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

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

"На правах рукопису"
УДК 004.4

«До захисту допущено»
В.о. зав.кафедри
_____ Олександр КОВАЛЬ
“___”_____2022р.

Магістерська дисертація

За освітньою програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем і веб-технологій»
Спеціальності 121 Інженерія програмного забезпечення
на тему: Система автоматичного визначення фейкових новин в сучасному інформаційному просторі

Виконав студент 2 курсу, групи ТВ-11мп
Кузьменчук Дмитро Олександрович _____
(прізвище, ім’я, по батькові) (підпис)
Науковий керівник ктн., доц. Ходаковський О. В. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Київ – 2022

Національний технічний університет України
«Київський політехнічний інститут ім. Ігоря Сікорського»

Навчально-науковий інститут атомної та теплової енергетики
Кафедра інженерії програмного забезпечення в енергетиці
Рівень вищої освіти другий, магістерський
За освітньою програмою «Інженерія програмного забезпечення інтелектуальних кібер-фізичних систем і веб-технологій»
Спеціальності 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
в.о. зав. кафедри
_____ Олександр КОВАЛЬ
(підпис)
«____» _____ 2022р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Кузьменчуку Дмитру Олександровичу
(прізвище, ім'я, по батькові)
1. Тема дисертації: Система автоматичного визначення фейкових новин в сучасному інформаційному просторі
Науковий керівник ктн., доц. Ходаковський Олексій Володимирович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від "07" листопада 2022 року №4067-с
2. Строк подання студентом дисертації 05 грудня 2022 року
3. Вихідні дані до роботи мови програмування Python та Java Script, комп'ютерні технології в області розпізнавання мов для вирішення задачі розпізнавання фейків.
4. Перелік питань, які потрібно розробити Провести аналіз існуючих систем в сфері визначення фейкових новин. Розробити дизайн користувацького інтерфейсу. Провести аналіз існуючих методів для визначення фейків та розробити на їх основі свій власний. Обрати стек технологій який дозволить створити модель для класифікації та розгорнути систему в хмарному сервісі. Розробити діаграму прецедентів. Розробити всі необхідні модулі необхідні для функціонування системи. Поєднати модулі разом та провести тестування системи.
5. Орієнтований перелік ілюстративного матеріалу інтерфейс головної сторінки системи, інтерфейс сторінки з документацією про API, діаграма прецедентів, графіки що демонструють принципи роботи різних методів машинного навчання, зображення з конфігурацією різних елементів інфраструктури системи.
6. Орієнтований перелік публікацій _____
7. Дата видачі завдання «01» жовтня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	строки виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	01.10.2021	виконано
2	Дослідження предметної області	01.10.2021 - 01.11.2021	виконано
3	Постановка вимог до проєктування системи	01.11.2021 - 15.11.2021	виконано
4	Дослідження існуючих рішень	15.11.2021 - 01.12.2021	виконано
5	Підготовка публікацій	01.12.2021 - 09.12.2021	виконано
6	Розробка програмного продукту	05.03.2022 - 07.10.2022	виконано
7	Тестування	07.10.2022 - 11.11.2022	виконано
8	Захист програмного продукту	17.10-2022 – 21.10.2022	виконано
9	Підготовка магістерської дисертації	22.10.2022 – 20.11.2022	виконано
10	Передзахист	21.11.2022 - 25.11.2022	виконано
11	Захист	19.12.2022 – 23.12.2022	виконано

Студент _____ Кузьменчук Д. О.
(підпис) (прізвище та ініціали)

Науковий керівник _____ Ходаковський О. В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Актуальність теми. Інформаційна безпека є дуже важливим аспектом сучасного життя при чому інформаційна безпека включає в себе не лише захист персональних даних від різного роду шахраїв, а й перевірку на достовірність інформації яку ми споживаємо чому і присвячена дана магістерська дисертація. Захист від фейкової інформації є особливо важливим оскільки вона може бути дуже шкідливою як для окремих людей так і для великих мас населення та спричинювати різні негативні явища.

Основними цілями застосування фейкової інформації є маніпуляція суспільною думкою для здобуття переваги у виборах, поширення пропаганди для контролю населення, дискредитація окремих осіб, компаній чи державних урядів, розповсюдження паніки серед людей та створення напруги в суспільстві для дестабілізації становища в країні. З переліченого вище стає зрозумілим наскільки небезпечним та деструктивним може бу вплив фейкової інформації, а найяскравішим прикладом її використання є виправдання повномасштабної війни Росії проти України яка почалась 24 лютого цього року та гібридної війни яка тривала попередні 8 років починаючи з 2014 року.

Оскільки фейкова інформація є дуже потужним інструментом для досягнення своїх цілей то скоріш за все її ніколи не перестануть використовувати і через це буде існувати постійна боротьба між двома сторонами: тою яка розповсюджує фейкову інформацію і тою що цьому протистоїть. Саме тому тема боротьби з фейковою інформацією завжди була, є та буде дуже актуальною темою.

Мета роботи: дослідження існуючих методів та програмних засобів які використовуються при роботі з людськими мовами для визначення намірів та сенсу який було вкладено в текст чи аудіо повідомлення та створення на їх основі власного методу. На основі отриманих теоретичних знань необхідно визначити переваги та недоліки існуючих методів, вибрати ті з них які найкраще підходять та за можливості вдосконалити їх. Після чого на основі отриманих алгоритмів слід

спроектувати та розробити систему яку з легкістю зможуть використовувати як звичайні користувачі так і розробники для інтеграції з іншими системами.

Методи дослідження. Для виконання поставлених задач використовувались наступні методи емпіричного дослідження:

— Експеримент під час якого випробовувались різні методи машинного навчання для вирішення задачі визначення фейкових новин.

— Вимірювання за допомогою якого визначалась точність роботи алгоритмів при виконання поставленої задачі.

Також поряд з емпіричними методами дослідження використовувались загальнологічні методи дослідження такі як:

— Аналіз який використовувався для виявлення впливу змін в налаштуваннях методів штучного інтелекту на отримуваний результат.

— Порівняння яке використовувалось для визначення найбільш результативного методу з поміж інших.

Практичне значення отриманих результатів. Розробка системи для автоматизованого визначення фейкових новин яку в подальшому можна буде інтегрувати з іншими системами для боротьби з розповсюдженням неправдивих новин у них. Таким чином розроблений програмний продукт повинен сприяти збільшенню рівня довіри до інформації в системах де його було використано. З іншого боку створені та удосконалені в ході виконання роботи алгоритми повинні давати розвиток обробці природніх мов, а саме розділу машинного навчання під назвою NLP (Natural language processing).

Особистий внесок магістранта. Дослідження застосування різних методів машинного навчання для виконання задачі з виявлення фейкових новин та аналіз отриманих результатів.

Структура та обсяг роботи. Магістерська дисертація викладена на 80 сторінках і містить 43 рисунків, 1 додаток та має 17 бібліографічних посилань.

Ключові слова: штучний інтелект, машинне навчання, NLP, датасет, API.

ABSTRACT

Relevance of the topic. Information security is a very important aspect of modern life, which includes not only the protection of personal data from various types of fraudsters, but also the verification of the reliability of the information we consume, which is what this master's thesis is dedicated to. Protection against fake information is particularly important because it can be very harmful both to individuals and to large masses of the population and cause various negative phenomena.

The main goals of using fake information are to manipulate public opinion to gain an advantage in elections, spread propaganda to control the population, discredit individuals, companies or state governments, spread panic among people and create tension in society to destabilize the situation in the country. From the stated above, it becomes clear how dangerous and destructive the influence of fake information can be, and the most striking example of its using is the justification of Russia's full-scale war against Ukraine, which began on February 24 of this year, and the hybrid war that lasted for the previous 8 years, starting in 2014.

Since fake information is a very powerful tool for achieving one's goals, it will most likely never stop being used, and because of this, there will be a constant struggle between two sides: the one that spreads fake information and the one that opposes it. That is why the topic of combating fake information has always been, is and will be a very relevant topic.

The goal of the work. The purpose of this work is to research the existing methods and software tools used when working with human languages to determine the intentions and meaning that was embedded in the text or audio message and to create an own method based on them. On the basis of the obtained theoretical knowledge, it is necessary to determine the advantages and disadvantages of existing methods, to choose the ones that are most suitable and, if possible, to improve them. After that, on the basis of the obtained algorithms, a system should be designed and developed that can be easily used by both ordinary users and developers for integration with other systems.

Research methods. The following methods of empirical research were used to fulfill the tasks:

— An experiment during which various machine learning methods were tested to solve the problem of identifying fake news.

— The measurement used to determine the accuracy of the algorithms when performing the given task.

Also, along with empirical research methods, general logical research methods such as:

— Analysis that was used to identify the impact of changes in the settings of artificial intelligence methods on the obtained result.

— A comparison used to determine the most effective method among others.

Practical significance of the obtained results. Development of a system for automated detection of fake news, which can be integrated with other systems to combat the spread of false news in them. The software product developed in this way should contribute to increasing the level of trust in information in the systems where it was used. On the other hand, the algorithms created and improved in the course of the work should give development to the processing of natural languages, namely the section of machine learning called NLP (Natural language processing).

Personal contribution of the master's student. A personal contribution to the implementation of this master's thesis is the study of the application of various methods of machine learning to perform the task of detecting fake news and the analysis of the obtained results.

Structure and scope of work. The master's thesis is laid out on 80 pages and contains 43 figures, 1 appendix and has 17 bibliographic references.

Keywords: artificial intelligence, machine learning, NLP, dataset, API.

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ 9
ВСТУП 10
1 ПОСТАНОВКА ЗАВДАННЯ ПОБУДОВИ СИСТЕМИ АВТОМАТИЧНОГО ВИЗНАЧЕННЯ ФЕЙКОВИХ НОВИН В СУЧАСНОМУ ІНФОРМАЦІЙНОМУ ПРОСТОРІ 12
1.1 Ключові завдання при розробці серверної частини застосунку 12
1.2 Ключові завдання при створенні моделі для класифікації фейкових новин 13
1.3 Ключові завдання при розробці користувацького інтерфейсу 14
1.4 Вимоги до готового програмного продукту 15
2 ДОСЛІДЖЕННЯ ІСНУЮЧИХ МЕТОДІВ ДЛЯ ВИЗНАЧЕННЯ ФЕЙКОВИХ НОВИН 16
2.1 Дослідження предметної області 16
2.2 Обґрунтування обраних програмних засобів 18
3 АПАРАТ ВИРІШЕННЯ ДЛЯ ПОСТАВЛЕНОЇ ЗАДАЧІ 20
3.1 Підготовка даних для навчання моделі 20
3.2 Застосування методу Random forest 24
3.3 Застосування алгоритму Decision tree 27
3.4 Застосування методу опорних векторів 31
3.5 Застосування логістичної регресія 34
3.6 Розгортання інфраструктури системи 39
4 ОПИС ПРОГРАМНОЇ СИСТЕМИ 50
4.1 Опис інтерфейсу головної сторінки 50
4.2 Опис Інтерфейсу сторінки про API системи 52
5 РОЗРОБКА СТАРТАПУ ПРОЄКТА 55
5.1 Етапи реалізації стартап проекту 55
5.2 Аналіз поточної ситуації на ринку 56
5.3 Аналіз цільової аудиторії стартапу 58

5.4 Аналіз можливих ризиків 59
5.5 Прогнозування витрат на розробку програмного продукту 61
5.6 Прогнозування витрат для підтримки інфраструктури 65
5.7 Запуск маркетингової кампанії 72
5.8 Обґрунтування рівня рентабельності стартапу 75
ВИСНОВКИ 78
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 79
ДОДАТОК А 81

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Датасет — це набір інформації пов’язаної між собою однією спільною темою, що може бути розділена на окремі елементи та сприймається комп’ютером як єдине ціле.

Штучний інтелект — це комплекс теоретичних знань та практичних підходів, що дозволяють створювати системи здатні виконувати задачі які вимагають використання людського інтелекту. Прикладами таких систем є: розпізнавання об’єктів на зображеннях, переклад тексту на різні мови, допомога в прийнятті рішень та розпізнавання людського голосу.

Машинне навчання — це процес розробки систем, що здатні навчатись та адаптуватись використовуючи вхідний набір даних, а також статистичні та спеціалізовані алгоритми які дозволяють побудувати зв’язки між вхідними даними.

NLP (natural language processing) — це розділ штучного інтелекту діяльність якого направлена на побудову алгоритмів для аналізу природних мов, а точніше написаного тексту та аудіо даних.

