. Движок Rhino створений Норрісом Бойдом (Norris Boyd) і реалізований на мові Java. Як і SpiderMonkey, Rhino відповідає ECMA-262 Edition 5.

Кілька оптимізацій, таких як TraceMonkey (Firefox 3.5), JägerMonkey (Firefox 4) і IonMonkey, додали в SpiderMonkey згодом. Робота завжди триває, щоб поліпшити продуктивність виконання JavaScript.

Не слід плутати JavaScript с мовою програмування Java. І "Java", і "JavaScript" є торговими марками або зареєстрованими торговими марками Oracle в США та інших країнах. Однак, в обох мов різний синтаксис, семантика і застосування.

Python – інтерпретована об'єктно-орієнтована мова програмування високогорівня з динамічною семантикою. Розроблена в 1990 році Гвідо ван Россумом.Структури даних високого рівня разом із динамічною семантикою та динамічнимзв'язуванням роблять її привабливою для швидкої розробки програм, а також якзасіб поєднання існуючих компонентів. Python підтримує модулі та пакети модулів,що сприяє модульності та повторному використанню коду. Інтерпретатор Python тастандартні бібліотеки доступні як у скомпільованій так і у вихідній формі на всіхосновних платформах. В мові програмування Python підтримується декількапарадигм програмування, зокрема: об'єктно-орієнтована, процедурна,функціональна та аспектно-орієнтована [30].

Основним недоліком мови C# перед Java та Python є її проприєтарна ліцензіята можливість використання усіх її можливостей в повному обсязі лише на OCWindows. Оскільки для розроблюваної система бажаною є повна підтримка

UNIX-подібних операційних систем, зокрема GNU/Linux. Тому реальним є вибір між Javata Python.

Швидкість розробки програмного забезпечення мовою Python є на порядок вищою у порівнянні із Java завдяки його динамічній типізації. Крім того Python єінтерпретованою мовою, що полегшує внесення змін та налаштування програмноїсистеми на етапі впровадження.

Проте основною перевагою при виборі мови Python було те, що для цієї мовипрограмування є найбільше нейромережевих бібліотек, такі як: Theano, Pylearn2,Deepnet, Caffe, Keras.

Тому основною мовою програмування обрано мову Python.

## 3.2 Обґрунтування вибору середовища розробки

При виборі середовища розробки потрібно оператися на операційну систему встановлену на вашому комп’ютері та можливості середовища розробки.

Notepad++ Notepad++ — текстовий редактор, призначений для програмістів і тих, кого не влаштовує скромна функціональність блокнота, що входить до складу Windows. Notepad++ базується на компоненті Scintilla (потужному компоненті для редагування), написаному на C++ з використанням тільки Win32 API і STL, що забезпечує максимальну швидкість роботи при мінімальному розмірі програми. Інтерфейс у Notepad++ — багатомовний. Серед особливостей програми — підсвічування синтаксису, підтримка великої кількості мов (C, C++, Java, XML, HTML, PHP, Java Script, ASCII, VB/VBS, SQL, CSS, Pascal, Perl і Python), багатомовна підтримка, робота з декількома документами. Підсвічування тексту і можливість згортання блоків, згідно з синтаксисом мови програмування WYSIWYG (друкуєш і отримуєш те, що бачиш на екрані) Режим підсвічування синтаксису, що налаштовується користувачем Авто-завершення слова, що набирається Одночасна робота з безліччю документів Підтримка регулярних

виразів для пошуку/заміни Повна підтримка перетягування фрагментів тексту Динамічна зміна вікон перегляду Автоматичне визначення стану файлу Підтримка великої кількості мов Замітки Плагіни Запис макросу і його виконання. На рисунку 3.1 можна побачити головне вікно редактору.

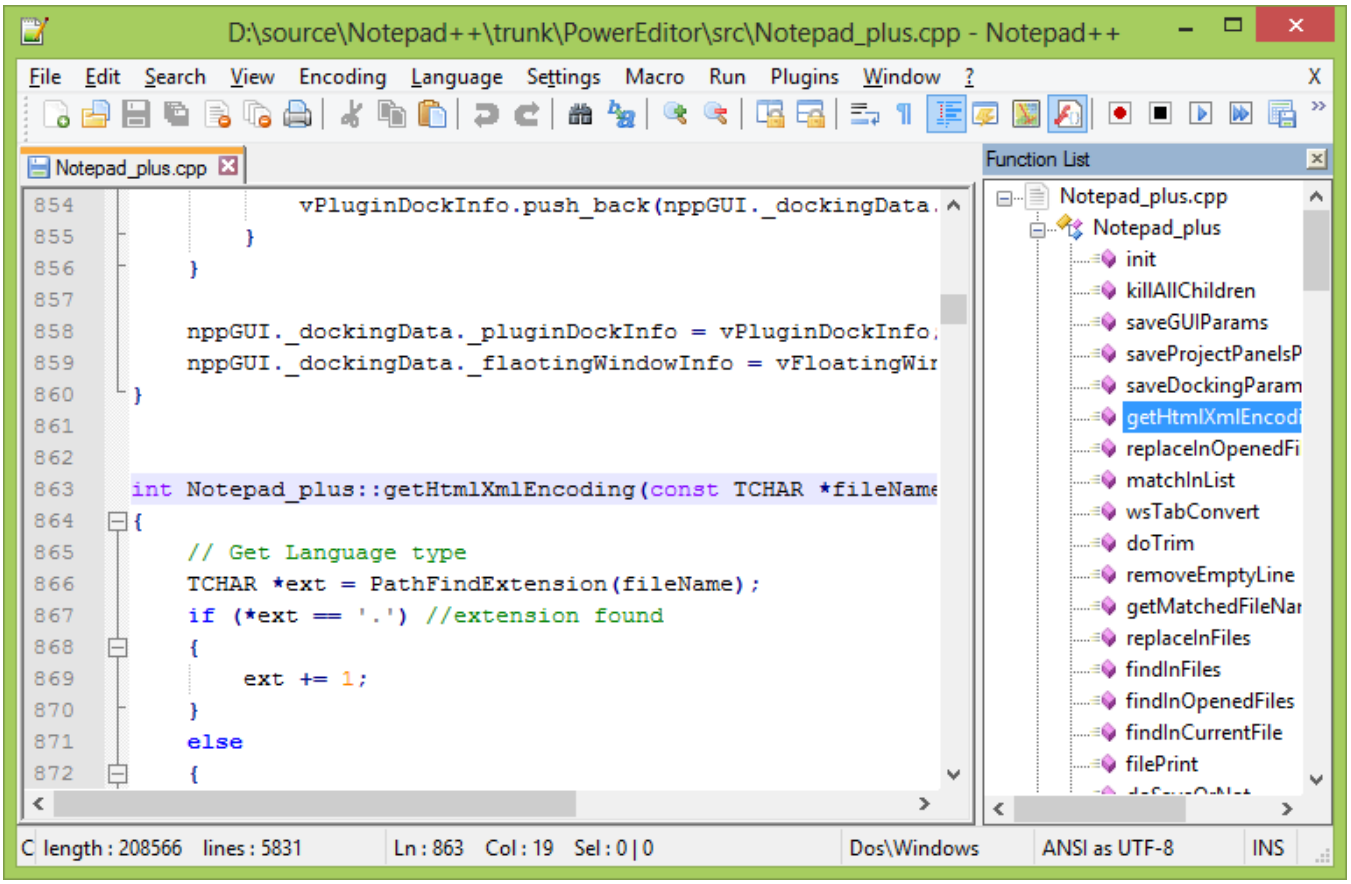


Рисунок 3.1 – Головне вікно редактору Notepad++

ST3 - швидкий кросплатформенний редактор початкових текстів програм. ST3 не є вільним чи відкритим програмним забезпеченням, але деякі його плагіни розповсюджуються з вільною ліцензією, розробляються і підтримуються спільнотою розробників. В даному середовищі можливе покрокове тестування системи та окремих модулів, що значно зменшує можливість виникнення критичної помилки на кінцевому етапі реалізації системи та дозволяє коригувати реалізацію,

покращуючи систему після кожного набору сценаріїв. Головне вікно даного середовища зображено на рисунку 3.2

Можливості:

- автозбереження;
- закладки в файлах;
- мінімапа: попередній перегляд повного початкового тексту;
- редагування на кількох панелях
- вбудована підтримка 27 мов програмування
- підсвітка синтаксису, що повністю налаштовується
- відповідність дужок, автодоповнення
- підтримка макросів та плагінів
- користувацькі гарячі клавіші
- функція швидкого переходу до певної ділянки коду
- надання швидкого доступу до команд

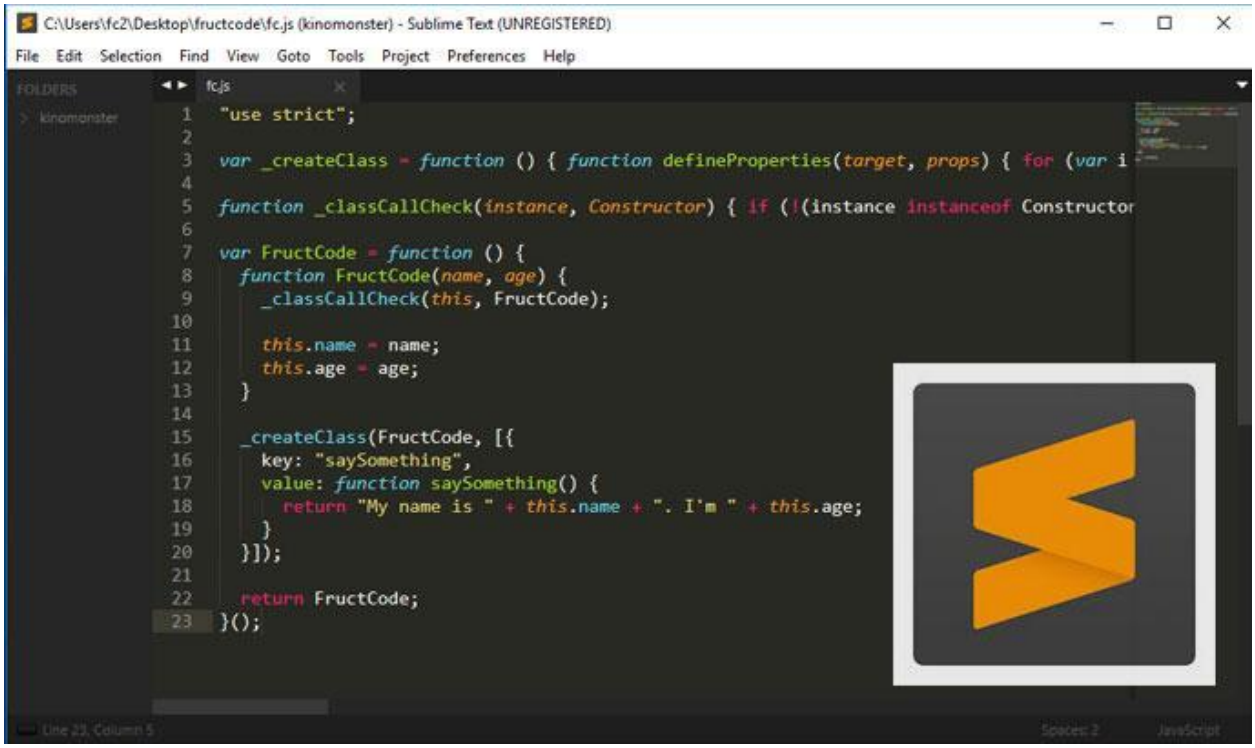


Рисунок 3.2 - Головне вікно середовища ST3

Microsoft Visual Studio — серія продуктів фірми Майкрософт, які включають інтегроване середовище розробки програмного забезпечення та ряд інших інструментальних засобів. Ці продукти дозволяють розробляти як консольні програми, так і програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-застосунки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, що підтримуються Microsoft Windows, Windows Mobile, Windows Phone, Windows CE, .NET Framework, .NET Compact Framework та Microsoft Silverlight. У Microsoft Visual Studio 2015 присутня низка нових функцій: розробки для Windows 8, Інтернету, SharePoint, мобільних пристроїв і хмарних служб, а також засобу управління життєвим циклом додатків, що дозволяють усунути перешкоди між співробітниками і скоротити цикли для безперервного нарощування цінності кінцевого результату. На рисунку 3.3 зображено середовище Visual studio.