Фейкова новина — це інформація що містить неправдиву або викривлену інформацію стосовно певних подій чи явищ та зазвичай створюється навмисно для досягнення певних цілей.

AWS — Amazon Web Services.

EC2 — Elastic Compute Cloud.

ВСТУП

В останні десятиліття з розвитком технологій доступ до інтернету став широкодоступним, а разом з цим з’явилися мільйони інформаційних систем серед яких є соціальні мережі такі як Facebook, Instagram, Twitter та багато інших які налічують мільйони та мільярди користувачів.

В кожній з таких інформаційних систем користувачі кожного дня обмінюються великими обсягами інформації. Незважаючи на те, що сьогодні є безліч можливостей перевірити достовірність інформації більшість людей цього не роблять через те, що інформації забагато або бракує на це часу чи просто лінь.

Саме тому цим вдало користуються як окремі люди так і цілі організації які за допомогою фейкових новин поширюють неправдиву інформації на окремі групи людей, а інколи навіть на цілі країни з метою маніпулювання суспільною думкою для досягнення певних корисних цілей.

Яскравим прикладом таких фейкових новин є інформація яку активно розповсюджує Російська Федерація. Починаючи з 2014 року вона витрачала мільярди доларів для того щоб виправдати свою агресію та приховати військові злочини стосовно України. При цьому використовується як викривлення фактів так і цілком вигадані історії як про секретні біолабораторії чи бази НАТО. Нажаль через величезні об’єми таких новин Росія має певні успіхи в інформаційній війні впливаючи на людей по всьому світу.

Через це розробка системи для автоматизованого виявлення фейкових новин є дуже актуальною темою. За допомогою такої системи можна було б аналізувати великі об’єми інформації при цьому витрачаючи значно менше коштів і часу порівняно з тим якби цим займались реальні люди. Після виявлення такої інформації її можна видаляти або приховувати, а стосовно користувачів які розповсюджували її вживати певні заходи.

Таким чином з використанням даної системи було б можливим контролювати розповсюдження фейкових новин в різних інформаційних системах запобігаючи

маніпуляціям соціальною думкою, а також іншим негативним наслідкам які з цього випливають.

У першому розділі розглянуто постановку задачі для побудови системи для автоматичного визначення фейкових новин в сучасному інформаційному просторі та визначено найважливіші цілі.

У другому розділі розглянуто особливості предметної області розроблюваної системи.

У третьому розділі викладено можливості інструментів обраних для реалізації системи, а також причини їх вибору.

У четвертому розділі розкривається архітектура побудованої системи а також способи взаємодії з нею.

У п’ятому розділі проводиться аналіз можливостей створення стартапу за темою магістерської дисертації.

1 ПОСТАНОВКА ЗАВДАННЯ ПОБУДОВИ СИСТЕМИ АВТОМАТИЧНОГО ВИЗНАЧЕННЯ ФЕЙКОВИХ НОВИН В СУЧАСНОМУ ІНФОРМАЦІЙНОМУ ПРОСТОРІ

При виконанні даної магістерської роботи необхідно було визначити ключові завдання за наступними напрямами:

— розробка серверної частини застосунку;

— створення моделі для класифікації;

— розробка користувацького інтерфейсу.

Також необхідно визначити якими характеристиками повинен володіти готовий програмний продукт.

1.1 Ключові завдання при розробці серверної частини застосунку

Перед початком розробки серверної частини застосунку необхідно обрати технології які будуть відповідати наступним вимогам:

— Мати активну підтримку зі сторони розробників та достатньо широку спільноту користувачів для того щоб була можливість отримати консультацію з проблемних питань.

— Мати достатній набір вбудованих функцій та бібліотек щоб покрити всі основні проблеми та завдання, що виникають при розробці будь-якого веб-застосунку.

— Забезпечувати високий рівень безпеки персональних даних користувачів даних користувачів.

— Забезпечувати високу продуктивність роботи при високому навантаженні на систему.

При розробці програмного продукту були поставлені наступні цілі:

— Дослідити існуючі системи з аналогічним призначенням та визначити їх слабкі та сильні сторони.

— Визначити перелік функцій яких немає в системах конкурентів, а також функції які можливо покращити.

— Розробити архітектуру системи яку можна було легко створити та при необхідності легко модифікувати.

— Визначити типи даних які надходять в систему та які користувач отримує на виході.

— розгорнути систему в хмарному сервісі щоб нею могли скористатись реальні користувачі.

— провести аналіз коду і виправити можливі недоліки.

— протестувати готову систему за допомогою реальних користувачів на базі практики.

1.2 Ключові завдання при створенні моделі для класифікації фейкових новин

При розробці моделі для класифікації фейкових новин було поставлено наступні цілі:

— Проаналізувати доступну на базі практики літературу, що відноситься до обробки природних мов засобами штучного інтелекту, а також праці інших авторів які присвячені виявленню фейкових новин.

— Зібрати з відкритих джерел датасет який буде містити достатнє число правдивих та фейкових новин для подальшого тренування моделей.

— Провести підготовку даних з датасету для отримання кращої точності отримуваних при тренуванні моделей.

— Провести аналіз існуючих методів, визначити їх сильні та слабкі сторони та на їх основі розробити свій власний метод.

— Провести тренування моделей використовуючи різні методи та підходи.

— Обрати методи для оцінки точності роботи отримуваних в ході тренування моделей.

— Проаналізувати отримані результати та обрати модель, що показала найвищу точність роботи.

1.3 Ключові завдання при розробці користувацького інтерфейсу

Інтерфейс користувача розробленої системи повинен відповідати наступним критеріям:

— Мати можливість зміни мови інтерфейсу користувача на англійську та українську.

— Мати можливість перевірки однієї новини, що була введена в текстовому форматі.

— Мати можливість перевірки набору новин шляхом завантаження файлу в форматі csv з подальшим отриманням результату в такому ж форматі.

— Мати детальну інструкцію щодо того як взаємодіяти з системою за допомогою API.

— Мати зручну та зрозумілу навігацію між різними сторінками застосунку.

— Забезпечувати швидке завантаження сторінок для створення позитивного досвіду користування застосунком у користувача.

1.4 Вимоги до готового програмного продукту

Створений програмний продукт повинен відповідати наступному ряду вимог та характеристик:

— запускатися на будь-якому пристрої на якому встановлено браузер з підтримкою HTML, CSS та JavaScript;

— мати зручний та інтуїтивно зрозумілий інтерфейс;

— мати підказки для користувача які говорять про призначення тих чи інших елементів інтерфейсу;

— динамічно реагувати на дії користувача;

— мати низький час відповіді від серверу для створення позитивного досвіду користування системою;

— мати високий рівень точності при класифікації фейкових новин.

Висновки до розділу 1

Задача визначення фейкових новин в сучасних інформаційних системах полягає в класифікації текстових повідомлень за мірою їх достовірності. Створення автоматизованої системи для виконання цієї задачі є дуже важливим завданням оскільки це дозволить зменшити швидкість розповсюдження фейків таким інформаційним системам як соціальні мережі та месенджери.

2 ДОСЛІДЖЕННЯ ІСНУЮЧИХ МЕТОДІВ ДЛЯ ВИЗНАЧЕННЯ ФЕЙКОВИХ НОВИН

Вибір правильних методів та програмних засобів є запорукою для реалізації успішного програмного продукту. В ході виконання даної магістерської дисертації досліджується ефективність таких методів машинного навчання як: метод опорних векторів, метод Random forest, дерева прийняття рішень та логістична регресія. Також проводиться аналіз існуючих програмних засобів які дозволять якнайкраще реалізувати всі поставлені цілі.

2.1 Дослідження предметної області

Предметною областю даної магістерської дисертації є розділ штучного інтелекту під назвою Natural language processing (NLP) [1-5] який відповідає за аналіз тексту та людської мови.

NLP в ході своєї роботи використовує такі прийоми як комп’ютерна лінгвістика, моделювання людської мови, статистичні моделі, машинне, а також глибоке навчання за допомогою нейронних мереж. Разом всі ці технології дають змогу комп’ютеру обробляти текстові дані, а також людський голос в аудіо форматі та «розуміти» який сенс, почуття та наміри було вкладено.

Прикладом систем які використовують цю технологію можна назвати програми які перекладають текст з однієї мови на іншу, елементи розумного дому які відповідають на голосові команди, а також ті які конспектують великі об’єми інформації інколи у режимі реального часу. Майже кожна людина хоча б раз у своєму житті взаємодіяла з такими системами у формі GPS систем з голосовим управлінням.

Дана предметна область є неймовірно складною через те, що людська мова наповнена безліччю неоднозначних речей через які важко точно визначити який сенс було вкладено в текстові чи аудіо дані. Омоніми, сарказм, метафори, відмінності в граматиці та різні винятки у застосуванні слів та конструкцій це лише невелика частина складнощів через які вивчення мов у людей займає роки проте програмісти мусять навчити модель одразу та з високою точністю виконувати свою задачу для того щоб застосунок який її використовує можна було назвати корисним.

До задач NLP можна віднести наступне:

— Розпізнавання мови (Speech recognition) — це завдання яке полягає в конвертації аудіо даних в текстові дані. Розпізнавання мови використовується застосунками які реагують на голосові команди або відповідають на питання. Ця задача є особливо складною через те, що люди не завжди вимовляють слова ідеально, а саме: вимовляють їх швидко, вимовляють декілька слів без інтервалу як одне, з різними інтонаціями та наголосами та часто з неправильним використанням граматики

— Тегування частини мови (Part of speech tagging) — це процес визначення до якої частини мови належить те чи інше слово з певного фрагменту тексту зважаючи на його використання та контекст. Наприклад слово ‘make’ є дієсловом в реченні ‘I can make a paper plane’, але іменником в реченні ‘What make of car do you own?’.

— Визначення сенсу слова (Word sense disambiguation) — це процес семантичного аналізу задачею якого є визначити який сенс було вкладено в слово зважаючи на контекст. Наприклад слово ‘make’ у виразі ‘make the grade’ означає досягнути, а у виразі ‘make a bet’ має значення зробити.

— Розпізнавання іменованих сутностей (Named entity recognition) — це задача яка полягає у визначенні до якого саме класу належить та чи інша сутність. Наприклад така сутність як ‘Kentucky’ буде визначена як географічна локація, тоді як ‘Fred’ буде визначено як чоловіче ім’я.

2.2 Обґрунтування обраних програмних засобів

Для реалізації користувацького інтерфейсу системи було обрано наступні технології:

— Мова високого рівня Java Script [6,7], що має динамічну типізацію, підтримується всіма популярними веб браузерами, дозволяє створювати інтерактивний інтерфейс який швидко реагує на будь-які дії користувача та автоматично виконує задачі менеджменту ресурсами комп'ютера.

— Бібліотека для побудови інтерфейсів React [8,9], що розповсюджується компанією Facebook і дозволяє створювати комплексні застосунки з невеликих елементів що звуться компонентами, а також керувати всіма етапами їх життєвого циклу.

— Мова розмітки HTML [9,10], що є стандартом для розмітки гіпертексту який є основою будь-якого веб сайту в мережі інтернету.

— Каскадні таблиці стилів CSS [11], що дозволяють надати елементам гіпертексту будь-якого кольору та форми.

Для розробки серверної частини застосунку та тренування моделі для класифікації фейкових новин було обрано такі технології як:

— Мова програмування Python [12-15], що є інтерпретованою мовою високого рівня з об'єктно-орієнтованим підходом, дозволяє писати компактний та зрозумілий для інших розробників програмний код, має безліч вбудованих конструкцій та методів які дозволяють вирішувати широкий клас задач та є одним з основних інструментів в галузі аналітики даних та розробки систем з використанням штучного інтелекту.

— Веб-фреймворк Django [16,17], що написаний на мові програмування Python та має відкритий код. Даний фреймворк дозволяє в стиснуті терміни створювати безпечні, високоякісні застосунки до яких можна легко вносити зміни та при цьому не хвилюючись що весь програмний код доведеться переписувати з

нуля. Характерними рисами даного фреймворку можна назвати універсальність, масштабованість та завершеність

Висновки до розділу 2

В ході даної роботи було обрано програмні засоби обрані для реалізації поставлених задач, а також наведено їх особливості які стали вирішальним фактором при їх виборі. Було розглянуто предметну область даної магістерської дисертації та наведено основні напрямки за якими на даний час ведуться роботи та проводяться дослідження.

3 АПАРАТ ВИРІШЕННЯ ДЛЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

При розробці системи для визначення фейкових новин основною та найскладнішою частиною реалізації є вибір правильного алгоритму для навчання моделі а також збір та підготовка даних для навчання самої моделі.

3.1 Підготовка даних для навчання моделі

На рисунку 3.1 зображено кроки які виконуються при підготовці даних перед тим як їх використовують для навчання моделей.

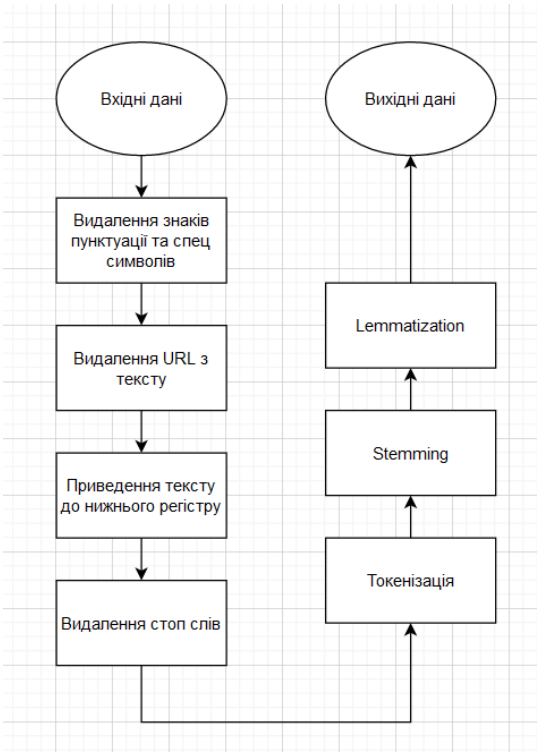


Рисунок 3.1 — Послідовність дій при підготовці даних

Всі ці кроки направлені на те щоб якомога більше зменшити час тренування моделей, збільшити точність їх роботи та зробити більш придатними для

використання на нових даних а не тільки тих що було в датасеті на якому проводилося тренування.

На першому кроці з тексту прибираються всі розділові знаки а також всі інші спеціальні символи такі як . , ! $( ) * % @ оскільки вони не несуть в собі ніякої цінної інформації. На рисунку 3.2 зображено як виглядає текст після роботи функції для видалення спеціальних символів. В лівій половині знаходиться початковий текст, а у правій текст після обробки.

v2	clean_msg
Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...	Go until jurong point crazy Available only in bugis n great world la e buffet Cine there got amore wat
Ok lar... Joking wif u oni...	Ok lar Joking wif u oni
Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005 Text FA to 87121 to receive entry questionstd txt rateTCs apply 08452810075over18s
U dun say so early hor... U c already then say...	U dun say so early hor U c already then say
Nah I don't think he goes to usf, he lives around here though	Nah I dont think he goes to usf he lives around here though

Рисунок 3.2 — Текст після видалення спеціальних символів

Наступним кроком є видалення веб адрес тому, що вони не несуть в собі інформації про що саме той чи інший текст, а також тому що їх не можна вважати словами та їх існують мільярди в мережі інтернет.

На третьому етапі весь текст приводиться до нижнього регістру (рис. 3.3).

clean_msg	msg_lower
Go until jurong point crazy Available only in bugis n great world la e buffet Cine there got amore wat	go until jurong point crazy available only in bugis n great world la e buffet cine there got amore wat
Ok lar Joking wif u oni	ok lar joking wif u oni
Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005 Text FA to 87121 to receive entry questionstd txt rateTCs apply 08452810075over18s	free entry in 2 a wkly comp to win fa cup final tkts 21st may 2005 text fa to 87121 to receive entry questionstd txt ratetcs apply 08452810075over18s

Рисунок 3.3 — Приведення тексту до нижнього регістру

Дана операція проводиться для того щоб одне й те саме слово яке було написано з великої букви, прописними літерами, маленькими або якимсь іншим чином мало єдиний формат.

В наступному етапі з тексту видаляються стоп слова (рис. 3.4). Під стоп словами слід розуміти такі речі як сполучники (і, а, але, та, бо, як, щоб, та ін.), прийменники (в, до, по, за, на, під, для, поза, понад, та ін.), частки (а, не, ж, би, та, і, та ін.) та займенники (я, ти, він, вона, воно, ми, ви, вони, та ін.). Дані слова складають значну частину будь-якого тексту та не несуть в собі основного значення. Даний етап є дуже важливим бо значно зменшує розмір вихідної матриці, що прискорює швидкість тренування моделей таким чином даючи розробникам більше можливостей для експериментів та зменшуючи витрати на дороговартісні сервери які використовують для тренування.

msg_tokenied	no_stopwords
[go, until, jurong, point, crazy, vailable, only, in, bugis, n, great, world, la, e, buffet, cine, there, got, amore, wat]	[go, jurong, point, crazy, available, bugis, n, great, world, la, e, buffet, cine, got, amore, wat]
[ok, lar, joking, wif, u, oni]	[ok, lar, joking, wif, u, oni]
[free, entry, in, 2, a, wkly, comp, to, win, fa, cup, final, tkts, 21st, nay, 2005, text, fa, to, 87121, to, receive, entry, questionstd, txt, ratetcs, apply, 08452810075over18s]	[free, entry, 2, wkly, comp, win, fa, cup, final, tkts, 21st, may, 2005, text, fa, 87121, receive, entry, questionstd, txt, ratetcs, apply, 08452810075over18s]
[u, dun, say, so, early, hor, u, c, already, then, say]	[u, dun, say, early, hor, u, c, already, say]
[nah, i, dont, think, he, goes, to, usf, he, lives, around, here, though]	[nah, dont, think, goes, usf, lives, around, though]

Рисунок 3.4 — Текст після видалення стоп слів

Наступним етапом в підготовці даних є токенізація. Токенізацією називають процес поділу тексту на малі структурні одиниці такі як слова чи речення в залежності від поставленої задачі. В даній роботі використовується токенізація з розбиттям на слова. Дану операцію легко виконати після всіх попередніх кроків для цього необхідно розбити текст за символом пробілу. В результаті з кожного фрагменту тексту ми отримуємо масив який містить послідовність з рядкових

значень кожне з яких репрезентує одне зі слів з початкового тексту. На рисунку 3.5 зображено як виглядає текст до і після токенізації

msg_lower	msg_tokenied
go until jurong point crazy available only in bugis n great world la e buffet cine there got amore wat	[go, until, jurong, point, crazy, available, only, in, bugis, n, great, world, la, e, buffet, cine, there, got, amore, wat]
ok lar joking wif u oni	[ok, lar, joking, wif, u, oni]
free entry in 2 a wkly comp to win fa cup final tkts 21st may 2005 text fa to 87121 to receive entry questionstd txt ratetcs apply 08452810075over18s	[free, entry, in, 2, a, wkly, comp, to, win, fa, cup, final, tkts, 21st, may, 2005, text, fa, to, 87121, to, receive, entry, questionstd, txt, ratetcs, apply, 08452810075over18s]
u dun say so early hor u c already then say	[u, dun, say, so, early, hor, u, c, already, then, say]
nah i dont think he goes to usf he lives around here though	[nah, i, dont, think, he, goes, to, usf, he, lives, around, here, though]

Рисунок 3.5 — Текст після токенізації

Передостаннім етапом підготовки даних є Stemming (рис. 3.6). Stemming — це процес який називають стандартизацією тексту.

no_stopwords	msg_stemmed
[go, jurong, point, crazy, available, bugis, n, great, world, la, e, buffet, cine, got, amore, wat]	[go, jurong, point, crazi, avail, bugi, n, great, world, la, e, buffet, cine, got, amor, wat]
[ok, lar, joking, wif, u, oni]	[ok, lar, joke, wif, u, oni]
[free, entry, 2, wkly, comp, win, fa, cup, final, tkts, 21st, may, 2005, text, fa, 87121, receive, entry, questionstd, txt, ratetcs, apply, 08452810075over18s]	[free, entri, 2, wkli, comp, win, fa, cup, final, tkt, 21st, may, 2005, text, fa, 87121, receiv, entri, questionstd, txt, ratetc, appli, 08452810075over18]

Рисунок 3.6 — Стандартизація тексту

На цьому етапі спільнокореневі слова приводять до єдиної базової форми, що як і у випадку з видаленням стоп слів значно зменшує розміри вихідної матриці.

Останнім етапом в підготовці даних є Lemmatization. Lemmatization — це процес схожий на стандартизацію тексту проте в ньому є одна відмінність. Набір токенів отриманий після цієї операції перевіряється за допомого словника який зберігає в собі контекст використання слів та перевіряє чи не був втрачений початковий сенс слова. На рисунку 3.7 зображено результати виконання Lemmatization та можна порівняти з тим що ми отримуємо після стандартизації тексту.

no_stopwords	msg_stemmed	msg_lemmatized
[go, jurong, point, crazy, available, bugis, n, great, world, la, e, buffet, cine, got, amore, wat]	[go, jurong, point, crazi, avail, bugi, n, great, world, la, e, buffet, cine, got, amor, wat]	[go, jurong, point, crazy, available, bugis, n, great, world, la, e, buffet, cine, got, amore, wat]
[ok, lar, joking, wif, u, oni]	[ok, lar, joke, wif, u, oni]	[ok, lar, joking, wif, u, oni]

Рисунок 3.7 — Текст після виконання Lemmatization

Таким чином після всіх кроків ми отримуємо текст який складається з мінімальної кількості слів при цьому наскільки це можливо зберігається сенс який було вкладено в початкове повідомлення.

3.2 Застосування методу Random forest

Першим з досліджуваних методів для визначення фейкових новин в цій роботі є метод Random forest. Random forest — це ансамблевий метод штучного інтелекту для класифікації, регресії та інших задач, який працює за рахунок побудови множини дерев прийняття рішень в процесі тренування моделі та продукує на їх основі усереднений прогноз (регресія) або моду для класів (класифікацій). Слабкою стороною цього методу є схильність до надмірного

навчання. Покращення алгоритму було запропоновано Аделем Катлером та Лео Брейманом. Алгоритм використовує в собі дві ідеї: метод випадкових підпросторів та метод беггінга Бреймана.

Кожне дерево будується незалежно одне від одного за наступним алгоритмом:

— Генеруємо випадковим чином підвибірку з повтореннями розміром k з навчальної вибірки. (В результаті деякі елементи потраплять до неї декілька разів, а приблизно K/3 елементів не потраплять до неї взагалі).

— Вибираємо випадковим чином n предикторів (ознак) із N.

— Будуємо дерево прийняття рішень для класифікації елементів даної підвибірки, причому під час створення кожного нового вузла будемо обирати ознаку для розбиття, не зі всього набору N ознак, а лише з n ознак які було обрано випадковим чином. Визначення найкращої з ознак може проводитись різними методами. В оригінальному алгоритмі Бреймана застосовується критерій Джині, який також використовується для створення вирішальних дерев CART. Деякі інші реалізації даного алгоритму використовують замість критерію Джині критерій приросту інформації.

— Поділяємо ознаку X на два окремих класи, Xi ≥ Si та Xi < Si.

— Визначаємо гомогенність у двох отриманих класах використовуючи критерій Джині.

— Вибираємо таке значення точки поділу Si ознаки X, при якому досягається найбільше значення гомогенності класу.

— Дерево будується до цілковитого вичерпання підвибірки без застосування процедури відсікання (в противагу від дерев прийняття рішень які будуються з використанням таким алгоритмам, як CART чи C4.5).

— Повертаємося до першого пункту. створюємо нову вибірку та повторюємо етапи 2-4 створюючи наступне дерево. Точність класифікатора на тестовій вибірці залежить від кількості побудованих дерев. Чим більше було побудовано тим більша точність

В спрощеному вигляді процес роботи даного алгоритму виглядає наступним чином: будується певне число дерев для класифікації після чого результат визначається на основі того варіанту який набрав найбільшу кількість голосів під час голосування (рис. 3.8).