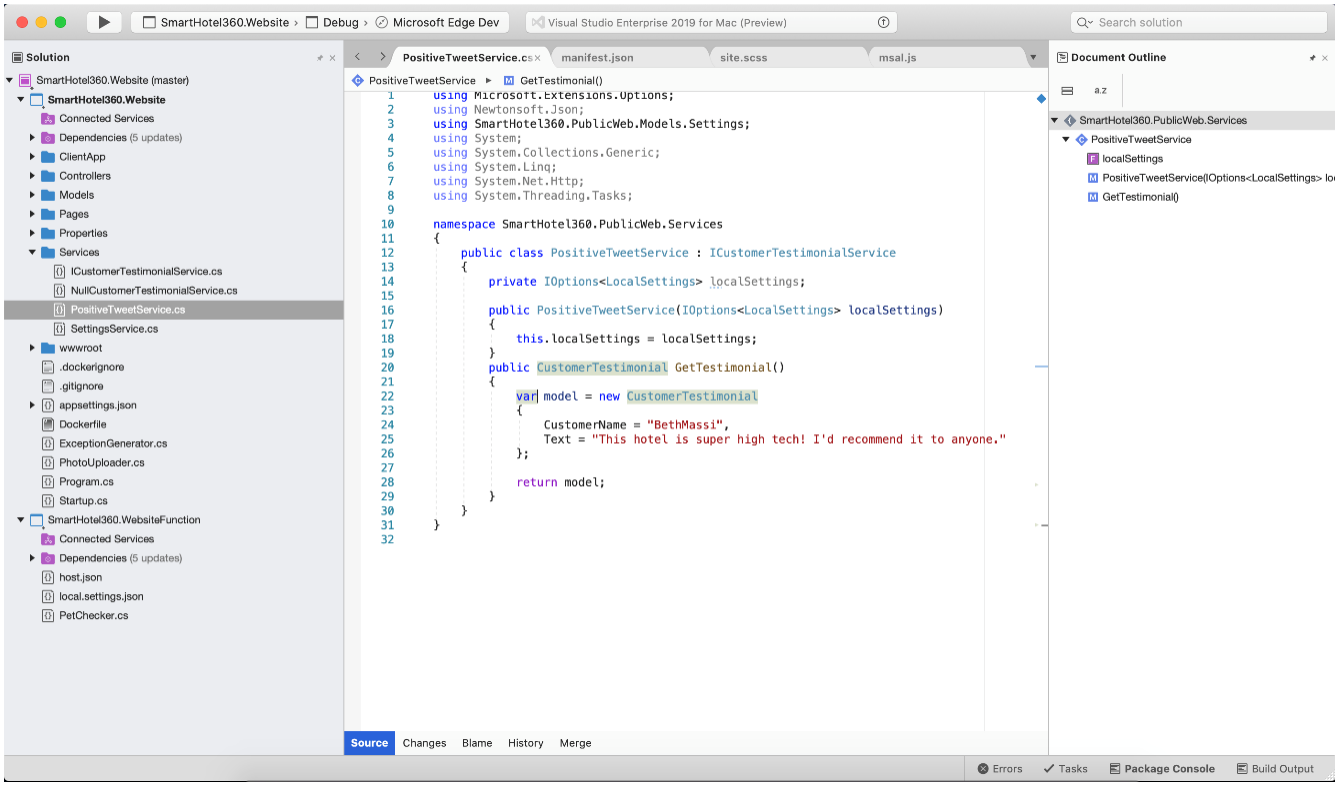


Рисунок 3.3 – Visual studio

Visual Studio Code — редактор для створення, редагування та зневадження сучасних веб-додатків, додатків для хмарних систем, додатків для . Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux і OS X.

Компанія Microsoft представила Visual Studio Code у квітні 2015 на конференції Build 2015 [30]. Це середовище розробки стало першим крос-платформовим продуктом у лінійці Visual Studio.

За основу для Visual Studio Code було взято готовий редактор проекту Atom, що розвивається компанією GitHub. Зокрема, Visual Studio Code є надбудовою над Atom Shell, що використовують браузерний рушій Chromium і Node.js. Примітно, що про використання напрацювань вільного проекту Atom на сайті Visual Studio Code і в прес-релізі і в офіційному блозі не згадується [9].

Редактор містить вбудований зневаджувач, інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки. Продукт підтримує розробку для платформ ASP.NET і Node.js, і позиціонується як легковагове рішення, що дозволяє обійтися без повного інтегрованого середовища розробки. Серед підтримуваних мов і технологій: JavaScript, C++, C#, TypeScript, jade, PHP, Python, XML, Batch, F#, DockerFile, Coffee Script, Java, HandleBars, R, Objective-C, PowerShell, Luna, Visual Basic, Markdown, JSON, HTML, CSS, LESS і SASS, Haxe.