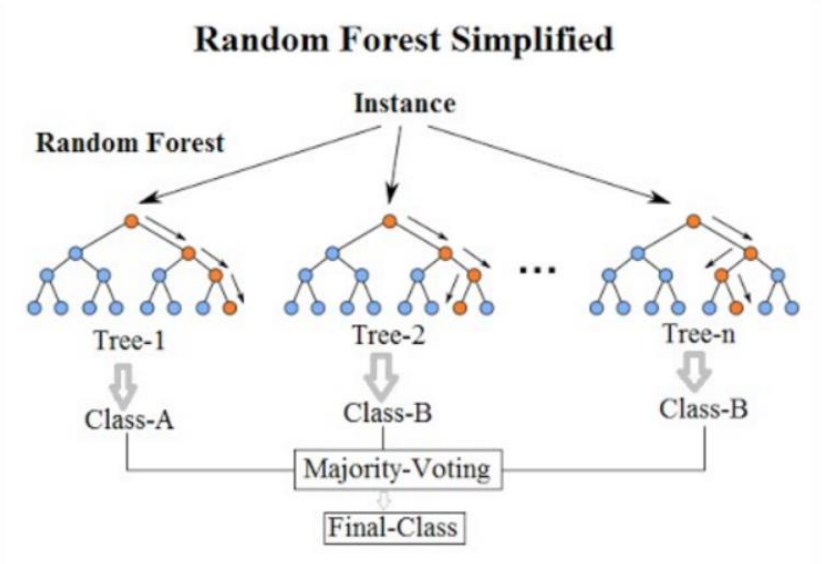


Рисунок 3.8 — Спрощена демонстрація роботи алгоритму Random forest

Сильними сторонами даного методу є:

— Можливість ефективно працювати з даними у яких велика кількість ознак та класів.

— Нечутливість до масштабування та монотонних перетворень значень ознак.

— Однаково добре обробляються як безперервні, так і дискретні ознаки.

— Можливість побудови дерев з використанням даних в яких були пропущені значення деяких ознак.

— Здатність працювати паралельно в багато потоків.

Слабкими сторонами алгоритму є:

— Схильність до перенавчання на деяких завданнях, особливо з великою кількістю шумів.

— Великий розмір отримуваних моделей.

3.3 Застосування алгоритму Decision tree

Decision tree (Дерево ухвалення рішень) — це метод який застосовується в області статистики, а також для створення прогнозних моделей в галузі аналізу даних. В результаті своєї роботи алгоритм продукує дерево яке показує набір можливих виборів та наслідків які з них випливають включаючи шанс настання випадкових подій та кількість ресурсів необхідних для виконання завдання.

Кожне дерево прийняття рішень складається з двох структурних одиниць таких як «листки» та «гілки» (рис. 3.9). На «гілках» дерева записуються атрибути на основі яких відбувається ухвалення рішення та залежить цільова функція. У вузлах дерева «листах» можуть записуватись значення цільової функції, а також атрибути на основі яких відбувається розгалуження в структурі дерева.

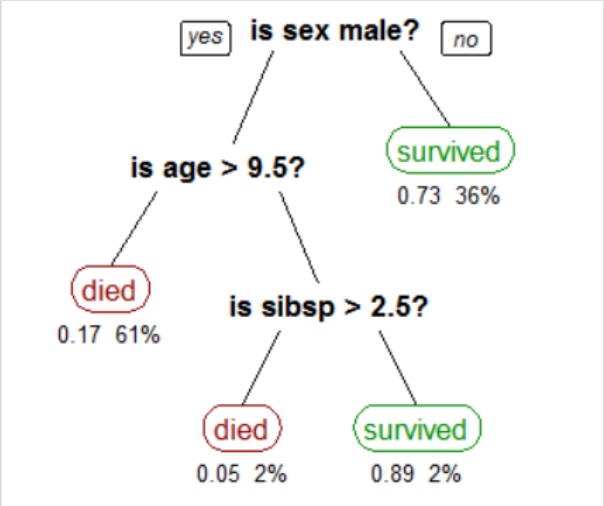


Рисунок 3.9 — Схематичний вигляд дерева прийняття рішень

Метою даного алгоритму є створення моделі яку можна буде використати для прогнозу певної величини на основі якогось числа вхідних змінних. Для того щоб отримати рішення на основі нових даних необхідно пройтись по дереву від його вершини до кінцевого листа в якому не буде відбуватись розгалуження і буде записано значення цільової функції. Такі дерева широко використовують як

інструмент допомоги прийняття рішень, а також вони є дуже популярними в галузі штучного інтелекту для задач класифікації.

На рисунку 3.10 зображено три основні типи позначень які зазвичай використовують при візуалізації дерев прийняття рішень:

— квадрати чи прямокутники для вузлів в яких описуються умови прийняття рішення;

— кола для позначення ймовірності настання події яка була описана в блоці вище;

— трикутники які є кінцевими вузлами в будь-якому дереві, що показують фінальне рішення.

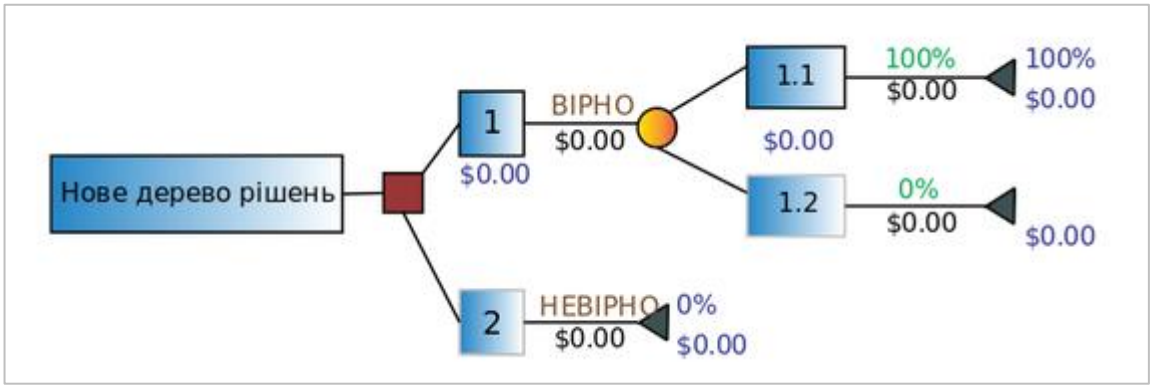


Рисунок 3.10 — Стандарти побудови дерев прийняття рішень

В деяких випадках дерева можуть будувати зліва на право як на рисунку вище або як це роблять частіше всього зверху до низу. Особливістю дерев прийняття рішень є те, що вони не можуть мати циклічних конструкцій. З цього слідує, що різні гілки одного дерева ніколи не можуть перетинатися між собою і можуть або розгалужуватись далі або закінчуватись кінцевим вузлом.

На практиці отримувані дерева часто бувають великого розміру, тому при роботі з ними зазвичай використовую спеціальне програмне забезпечення яке полегшує цю задачу.

Для побудови дерев прийняття рішень існує декілька методів серед яких: ID3,

C4.5, CART, MARS та інші. Кожен з цих методів використовує свою формулу для визначення ефективності розгалуження за певним з параметрів. Розглянемо процес побудови дерева рішень з використанням алгоритму CART. Даний алгоритм складається з наступних кроків:

— Для того щоб обрати параметр для кореня дерева обчислюємо значення параметру Джині для кожної з вхідних змінних. Даний параметр показує на скільки сильно кожна окрема незалежна змінна пов'язана з шуканою залежною змінною яку ми намагаємося передбачити.

— Вибираємо для кореня дерева ту змінну для якої параметр Джині має найменше значення.

— Виключаємо обраний параметр з процесу побудови подальших вузлів дерева.

— Для кожного з вузлів що ми отримали після розгалуження обчислюємо значення параметру Джині для кожної зі змінних що залишилися. Варто зазначити, що для кожного наступного вузла на дереві необхідно брати ті елементи які залишились після попередніх поділів.

— Продовжуємо будувати дерево до тих пір поки неможливо буде вибрати параметр при якому параметр Джині буде зменшуватись або до тих пір поки не будуть вичерпані всі можливі параметри.

Після того як дерево було побудовано постає нова задача, а саме: зменшення розміру дерева для економії пам'яті та пришвидшення його роботи. Для досягнення цієї цілі існують два підходи:

— скорочення дерева зверху вниз при якому видаляється корінь дерева;

— скорочення дерева знизу вверх при якому видаляється певне число нижніх вузлів дерева.

Найпопулярнішим методом для скорочення розміру дерев є скорочення знизу вверх. Суть обох методів полягає у тому щоб видалити якомога більше вузлів та при цьому максимально зберегти точність прогнозів які дає дерево.

До переваг дерев прийняття рішень можна віднести:

— Простота для розуміння та інтерпретації. Графічно зображене дерево прийняття рішень інтуїтивно зрозуміле, тому правила користування ним можна легко пояснити будь-кому.

— Дерево прийняття рішень можна створити без датасету опираючись лише на дані зі сценаріїв розроблених експертами в певній предметній області

— Можливість передбачити результати широкого ряду сценаріїв від оптимістичного до реалістичного та навіть песимістичного.

— Алгоритм працює за принципом білого ящика.

— Можливість використання отриманої моделі в комбінації з іншими методами допомоги прийняття рішень

До недоліків дерев прийняття рішень можна віднести:

— Нестабільність дерев яка виражається в тому, що навіть невелика зміна вхідних даних може вести до кардинально іншого оптимального рішення. Таким чином неможливо перебудувати існуюче дерево для нових даних, а потрібно постійно будувати нове.

— Одне окреме дерево прийняття рішень дуже часто програє в точності іншим алгоритмам машинного навчання. Даний недолік можна виправити скориставшись методами машинного навчання які в процесі своєї роботи використовують набір дерев, але дані методи є набагато важчими в своїй реалізації та результати їх роботи неможливо так просто інтерпретувати як у випадку з одним деревом.

— Значне підвищення складності обчислень у випадку коли серед вхідних змінних є ті, що репрезентують категорії, наприклад: країни, міста, мови, імена та інше.

— Значне підвищення складності обчислень у випадку коли багато вхідних значень є невизначеними.

— Значний розмір отримуваних дерев у випадку коли для навчання використовується датасет з великою кількістю змінних.

3.4 Застосування методу опорних векторів

Метод опорних векторів — це метод який використовується в машинному навчанні для задач класифікації та регресійного аналізу. В більшості випадків даний метод використовується при роботі з розміченими датасетами, тобто такими де є не тільки набір незалежних змінних, а й відомо значення залежної змінної.

Через принцип роботи даного алгоритму його неможливо напряму використовувати для задачі класифікації небінарних даних. Для того щоб обійти це обмеження можна скористатись методом який називається «один проти всіх». Суть даного методу полягає в тому, що для кожного класу тренується своя окрема модель. Перед тренуванням кожної з моделей створюється копія початкового датасету в якому значення залежної змінної того класу для якого тренується модель залишається без змін, а значення залежних змінних для всіх інших класів роблять однаковими. Саме тому цей підхід і отримав назву «один проти всіх».

Метод опорних векторів було розроблено в 1963 році двома вченими яких звали Володимир Вапник та Олексій Червоненікс. В базовому варіанті даного методу було використано лінійний класифікатор. Пізніше в 1992 році Володимир Вапник разом з Ізабель Гійон та Бернхардом Босером вдосконалили даний метод запропонувавши використовувати спеціальні функції ядра для створення нелінійних класифікаторів. На поточний момент застосовується варіація даного методу, що використовує так зване «м’яке розділення», його було створено в 1993 році Володимиром Вапником та Корінною Кортес та опубліковано у виді наукової праці в 1995 році.

В загальному робота будь-якої з варіацій методу опорних векторів зводиться до знаходження такої гіперплощини яка зможе якнайкраще розділити елементи різних класів таким чином щоб максимізувати відстань найближчих до площини елементів. Завдяки тому, що між гіперплощиною та елементами створюється максимальний відступ даний метод є стійким до перенавчання.

На рисунку 3.11 зображено результати роботи алгоритму з використанням лінійного класифікатора.

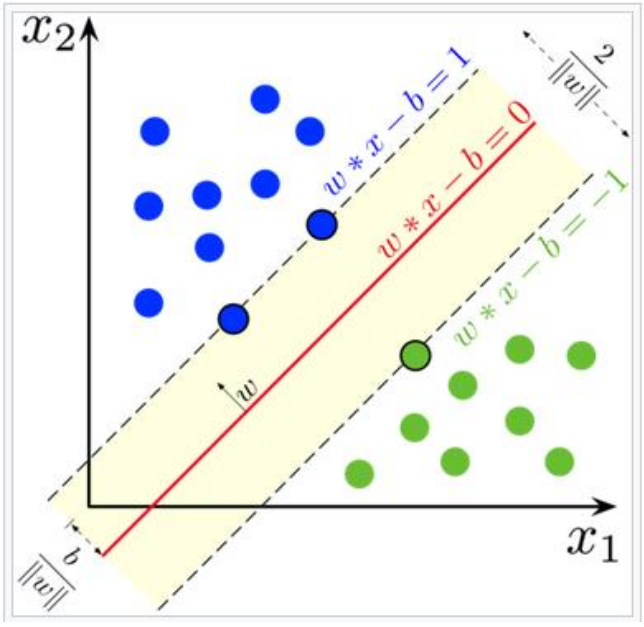


Рисунок 3.11 — Метод опорних векторів з лінійним класифікатором

Так як в даному прикладі дані є точками на двовимірній площині, то результатом роботи алгоритму є n-1 вимірна площина, що в конкретно взятому випадку є простою лінією. Інтерпретувати результати слід наступним чином: все що находиться вище або на верхній пунктирній лінії відноситься до першого класу, а все що знаходиться нижче або на нижній пунктирній лінії відноситься до другого класу. Пунктирні ліній про які йде мова — це відступи від площини до крайніх елементів. Чим більші ці відступи тим краще отримана модель буде справлятись з новими даними. Також через ці відступи даний метод часто називають методом великих відступів.