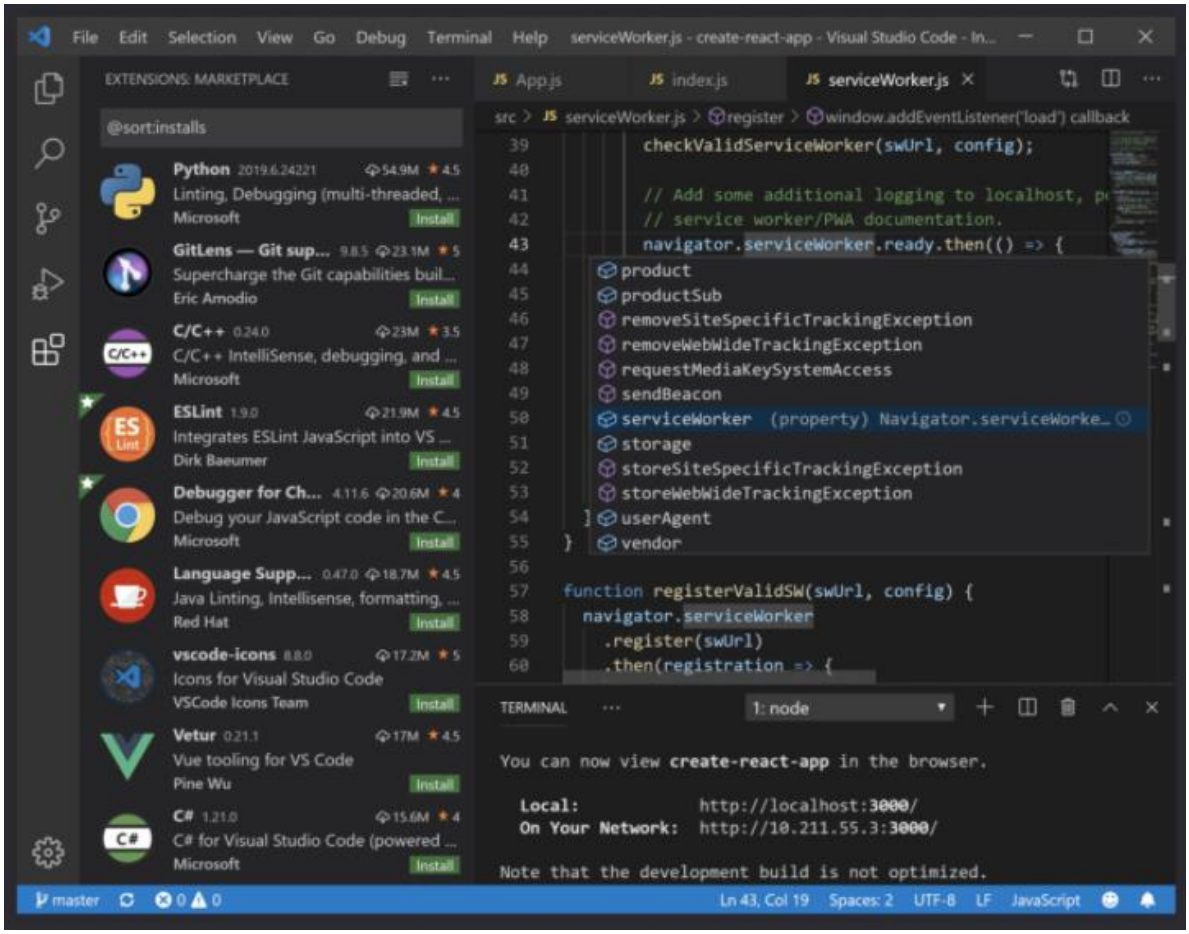


Рисунок 3.4 – Visual studio code

Порівняння середовищ розробки можна побачити на таблиці 3.1.

Таблиця 3.1 – Порівняння середовищ розробки

| | ST3 | VS code | VS 2019 | Notepad ++ |
|---|---|---|---|---|
| Інтелектуальна технологія | + | + | + | - |
| Підтримка авто запуску | + | + | + | + |
| Дебаг коду | +/- | +/- | + | +/- |
| Інструменти обробки | + | + | + | + |
| Автоконтроль | + | + | + | + |
| Інтеграція з іншими мовами | + | + | +/- | + |

Отже через свої переваги, в даному проекті буде використано VS Code, а саме через можливе покрокове тестування системи та окремих модулів, що значно зменшує можливість виникнення критичної помилки на кінцевому етапі реалізації.

## 3.3 Розробка UML-діаграм

На рисунку  зображено UML-діаграму класів розробленої системи. Розроблено класи ClassificatorBuilder, TrainDataBuilder, ClassificationResult, SpamCR, NonSpamCR, Package util.

Клас SpamClassifier відповідає за початкову активність програми. Викликає клас ClassificatorBuilder

Клас ClassificatorBuilder відповідає за навчання моделі і розпізнання тональності текстів.

Клас TrainDataBuilder відповідає за підготовку навчальної вибірки, розмітки тестового  і навчального набору.

Клас Package util відповідає за забезпечення проекту допоміжними бібліотеками.

Клас UserToInstanceMapper відповідає за трансформацію об’єктів користувачів

Клас userfeaturecomputor відповідає за визначення ознак користувача.

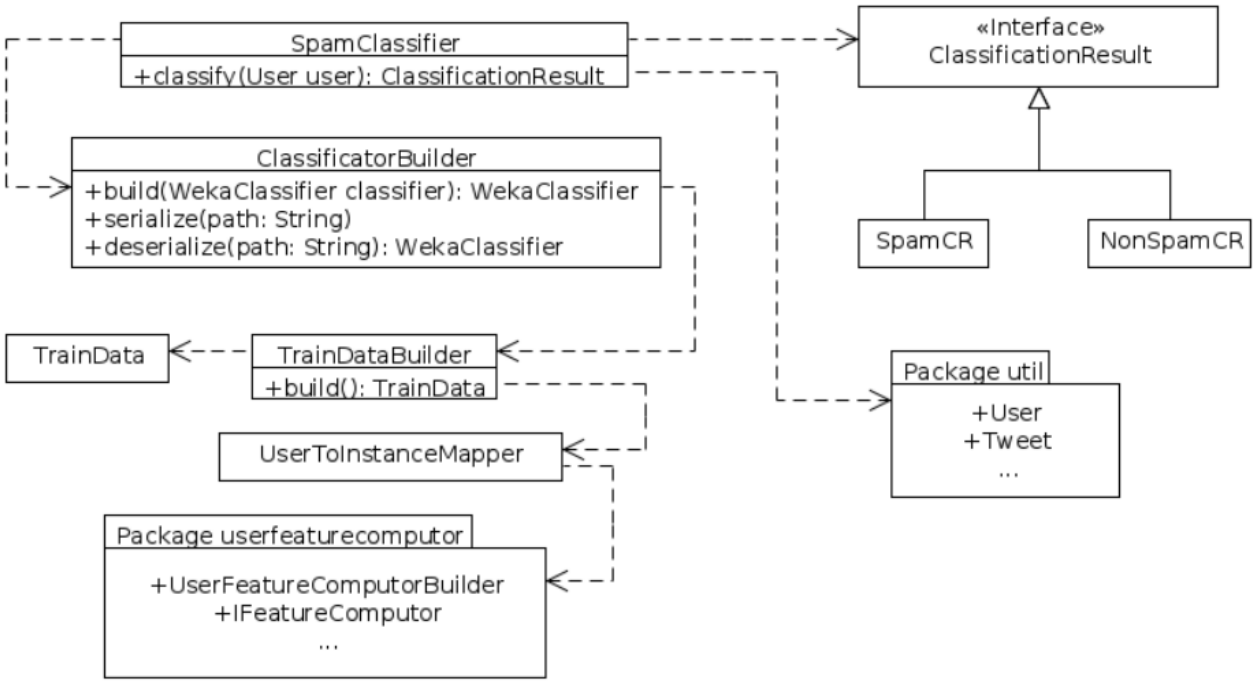


Рисунок 3.1 – UML-діаграма класів інтелектуального модуля розпізнавання спам-повідомлень