Принцип роботи методу опорних векторів з використанням нелінійного класифікатора в цілому збігається з тим, що використовує лінійний класифікатор, за виключенням того, що кожний скалярний добуток замінюється спеціальною нелінійною функцією ядра. Дана функція ядра дозволяє знайти найоптимальнішу гіперплощину для розділу в перетвореному просторі змінних. Недоліком цього

методу є зменшення точності отримуваних результатів при роботі з даними високої вимірності. Даний недолік є не таким суттєвим у випадку коли для навчання моделі використовується достатньо обширний датасет.

На рисунку 3.12 зображено результати роботи методу опорних векторів з використанням нелінійного класифікатора.

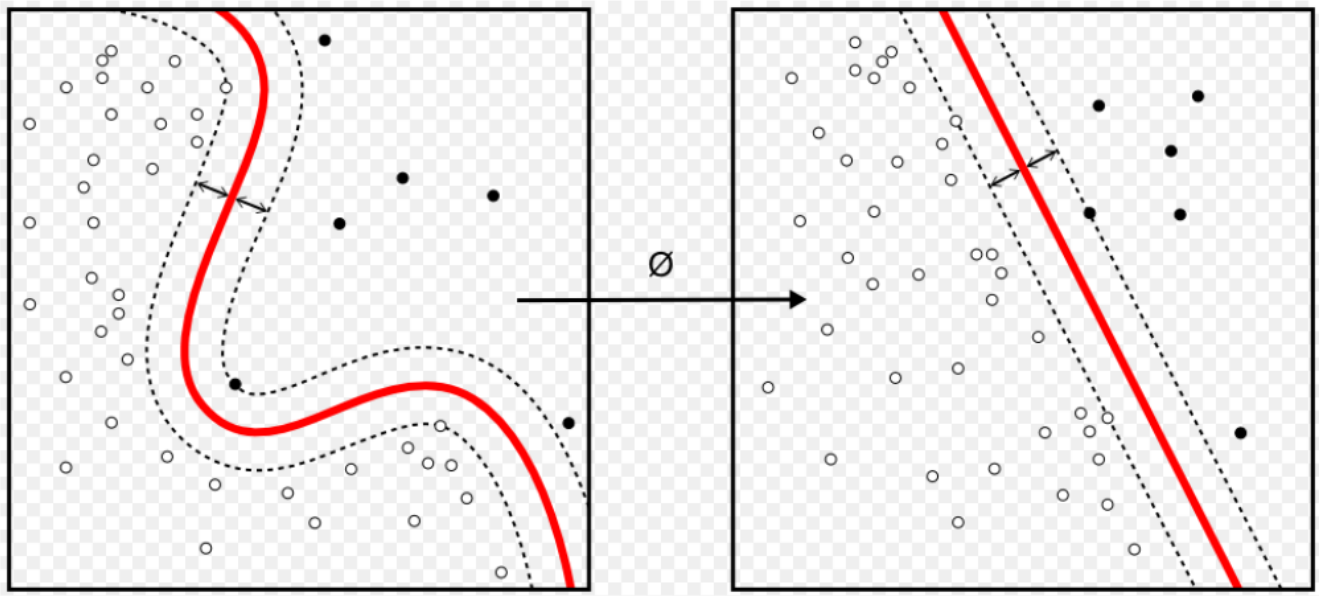


Рисунок 3.12 —Метод опорних векторів з нелінійним класифікатором

Одними з основних функцій ядра які використовують для нелінійної класифікації є:

— поліноміальне неоднорідне ядро;

— поліноміальне однорідне ядро;

— сигмоїдне (гіперболічний тангенс) ядро;

— Гаусове радіально-базисне ядро.

До недоліків методу опорних векторів можна віднести:

— Необхідність мати повністю розмічений датасет (той в якому відомо яке значення залежної змінної відповідає набору незалежних змінних).

— Неможливість використовувати напряму для класифікації даних в яких більше двох цільових класів.

— Параметри отримуваної в ході тренування моделі важкі для інтерпретування.

3.5 Застосування логістичної регресія

Логістична регресія є одним зі статистичних методів, що широко використовується в регресійному аналізі. Даний метод застосовують для задач де необхідно класифікувати бінарну залежну змінну. Проте дане обмеження можна обійти використовуючи метод «один проти всіх» подібно до того як це робиться для методу опірних векторів.

Логістичну регресію широко застосовують в різних сферах серед яких в основному медичні та соціальні дослідження, а також машинне навчання. Можна навести наступні приклади використання:

— Передбачення того, що пацієнт хворий на певну хворобу, або шансу того, що хвороба виникне в майбутньому, або передбачення шансу вижити при наборі певних травм, захворювань або симптомів.

— Передбачення того за кого може проголосувати на виборах людина базуючись на її статі, вікові, соціальному положенню, релігії, рівню доходів та історії її попередніх голосувань.

— Передбачення аномалій які можуть спричиняти брак продукції, зупинку якось процесу або навіть цілої системи.

— Передбачення шансу того, що певна людина купить той чи інший товар базуючись на її статі, вікові та вподобаннях.

В загальному принцип роботи логістичної регресії складається з такої послідовності кроків:

— Генеруємо випадковим чином у проміжку від -1 до 1 значення для вільного члену та значення ваг для кожного з вхідних параметрів.

— Обчислюємо значення функції передбачення для кожного з елементів датасету.

— До кожного з результатів передбачення застосовуємо одну з функцій активації, що проектує будь-яке вхідне значення на проміжок від нуля до одиниці, що вже можна інтерпретувати як ймовірність настання події чи належність елементу до одного з класів.

— Використовуючи отримані в попередньому кроці результати обчислюємо значення однієї з функцій для обчислення похибки.

— На основі отриманого значення функції похибки проводимо корегування значень вільного члену та ваг.

— Повторюємо всі кроки до тих пір поки різниця між значеннями функції похибки на поточному та попередньому кроках не стане меншою ніж певна встановлена межа.

На рисунку 3.13 зображено приклад такої поширеної у використанні функції активації як сигмоїда.

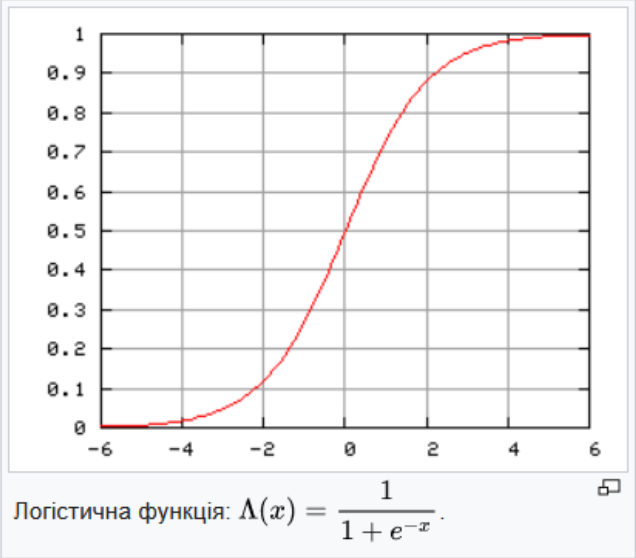


Рисунок 3.13 — Приклад функції активації

Результати роботи даної функції активації слід інтерпретувати наступним чином: елементи значення яких менше або рівне 0.5 слід віднести до негативного

класу, а елементи значення яких більше від 0.5 слід віднести до позитивного класу. В залежності від задачі межа для розділення двох класів може змінюватись, наприклад: у випадку якщо необхідно класифікувати захворювання на рак, то хибно негативний результат може призвести до смерті пацієнта тому в даному випадку межу поділу доцільно зсунути трохи вліво. З іншого боку такий підхід вірогідно стане причиною ряду хибно позитивних діагнозів, що спричинить сильні хвилювання в пацієнтів. Тому для кожної окремої задачі необхідно вирішувати що гірше: хибно позитивні чи хибно негативні результати.

Для того щоб оцінити точність роботи отриманої моделі використовують такі метрики як Accuracy та F1-score. Accuracy показує відсоткове відношення правильно класифікованих елементів до загального числа елементів. Даний метод застосовується у випадках коли відношення кількості між елементами позитивного та негативного класів є приблизно рівним.

На відміну від Accuracy, F1-score використовується для оцінки точності роботи у випадках коли один з класів значно переважає над іншим, наприклад: один до десяти, сотні чи тисячі. Для прикладу якщо розглядати дані обстежень пацієнтів з підозрою на рак не проводячи спеціальний відбір даних, то відсоток записів з позитивним діагнозом буде один до десяти тисяч. З цього випливає наступна проблема: якщо абсолютно всі записи класифікувати як негативні, то ми отримаємо величезний показник Accuracy 99.99% хоча з такої моделі не буде жодної користі.

Для того щоб знайти F1-score для початку потрібно побудувати confusion matrix (рис. 3.14). Дана матриця показує наступні речі:

— кількість істинно позитивних передбачень (TP);

— кількість істинно негативних передбачень (TN);

— кількість хибно позитивних передбачень (FP);

— кількість хибно негативних передбачень (FN).

Після цього необхідно обчислити значення таких двох параметрів як Precision та Recall. F1-score вираховується на основі цих двох параметрів і виражає їх гармонічне середнє.

	Predicted Positives	Predicted Negatives
Positives	True Positives	False Negatives
Negatives	False Positives	True Negatives

Рисунок 3.14 — Принцип побудови confusion matrix

Precision обчислюють за наступною формулою:

$$\frac{TP}{TP + FP} \tag{3.1}$$

Recall обчислюють за наступною формулою:

$$\frac{TP}{TP + FN} \tag{3.2}$$

F1-score обчислюється за формулою:

$$2 * \frac{Precision\ * Recall}{Precision + Recall} \tag{3.3}$$

Для того щоб обчислити функцію втрат використовують логарифмічну функцію втрат. Дана функція виражає середнє від’ємне похибки класифікації. Функція середніх квадратів не підходить для даного типу задач оскільки вона може мати локальні мінімуми (рис. 3.15). Через це немає ніяких гарантій, що алгоритм

зможе знайти глобальний мінімум і при послідовних запусках тренування ми будемо отримувати кожного разу різний результат

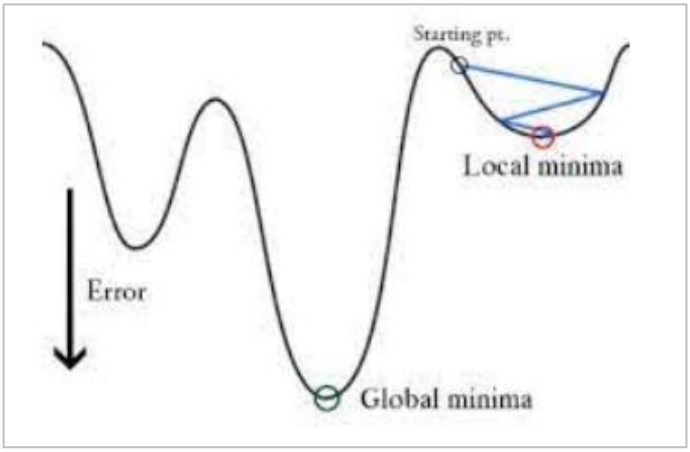


Рисунок 3.15 — Функція середніх квадратів для логістичної регресії

Логістична регресія має наступні переваги у використанні:

— Вона легка в програмній реалізації та є швидкою в тренуванні порівняно з іншими алгоритмами машинного навчання.

— Вона здатна швидко класифікувати нові елементи яких не було в тренувальному датасеті.

— Коефіцієнти отримуваної в процесі тренування моделі показують наскільки сильний зв’язок між незалежними змінними та залежною змінною.

— Алгоритм менш схильний до перенавчання на простих датасетах, хоча при використанні датасетів з великою розмірністю це може статися, проте дану проблему можна легко вирішити за допомогою регуляризації.

Також в даного алгоритму присутні наступні недоліки:

— Логістичну регресію неможливо застосовувати для тренування моделей на датасетах в яких кількість елементів менша від кількості незалежних змінних оскільки це призведе до перенавчання.

— Робота алгоритму базується на припущенні, що між незалежними змінними та залежною змінною існує лінійний зв’язок.

— За допомогою логістичної регресії неможливо побудувати складні зв’язки між вхідними змінними і тому даний алгоритм програє в цьому аспекті таким методам штучного інтелекту як нейронні мережі.

3.6 Розгортання інфраструктури системи

Для того щоб створити інфраструктуру системи та налаштувати процеси CI/CD було обрано скористатись послугами таких сервісів як AWS та Gitlab. AWS було обрано через широкий вибір сервісів та їх гарну підтримку, а Gitlab через наявність вбудованих рішень для CI/CD, що скорочує час для налаштувань.

Для того щоб почати користуватись Gitlab потрібно пройти просту реєстрацію, тоді як для створення облікового запису на AWS необхідно вказати набагато більше даних, а також прив’язати номер мобільного телефону та налаштувати платіжний метод.

Після того як облікові записи були створені перш за все необхідно створити сервер на якому ми зможемо розгорнути серверну та клієнтські частини. Для створення серверу потрібно скористатись сервісом EC2 який надає широкий вибір конфігурації потужностей обладнання, та наперед сконфігурованих дистрибутивів операційної системи Linux.

В загальному при створенні серверу необхідно виконати такі обов’язкові кроки як:

— Обрати один з дистрибутивів операційної системи Linux або вибрати одну з версій Windows що придатна для створення серверу (рис. 3.16).

— Використовуючи один з доступних типів шифрування створити пару ключів для подальшого встановлення безпечного доступу до серверу

— Відштовхуючись від прогнозованого навантаження на сервер обрати за кількістю ядер процесора, об’ємом RAM та максимальною пропускною здатністю мережі один з типів серверів який підходить найкраще.

— Обрати один з існуючих VPC (Virtual Private Network) або створити новий в якому всі налаштування будуть встановлені за замовченням.

— Обрати одну з існуючих Security Group або створити нову. При створенні нової Security Group будь-який вхідний та вихідний трафік буде заблоковано, тому необхідно дозволити трафік як мінімум для портів 22 ssh та 80 HTTP.

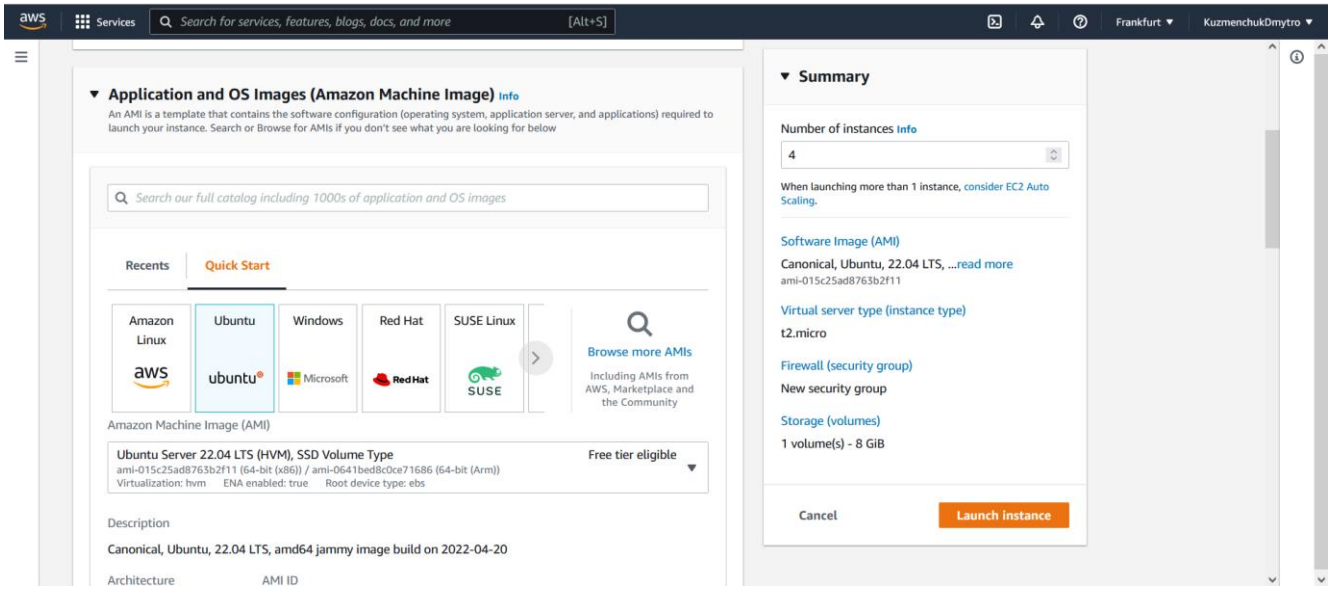


Рисунок 3.16 — Вибір операційної системи для серверу

Після того як всі базові налаштування були встановлені можна запустити сервер. Запуск серверу зазвичай займає не більше декількох хвилин і його стан можна відслідковувати на панелі поряд з іншими серверами.

Для того щоб під’єднатися до щойно створеного серверу необхідно помістити в одну з робочих директорій ключ з розширенням .pem та виконати команди для обмеження доступу до цього файлу. Для того щоб обмежити доступ до файлу з ключем на операцій системі Linux необхідно виконати наступну команду: chmod 400 ~/.ssh/key_name.pem. У випадку якщо використовується операційна система Windows необхідно відімкнути наслідування прав доступу в розширених налаштуваннях безпеки, видалити всі доступи та створити їх для єдиного користувача яким є ви.

Після того як було налаштовано доступ до файлу необхідно відкрити термінал та виконати команду: ssh -i your-key-pair.pem ec2-user@<PUBLIC-IP-ADDRESS> (рис. 3.17). Дана команда дозволяє за допомогою протоколу ssh встановити з'єднання з віддаленою машиною яка має доступ до мережі інтернету та підтримує даний протокол. Команда приймає такі обов'язкові параметри як: шлях до зашифрованого ключа та адресу віддаленої машини.

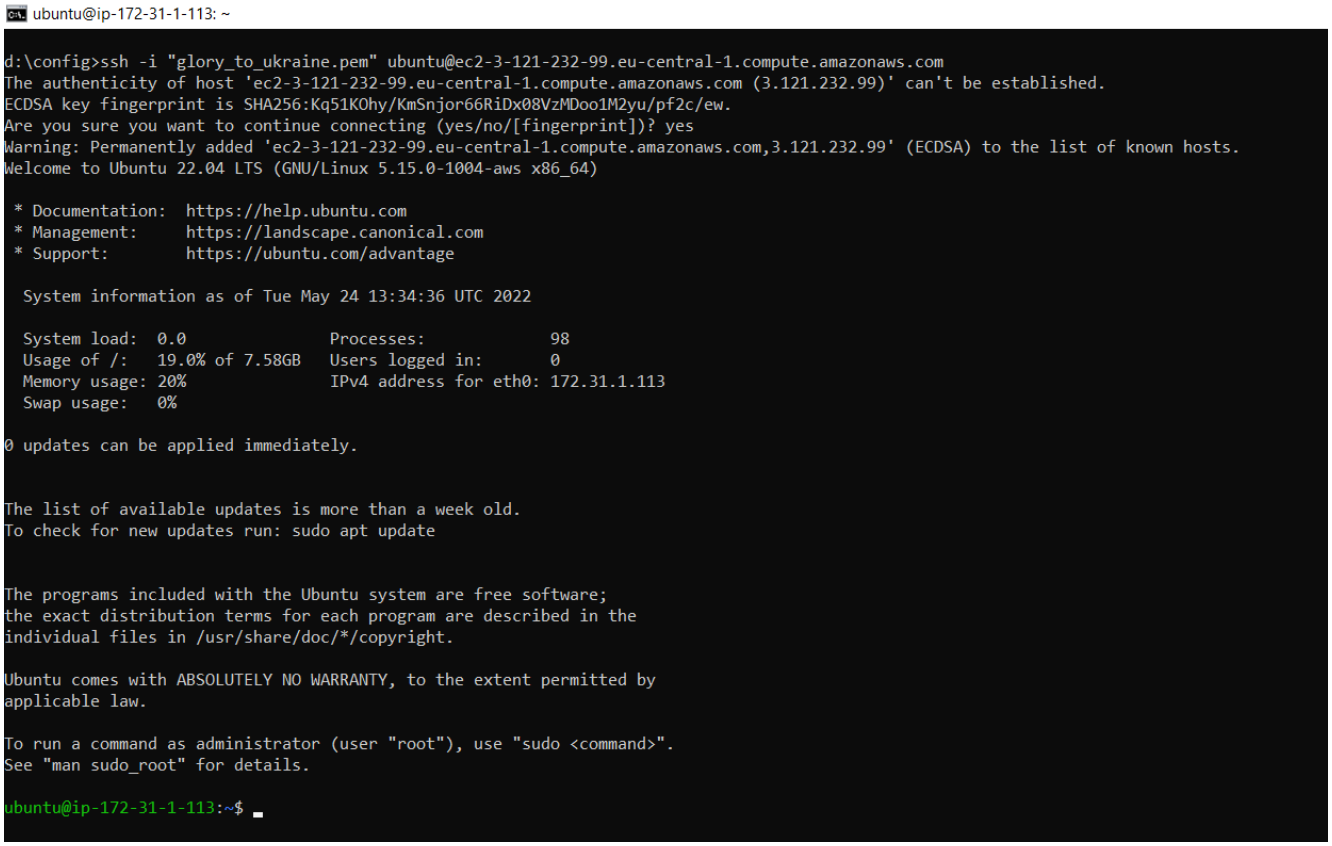


Рисунок 3.17 — Підключення до серверу за допомогою ssh

Після того як було встановлено з'єднання з сервером необхідно встановити такий інструмент для контейнеризації як Docker та Docker Compose. Docker дає можливість запускати будь-які контейнери на будь-якому пристрої де він встановлений, а Docker Compose дозволяє створювати файли зі скриптами в форматі yaml, для того щоб можна було за допомогою однієї команди запускати

набір контейнерів з наперед заданими конфігураціями і так само однією командою їх усі зупиняти.

Для того щоб встановити Docker необхідно за допомогою утиліти curl завантажити з офіційного сайту файл зі скриптом який дозволить швидко його встановити. На рисунку 3.18 зображено команду яка завантажує даний скрипт.

```
ubuntu@ip-172-31-1-113:~$ curl -fsSL https://get.docker.com -o get-docker.sh
ubuntu@ip-172-31-1-113:~$ ls -la
total 48
drwxr-x--- 4 ubuntu ubuntu  4096 May 24 13:38 .
drwxr-xr-x 3 root   root    4096 May 24 13:15 ..
-rw-r--r-- 1 ubuntu ubuntu   220 Jan  6 16:23 .bash_logout
-rw-r--r-- 1 ubuntu ubuntu  3771 Jan  6 16:23 .bashrc
drwx------ 2 ubuntu ubuntu  4096 May 24 13:34 .cache
-rw-r--r-- 1 ubuntu ubuntu   807 Jan  6 16:23 .profile
drwx------ 2 ubuntu ubuntu  4096 May 24 13:15 .ssh
-rw-rw-r-- 1 ubuntu ubuntu 19305 May 24 13:38 get-docker.sh
ubuntu@ip-172-31-1-113:~$
```

Рисунок 3.18 — Завантаження скрипта для встановлення Docker

Далі необхідно виконати команду: sudo sh get-docker.sh. Для того щоб встановити Docker compose необхідно виконати аналогічну послідовність дій.

Після того як було зроблено базові налаштування серверу необхідно налаштувати CI/CD процеси використовуючи користувацький інтерфейс системи контролю версій Gitlab. Для початку необхідно створити в кореневій папці кожного з проектів файл gitlab-ci.yml. В даному файлі прописуються різні етапи збірки проекту, змінні середовища, а також bash команди які дозволяють виконувати тестування нової версії програмного продукту та найголовніше автоматичного розгорнути все на віддаленому сервері (рис. 3.19).