На рисунку 3.2 можна побачити загальну архітектуру системи

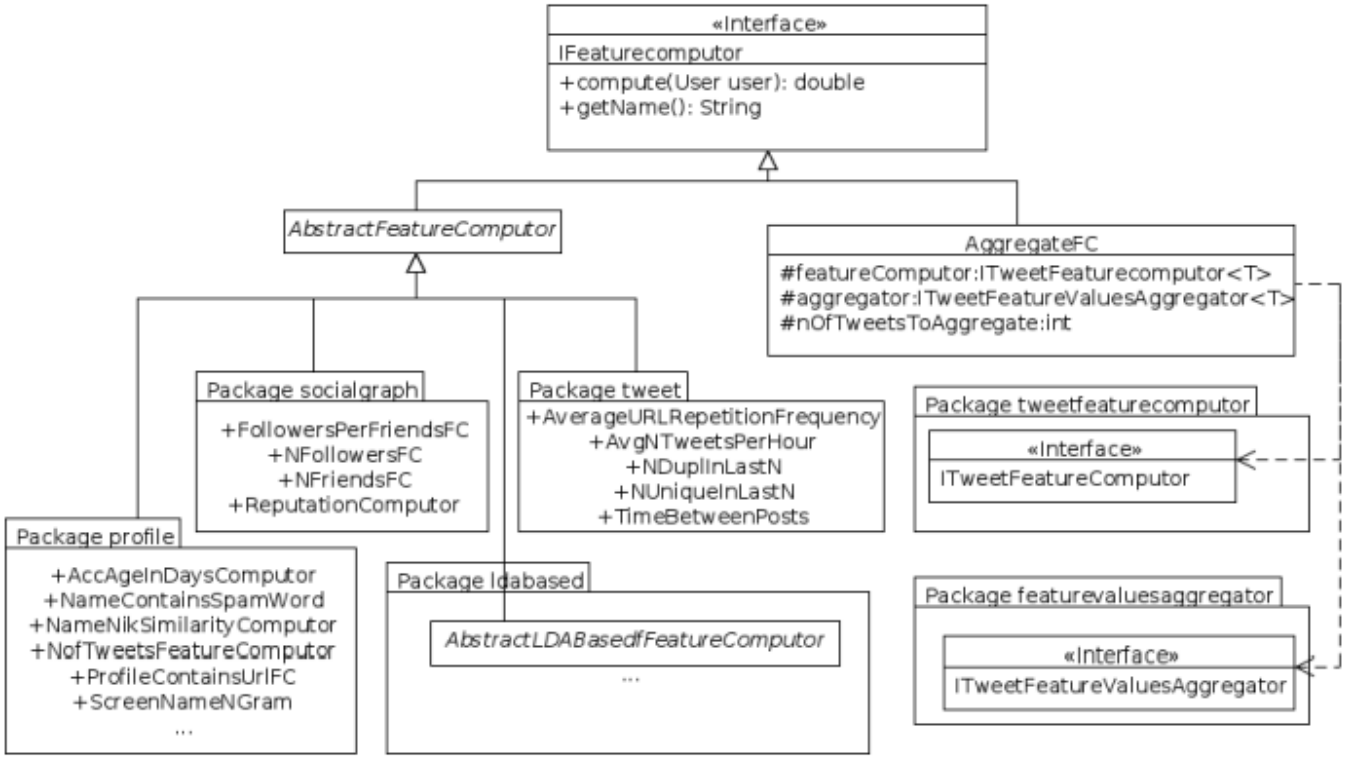


Рисунок 3.2 – Загальна архітектура системи розпізнавання спам-повідомлень

## 3.4 Тестування інформаційної технології розпізнавання спам-повідомлень

Для тестування алгоритмів визначення спам-повідомлень був використаний метод крос-валідації (або, по-іншому, перехресної перевірки). Процедура кросвалідації виконується наступним чином (рисунок 3.2):

1) Фіксується множина розбиттів навчальної вибірки зокрема на навчальну та тестову.
2) Для кожного розбиття відбувається навчання алгоритму на навчальній підвибірки і тестування на тестовій.
3) Результатом крос-валідації алгоритму є середні значення оцінок ефективності для тестових підвибірок.

Для 4900 не спам і 800 (рисунок 3.3) спам повідомлень ій було зроблено 4 підрозбить (5000 повідомлень - навчальна вибірка, 700 повідомлень - тестова). Для кожної групи розраховувалися оцінки ефективності, а потім вираховувались їх середні значення. Таким чином, вийшли усереднені оцінки ефективності. Результати усереднених оцінок ефективності для кожного з алгоритмів представлені нижче, в таблицях 3 і 4.

| | p' (Predicted) | n' (Predicted) |
|---|---|---|
| p (Actual) | True Positive | False Negative |
| n (Actual) | False Positive | True Negative |