Для даного проекту було створено один єдиний етап збірки під назвою build. Цей етап збірки відповідає за наступні операції:

— створення контейнеру з новою версією програмного продукту;

— розміщення нового контейнеру в приватний репозиторій який знаходиться в docker-hub;

— підключення до віддаленого серверу EC2;

— завантаження контейнеру на сервер;

— запуск контейнеру з новою версією програмного продукту.

```
build:
  stage: build
  before_script:
    - export IMAGE=$CI_REGISTRY/$CI_PROJECT_NAMESPACE/$CI_PROJECT_NAME
    - export WEB_IMAGE=$IMAGE:web
    - export NGINX_IMAGE=$IMAGE:nginx
  script:
    - apk add --no-cache bash
    - chmod +x ./setup_env.sh
    - bash ./setup_env.sh
    - docker login -u $CI_REGISTRY_USER -p $CI_JOB_TOKEN $CI_REGISTRY
    - docker pull $IMAGE:web || true
    - docker pull $IMAGE:nginx || true
    - docker-compose -f docker-compose.ci.yml build
    - docker push $IMAGE:web
    - docker push $IMAGE:nginx
```

Рисунок 3.19 — Файл з конфігурацією етапу build

Для того щоб налаштувати змінні середовища необхідно в інтерфейсі Gitlab відкрити розділ з налаштуваннями, далі перейти в підрозділ з назвою CI/CD та відкрити в ньому секцію зі змінними середовища (рис. 3.20). На даний момент дозволяє створювати до двохсот різних змінних середовища чого повинно бути цілком достатньо для будь-якого малого та середнього проектів. При створенні змінних середовища існують наступні можливості та обмеження:

— Назва ключа змінної повинна бути рядком який не містить жодних переносів та пробілів і складатись виключно з латинських літер та арабських цифр, а також в назві ключа дозволяється використання знаку нижнього підкреслення.

— Для значення ключа немає жодних обмежень за виключенням того .що він має бути в форматі рядка або файлу.

— Для змінної можна обмежити зону видимості одним з існуючих середовищ або залишити цей параметр за замовчування і тоді змінна буде доступною для будь-якого з них.

— Можливо обмежити використання змінної виключно гілками які є захищеними або такими до яких застосовано певний protected tag. Також існує можливість ввімкнути приховування змінних середовища у журналі який пишеться під час виконання кожної CI/CD задачі.

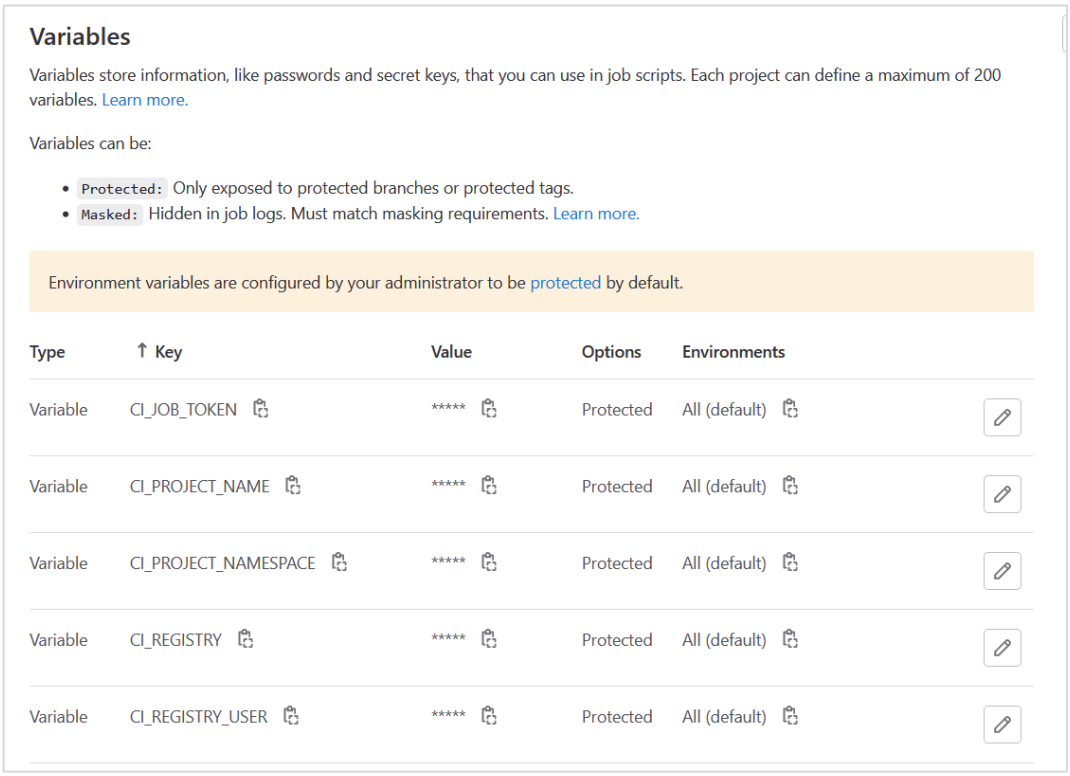


Рисунок 3.20 — Змінні середовища проекту.

Після того як CI/CD процеси були налаштовані необхідно встановити для серверу статичну IP адресу. Для цього необхідно перейти до розділу послуг EC2, далі необхідно перейти до підрозділу з налаштуваннями безпеки та мережного підключення та відкрити вкладку під назвою Elastic IP, після цього необхідно розпочати діалог створення адреси (рис. 3.21). Створення статичної IP адреси є дуже простим оскільки необхідно лише вибрати джерело з якого буде братися адреса. На даний момент можна обрати два варіанти: перший це випадкова вільна

адреса з множини тих якими володіє AWS, а другий це одна з адрес якими ви володієте та підключили до свого облікового запису в AWS.

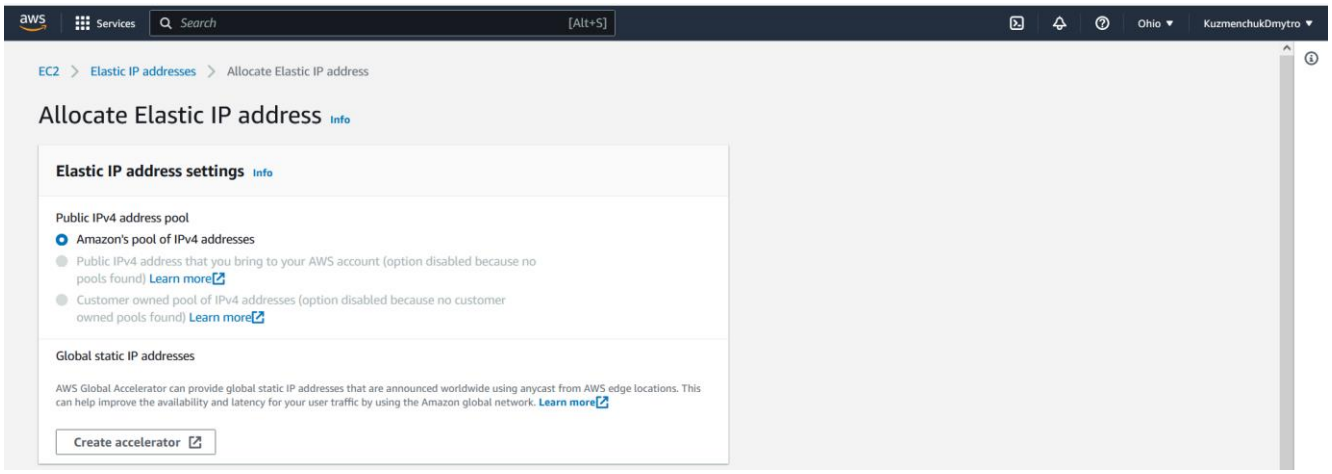


Рисунок 3.21 — Створення статичної IP адреси

Для того щоб прив’язати щойно отриману адресу до свого серверу необхідно відкрити панель зі списком ваших адрес, обрати одну з них та у списку доступних дій обрати прив’язку до серверу. Після цього відкриється діалогове вікно де потрібно буде обрати сам сервер.

Для того щоб налаштувати автоматичне створення серверів перш за все необхідно створити шаблон з конфігурацією за якою будуть створюватись нові сервери. При створенні шаблону необхідно вказати ті самі параметри що вказуються при створенні одного окремого сервісу, а також можна окремо налаштувати процес видалення серверів, використання сховища, оперативної пам’яті та іншого.

Після того як було створено шаблон для створення нових серверів необхідно встановити умови при якому навантаженні потрібно слід створювати нові та при якому вимикати існуючі сервери.

На рисунку 3.22 зображено налаштування Auto Scaling triggers на основі вихідного мережного трафіку. При налаштуванні можна вказати наступні умови:

— величина на основі якої зменшується та збільшується кількість серверів;

— метрика за якою встановлюють часові та кількісні обмеження (наприклад середнє значення);

— одиниця вимірів для обраної величини;

— період часу між повторними обчисленнями обраної метрик;

— максимальний час перевищення лімітів метрики;

— граничні межі навантаження для збільшення та зменшення кількості серверів;

— кількість серверів які створюються або зупиняються коли навантаження перевищує верхні межі або менше за нижні.

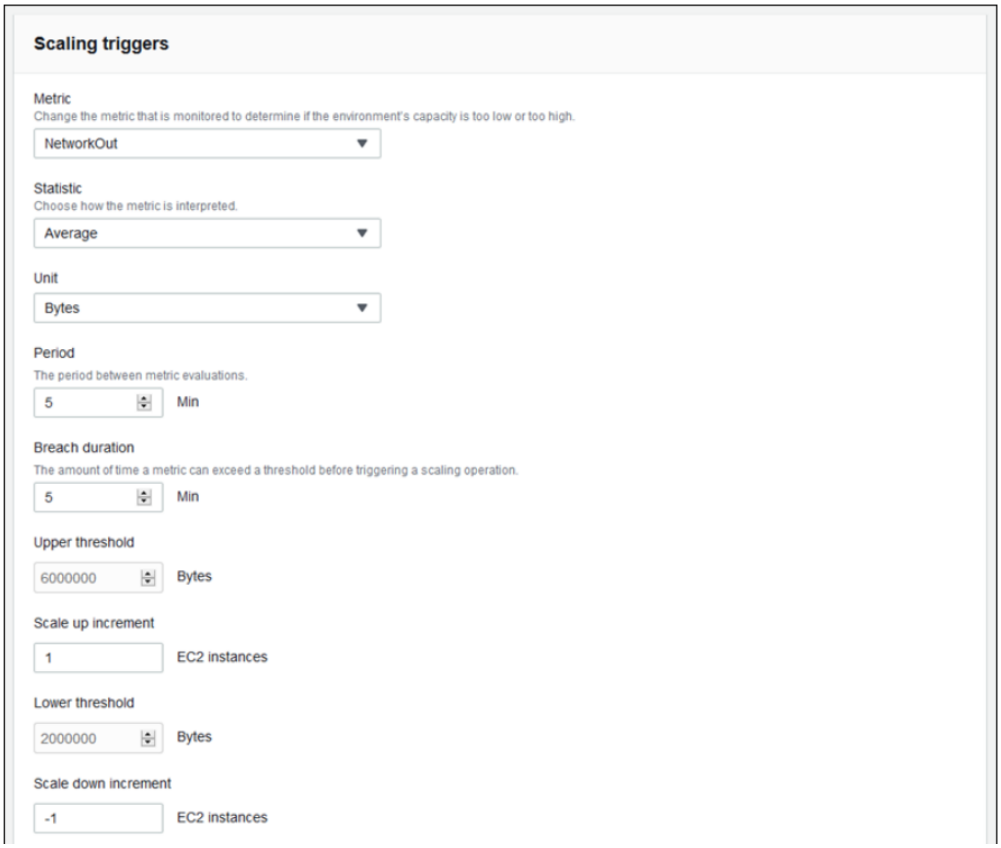


Рисунок 3.22 — Налаштування Auto Scaling triggers

Для того щоб збалансувати навантаження між всіма серверами необхідно налаштувати такий сервіс як AWS Load Balancer. Для цього необхідно:

— Обрати версію протоколу IP, трафік з яких регіонів має приймати балансир та список протоколів з яких приймати трафік.

— Налаштувати security group де можна обмежити вхідний та вихідний трафік за набором IP адрес.

— Обрати цільову групу та налаштувати параметри перевірки працездатності цільових серверів.

— Вказати набір серверів між якими необхідно балансувати навантаження (рис. 3.23).