Рисунок 3.2 – Таблиця критеріїв помилок

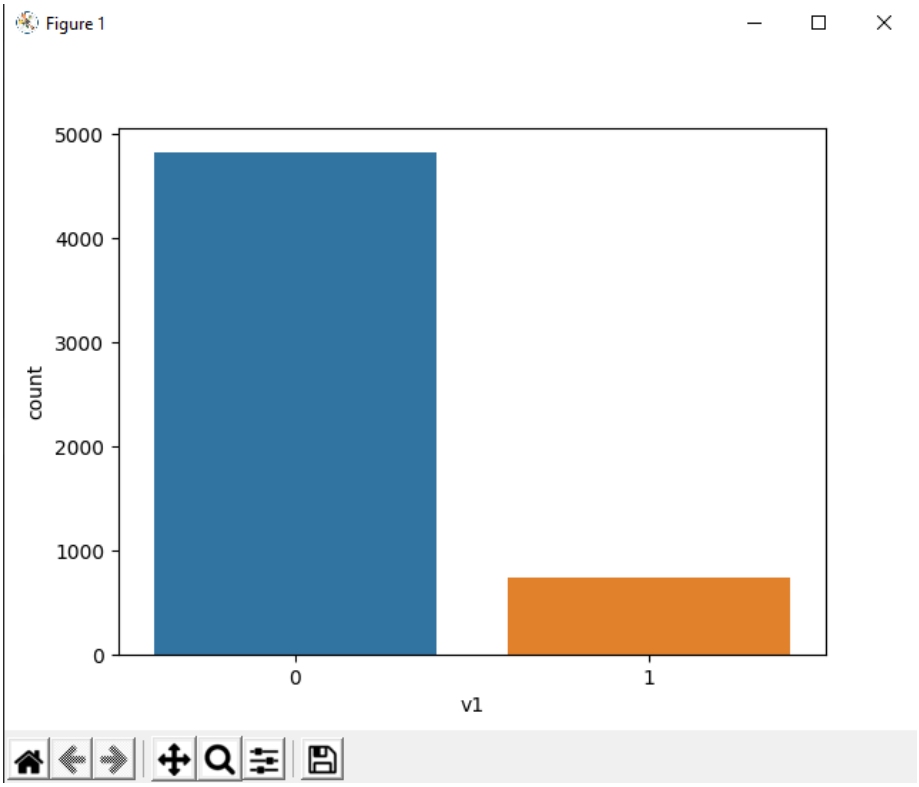


Рисунок 3.3 – Навчальна вибірка

Таблиця 3 – Наївний баєсовий класифікатор

| **Векторна модель** | **Уніграми** | | **Біграми** | |
|---|---|---|---|---|
| **Ваги векторів** | **Точність, %** | **Повнота, %** | **Точність, %** | **Повнота, %** |
| **Бінарні вектора** | 84,5 | 88,25 | 85,3 | 91,5 |
| **Частотні вектора** | 85,2 | 87,4 | **86,2** | **89,4** |

Наївний Байєсівський класифікатор показав дуже гарні результати, найбільш ефективною формою в плані точності виявилося поєднання биграмм і частотної функції зважування - точність 86%. Хоча, на розглянутій вибірці, для різних

параметрів класифікації, оцінка точності варіюється слабо - тому ефективним можна вважати будь-яке поєднання.

## 3.5 Висновок

Здійснено аналіз мов програмування, серед таких як: python, java, c#. Оскільки для реалізації інформаційної технології розпізнавання спам-повідомлень необхідно потужні інструменти створення та роботи з методами розпізнавання, було прийнято рішення використати мову програмування python, яка забезпечує широкий асортимент пакетів.

Здійснено аналіз середовищ розробок, таких як VS Code, ST3, Visual Studio, Notepad++. У ході аналізу було прийнято рішення використати VS Code, який є одним з найпопулярніших середовищ завдяки наявності великої кількості інструментів розробника, вбудованого терміналу, інструментів роботи з git і великої кількості інструментів, які можна отримати встановивши певні розширення. Ще однією суттєвою перевагою середовища є можливість її безкоштовного використання.

Здійснено проектування UML-діаграм класів. Після чого було здійснено програмну реалізацію засобу розпізнавання спам-повідомлень.

Протестовано розроблений додаток що був покладений в основу інформаційної технології розпізнавання спам-повідомлень. Під час тестування виявлено збільшення точності визначення спам-повідомлень приблизно на 2% – 4% порівняно з аналогами.

# 4 ЕКОНОМІЧНА ЧАСТИНА

## 4.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть Арсенюк І.Р. та Сілагін О.В.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою.

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 4.1.

Таблиця 4.1 – Результати оцінювання комерційного потенціалу розробки

| Критерії | Прізвище, ініціали, посада експерта | |
|---|---|---|
| | Арсенюк І.Р. | Сілагін О.В. |
| | Бали, виставлені експертами: | |
| 1 | 3 | 4 |
| 2 | 3 | 3 |
| 3 | 3 | 3 |
| 4 | 4 | 4 |
| 5 | 3 | 4 |
| 6 | 3 | 3 |
| 7 | 3 | 3 |
| 8 | 3 | 3 |
| 9 | 4 | 3 |
| 10 | 3 | 3 |
| 11 | 4 | 4 |
| 12 | 3 | 4 |
| Сума балів | $СБ_1 = 39$ | $СБ_2 = 41$ |
| Середньоарифметична сума балів $\overline{СБ}$ | $\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 40$ | |

Отже, з отриманих даних таблиці 4.1 видно, що нова розробка має високий рівень комерційного потенціалу.

4.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи.

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (4.1):

$$З_о = \frac{М}{Т_р} \cdot t, \quad (4.1)$$

де М – місячний посадовий оклад конкретного розробника;

$Т_р$ – кількість робочих днів у місяці, $Т_р$ = 22 дні;

t – число днів роботи розробника, t = 44 днів.

Розрахунки заробітних плат для керівника і програміста наведені в таблиці 4.2.

Таблиця 4.2 – Розрахунки основної заробітної плати

| Працівник | Оклад М, грн. | Оплата за робочий день, грн. | Число днів роботи, t | Витрати на оплату праці, грн. |
|---|---|---|---|---|
| Науковий керівник | 6000 | 272,72 | 5 | 1363 |
| Інженер-програміст | 5000 | 227,27 | 44 | 10000 |
| Всього: | | | | 11363 |

Розрахуємо додаткову заробітну плату:

$$З_{дод} = 0,1 \cdot 11363 = 1136.3 \text{ (грн.)}$$

Нарахування на заробітну плату операторів НЗП розраховується як 37,5…40% від суми їхньої основної та додаткової заробітної плати:

$$Н_{зп}= (З_о + З_р)\cdot \frac{\beta}{100}, \quad (4.2)$$

$$Н_{зп}= (11363+1136.3)\cdot \frac{22}{100} = 2749.84 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$А=\frac{Ц\cdot Н_а}{100}\cdot \frac{Т}{12}, \quad (4.3)$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Таблиця 4.3 – Розрахунок амортизаційних відрахувань

| Найменування програмного забезпечення | Балансова вартість, грн. | Норма амортизації, % | Термін використання, міс. | Величина амортизаційних відрахувань,грн |
|---|---|---|---|---|
| Персональний комп’ютер | 15000 | 25 | 2 | 625 |
| Всього: | | | | 625 |

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (4.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Таблиця 4.4 - Витрати на комплектуючі, що були використані для розробки ПЗ.

| Найменування матеріалу | Одиниці виміру | Ціна, грн. | Витрачено | Вартість витрачених матеріалів, грн. |
|---|---|---|---|---|
| Флешка | шт. | 200 | 1 | 200 |
| Пачка паперу | уп. | 130 | 1 | 130 |
| Ручка | шт. | 10 | 1 | 10 |
| Всього з урахуванням транспортних витрат | | | | 374 |

Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п , \quad (4.5)$$

де В – вартість 1кВт-години електроенергії (В=2 грн/кВт);

П – установлена потужність комп’ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп’ютера (Ф=224 год.);

$К_п$ – коефіцієнт використання потужності ($К_п < 1$, $К_п = 0{,}8$).

$$В_е = 2 \cdot 0{,}6 \cdot 224 \cdot 0{,}8 = 215 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін} = (1..3) \cdot (З_о + З_р). \quad (4.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 * (11363+1136.3) = 12499.8 \text{ (грн)}.$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

$$В = 11363+1136.3+2749.84 +625+374+215+12499.8= 28962.94 \text{ (грн.)}$$

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою:

$$В_{заг} = \frac{В_{ін}}{\alpha}, \quad (4.7)$$

де $\alpha$ – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{28962.94}{1} = 28962.94 .$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta}, \quad (4.8)$$

де $\beta$ – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{28962.94 .}{0,9} = 32181,04 \text{ (грн.)}$$

## 4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i \qquad (4.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

$\Delta N$ – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки витрати на виготовлення інформаційної технології зменшаться на 30 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 30 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 200 користувачів, протягом другого року – на 125 користувачів, протягом третього року – 100 користувачів. Реалізація інформаційної технології до впровадження результатів наукової розробки складала 550 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 300 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 20 \cdot 550 + (300 + 20) \cdot 200 = 75000 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = 20 \cdot 550 + (300 + 20) \cdot (200 + 125) = 115000 \text{ грн.}$$

Протягом третього року:

$$\Delta П_3 = 25 \cdot 500 + (400 + 25) \cdot (150 + 125 + 100) = 147000 \text{грн.}$$

## 4.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $E_{абс}$вкладених інвестицій розраховується за формулою:

$$E_{абс} = (ПП - PV), \quad (4.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 4.1.

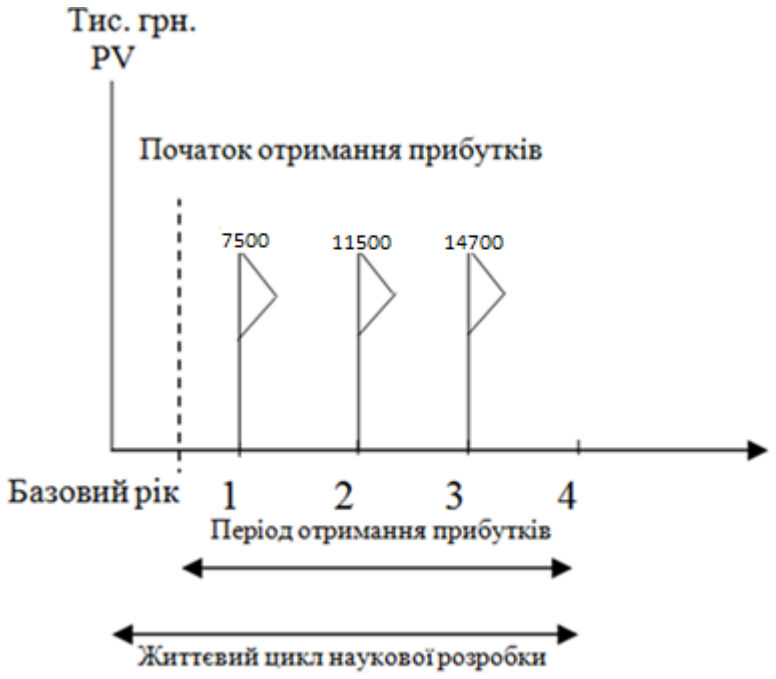

Рисунок 4.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t} \quad (4.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

$\tau$ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{32181,04}{(1+0,1)^0} + \frac{75000}{(1+0,1)^2} + \frac{115000}{(1+0,1)^3} + \frac{147000}{(1+0,1)^4} = 280967.97 \text{ (грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$Е_{абс} = 280967.97 - 32181,04 = 248786.92 \text{ грн.}$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою:

$$E_в = \sqrt[T]{1 + \frac{E_{абс}}{PV}} - 1, \quad (4.12)$$

де $E_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$T_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$E_в = \sqrt[3]{1 + \frac{248786.92}{32181,04}} - 1 = 1,05 \text{ або } 105\ \%.$$

Далі, розраховану величина $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2020 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0,2 + 0,1 = 0,3$$

Оскільки $E_в$ = 105% > $\tau_{мін}$ = 0,3 = 30%, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в},$$

$$T_{ок} = \frac{1}{1.05} = 0{,}95 \text{ року}.$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

# ВИСНОВКИ

У ході виконання магістерської кваліфікаційної роботи виконано аналіз предметної області розпізнавання спам-повідомлень. Проаналізовано основні способи поширення спаму серед яких можна відокремити: використання серверів, які помилково сконфігуровані; використання webmail-сервісів, які дозволяють анонімний доступ; використання комп'ютерів-зомбі тощо. Деякі спамери використовують відомі уразливості в програмному забезпеченні чи комп'ютерні віруси для того, щоб захопити керування великою кількістю комп'ютерів.

Розглянуто аналоги програмних засобів, зокрема Minecast, Baracuda Essentials. Mimecast забезпечує надійний багаторівневий захист від загроз для боротьби з вірусами та спамом, а також цілеспрямованими атаками електронної пошти, такими як фішинг. Barracuda Essentials пропонує ряд функцій захисту як від вхідних, так і від вихідних загроз для бізнесу. Ці програмні засоби мають такі основні недоліки: низька швидкість векторизації, неточність виявлення спаму.

Враховуючи недоліки існуючих програмних засобів здійснено постановку задачі, виконання якої дозволить підвищити точність визначення спаму.

Здійснено дослідження основних методів класифікації спаму зокрема: наївний баєсовий класифікатор, методи опорних векторів, метод k-найближчих сусідів, метод логістичної регресії, random forest, дерево ухваленння рішень. Надано перевагу баєсівому класифікатору через його точність. Запропоновано матиматичну модель для інформаційної технології розпізнавання спам-повідомлень.

Розроблено алгоритм функціонування інформаційної системи розпізнавання спам-повідомлень. Розглянуто способи поліпшення алгоритму розпізнавання спам-повідомлень на основі наївного байєсового класифікатора. Проаналізовано оцінки

ефективності кількох варіантів реалізацій алгоритму, за результатам яких було запропоновано та розроблений модифікований алгоритм розпізнавання спам-повідомлень.

Розроблена структура інформаційної технології розпізнавання спам-повідомлень, яка складається з 5 етапів та дозволяє підвищити точність розпізнавання спам-повідомлень.

Здійснено аналіз мов програмування, серед таких як: python, java, c#. Оскільки для реалізації інформаційної технології розпізнавання спам-повідомлень необхідно потужні інструменти створення та роботи з методами розпізнавання, було прийнято рішення використати мову програмування python, яка забезпечує широкий асортимент пакетів.

Здійснено аналіз середовищ розробок, таких як VS Code, ST3, Visual Studio, Notepad++. У ході аналізу було прийнято рішення використати VS Code.

Здійснено проектування UML-діаграм класів. Після чого було здійснено програмну реалізацію засобу розпізнавання спам-повідомлень.

Проведено тестування програмного засобу, яке показало, що удосконалена архітектура наївного баєсового класифікатора дала можливість покращити якість класифікації на в середньому на 2% – 4%. В результаті виконання даної кваліфікаційної роботи поставлені задачі були виконані в повній мірі. Отже, мета магістерської роботи досягнута.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бондарчук В. Ю., Арсенюк І. Р. Обробка та аналіз природної мови методами машинного навчання. // Тези доповідей XLIX науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії. – Вінниця: ВНТУ, 2020. Електронний ресурс (режим доступу https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2020/paper/view/9161/7764)

2. Свідоцтво про реєстрацію авторського права на твір Інформаційна технологія розпізнавання спам-повідомлень Бондарчук В. Ю., Арсенюк І. Р.

3. Wang . H. Don't follow me: Spam detection in twitter // Proceedings of the 2010 International Conference on Security and Cryptography (SECRYPT).— Vol. 13, N 3. — 1968. — P. 43–55.

4. Yang C., Harkreader R.C., Gu G. Die free or live hard? empirical evaluation and new design for fighting evolving twitter spammers / K. Fukushima // Biol. Cybern. — Vol. 36. —1980. — P. 193–202.

5. Schmidhuber J. Deep learning in neural networks: An overview / J. Schmidhuber // Neural networks. — Vol. 61. — 2015. — P. 85–117.

6. Hochreiter S. Long short-term memory / S. Hochreiter, J. Schmidhuber // Neural computation. — Vol. 9, N 8. — 1997. — P. 1735–1780.

7. Kelly R. Twitter Study – August 2009 [PDF] – [Електронний ресурс]. – Режим доступу: http://www.pearanalytics.com/wp-content/uploads/2012/12/Twitter-Study-August-2009.pdf

8. ABBY FineReader – [Електронний ресурс]. – Режим доступу: https://www.abbyy.com/uk/finereader/

9. Левенштейн В.И. Двоичные коды с исправлением выпадений, вставок и замещений символов // Доклады Академий Наук СССР. 1965. [163.] № 4. С. 845-848.

10. Srihari S. N. Offline Chinese Handwriting Recognition: A Survey./ Srihari S. N., Yang X., Ball G. R. –Frontiers of Computer Science in China, 1(2), 2007.

11. Zhang G. P.Neural networks for classification: a survey./ Zhang G.P. –IEEE Transactions on Systems, Man, and Cybernetics, 2000. –Part C: Applications and Reviews, 30(4). –p. 451–462.

12. Gomez Sanchez E. On-Line Character Analysis and Recognition with Fuzzy Neural Networks./ Gomez Sanchez E., Dimitriadis Y. A., Mas M.S.-R., Garcia P. S., Cano Izquierdo J. M., Coronado J. L.–Intelligent Automation and Soft Computing, Vol. 7, No. 3., 1998. –p. 161–162

13. Monge A., Elkan C. The Field Matching Problem: Algorithms and Applications // Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Vol. 7, No. 3., 1996. –p. 181–182

14. Карти Кохонена – [Електронний ресурс]. – Режим доступу: http://matlab.exponenta.ru/neuralnetwork/book2/20/kokhonen.php

15. Hammer, Barbara; Micheli, Alessio; Sperduti, Alessandro; Strickert, Marc (2004). Recursive self-organizing network models. Neural Networks 17: 1061–1085.

16. Manning C. D., Schütze H. Foundations of statistical natural language processing. MIT press., 7: 985—1004; 1994.

17. Happel, Bart and Murre, Jacob. The Design and Evolution of Modular Neural Network Architectures. Neural Networks, 7: 985—1004; 1994.

18. A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In NIPS, 2012.

19. Y. Bengio, P. Y. Simard, and P. Frasconi. Learning longterm dependencies with gradient descent is difficult. NN, 5(2):157–166, 1994.

20. A. Graves, S. Fern´andez, F. J. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In ICML, 2006.

21. A. Graves, S. Fern´andez, F. J. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In ICML, 2006.

22. M. Mukkamala and M. Hein. Variants of RMSProp and Adagrad with Logarithmic Regret Bounds. In Proceedings of the 3\ th International Conference on Machine Learning, pages 2545-2553, 2017.

23. V. Papavassiliou, T. Stafylakis, V. Katsouros, and G. Carayannis. Handwritten document image segmentation into text lines and words. Pattern Recognition, 43(l):369-377, 2010.

24. Що таке Python? [Електронний ресурс]. – Режим доступу: URL: http://www.plug.org.ua/documentation/about-python

25. . Сравнение Python с другими языками [Електронний ресурс]. – Режим доступу: URL: http://www.edu-main.narod.ru/Programming/stats/python

26. PyCharm — интеллектуальная Python IDE [Електронний ресурс]. – Режим доступу: URL: https://jetbrains.ru/products/pycharm

27. F. Chollet, “Keras” [Електронний ресурс]. – Режим доступу: URL: https://github.com/fchollet/keras, 2015

28. Tensorflow: Large-scale machine learning on heterogeneous distributed systems [Електронний ресурс]. – Режим доступу: URL: https://arxiv.org/abs/1603.04467

29. OpenCV [Електронний ресурс]. – Режим доступу: URL: https://opencv.org/about/

30. IAM Handwriting Database [Електронний ресурс]. – Режим доступу: URL: https://fki.tic.heia-fr.ch/databases/iam-handwriting-database