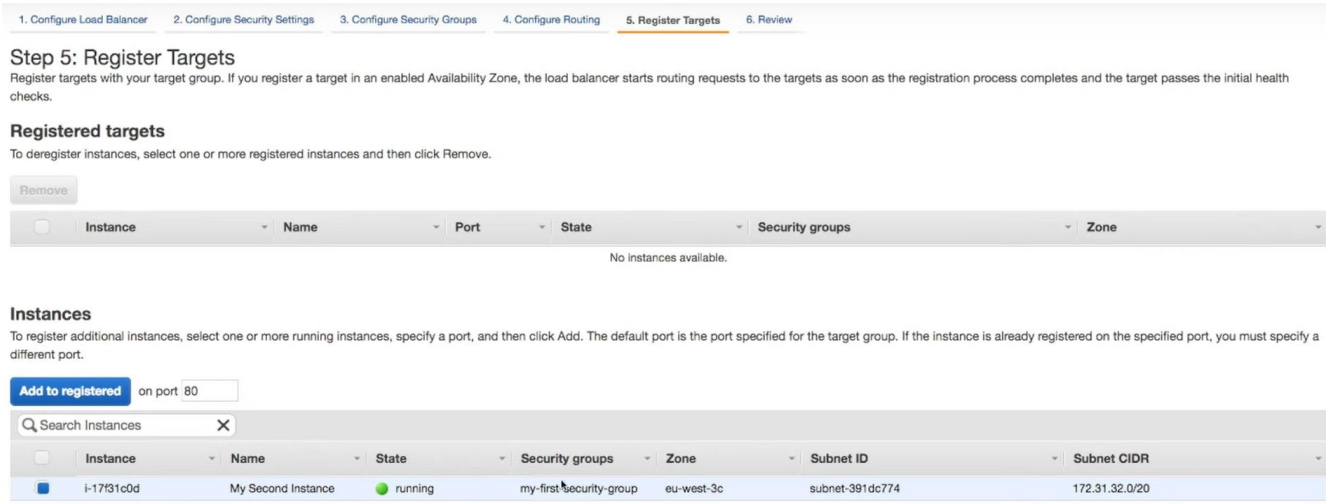


Рисунок 3.23 — Налаштування балансира навантаження

Для того щоб AWS Load Balancer приймав трафік який надсилається на певний домен необхідно провести певні налаштування в сервісі Route 53. Для налаштування необхідно виконати наступне:

— створити hosted zone;

— створити новий запис в Route 53;

— обрати регіони для яких будуть діяти правила;

— вказати один з балансирів навантаження.

З метою моніторингу навантаження на сервери системи можна скористатись сервісом CloudWatch (рис. 3.24). Даний сервіс дозволяє відслідковувати багато

різних метрик та налаштовувати персональні панелі з тими метриками які потрібні вам в даний момент часу.

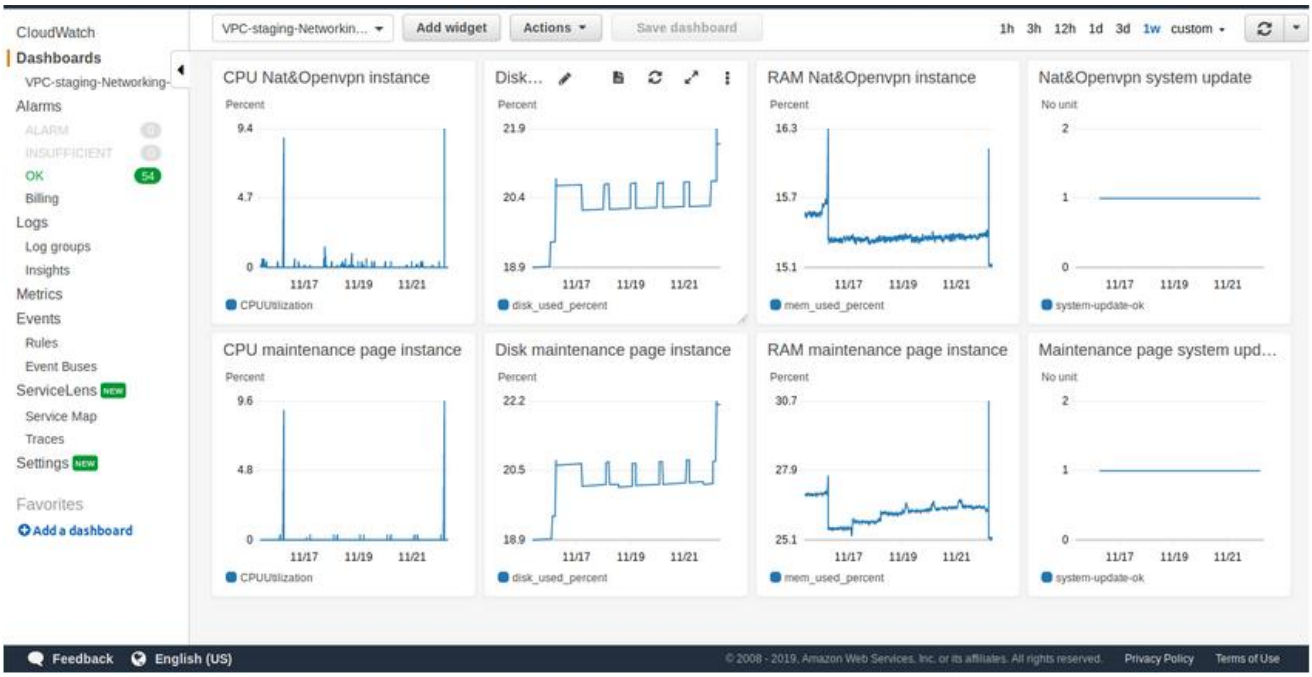


Рисунок 3.24 — Приклад персонально налаштованої панелі з метриками

На даний момент можливо використання наступних метрик:

— CPUUtilization: дана метрика відображає середню завантаженість центрального процесору протягом певного відрізку в часі.

— DiskReadOps: дана метрика відображає частоту операції зчитування інформації з диску протягом певного відрізку часу.

— DiskWriteOps: дана метрика відображає частоту операцій запису інформації на диск протягом певного періоду часу.

— DiskReadBytes: дана метрика показує в байтах інтенсивність зчитування інформації з диску серверу протягом певного періоду часу.

— DiskWriteBytes: дана метрика показує в байтах інтенсивність запису інформації на диск серверу протягом певного періоду часу.

— NetworkIn: дана метрика показує об’єми вхідного трафіку протягом певного періоду часу

— NetworkOut: дана метрика показує об’єми вихідного трафіку протягом певного періоду часу.

— CPUCreditUsage: дана метрика відображає кількість кредитів процесора кожен з яких еквівалентний роботі яку виконує одне ядро протягом однієї хвилини за максимального навантаження.

— CPUCreditBalance: дана метрика показує сумарну кількість кредитів процесора яка була нарахована починаючи з моменту запуску серверу.

Висновки до розділу 3

В ході виконання роботи було розроблено алгоритм підготовки текстових даних перед їх використанням для навчанні моделі класифікації фейкових новин. Також було проаналізовано способи реалізації різних методів машинного навчання для виконання поставленої задачі, а також їх переваги та недоліки. В кінці даного розділу описується архітектура створеної інфраструктури системи та послідовність дій, що необхідна для її розгортання та налаштування.

4 ОПИС ПРОГРАМНОЇ СИСТЕМИ

Розроблена система не потребує жодної інсталяції оскільки вона була розгорнута в хмарному сервісі AWS (Amazon Web Services), тому все що потрібно кінцевому користувачу це:

— персональний комп’ютер;

— доступ до мережі інтернет.

— один з браузерів;

Також є можливість взаємодіяти з системою використовуючи відкрите API яке має такий самий функціонал як і користувацький інтерфейс.

4.1 Опис інтерфейсу головної сторінки

На рисунку 5.1 зображено головну сторінку системи. Дана сторінка складається з наступних структурних елементів:

— Панель навігації яка містить в собі посилання для переходу між основною сторінкою та сторінкою яка описує правила для взаємодії з системою за допомогою API. Також в правому куті є можливість зміни мови інтерфейсу. В поточній версії системи є можливість вибору між українською та англійською мовами.

— В центральній частині знаходиться вибір типу введення новини для перевірки. На даний момент є можливість перевірки однієї новини в разі якщо було вибрано текстовий формат вводу, а також перевірка набору новин якщо було обрано текстовий формат вводу.

— Для завантаження файлу з даними для перевірки реалізовано принцип drag and drop за допомогою якого бажаний файл можна просто перетягнути з провідника у відведену для цього область. Також є можливість обрати файл для завантаження

натиснувши на область для завантаження після чого має відкритись діалогове вікно з провідником для вибору файлу

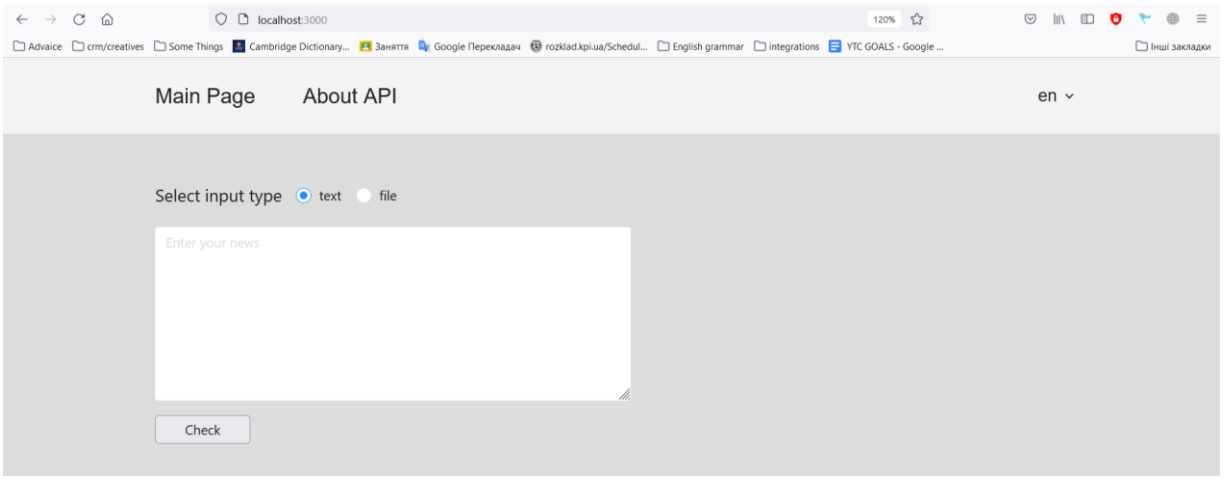


Рисунок 5.1 — Інтерфейс головної сторінки системи

На рисунку 5.2 зображено як виглядає інтерфейс для завантаження файлів для перевірки.

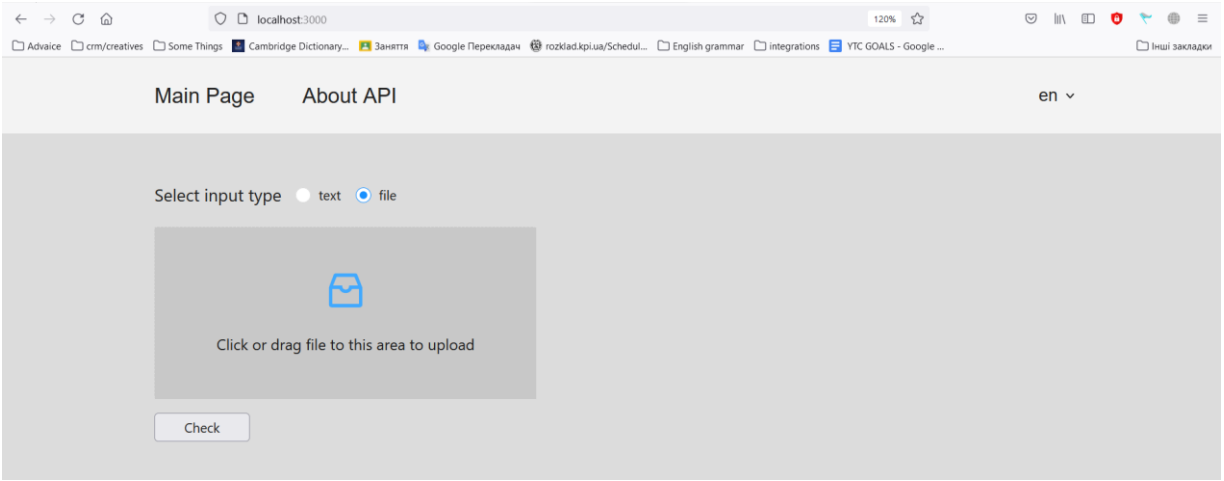


Рисунок 5.2 — Інтерфейс для завантаження файлів

Завантажений файл повинен бути в форматі csv та містити новини в колонці під назвою text. Після того як файл було завантажено зверху з’являється

сповіщення яке говорить про те що файл було завантажено успішно або пише про помилку у разі якщо щось пішло не так (рис. 5.3).

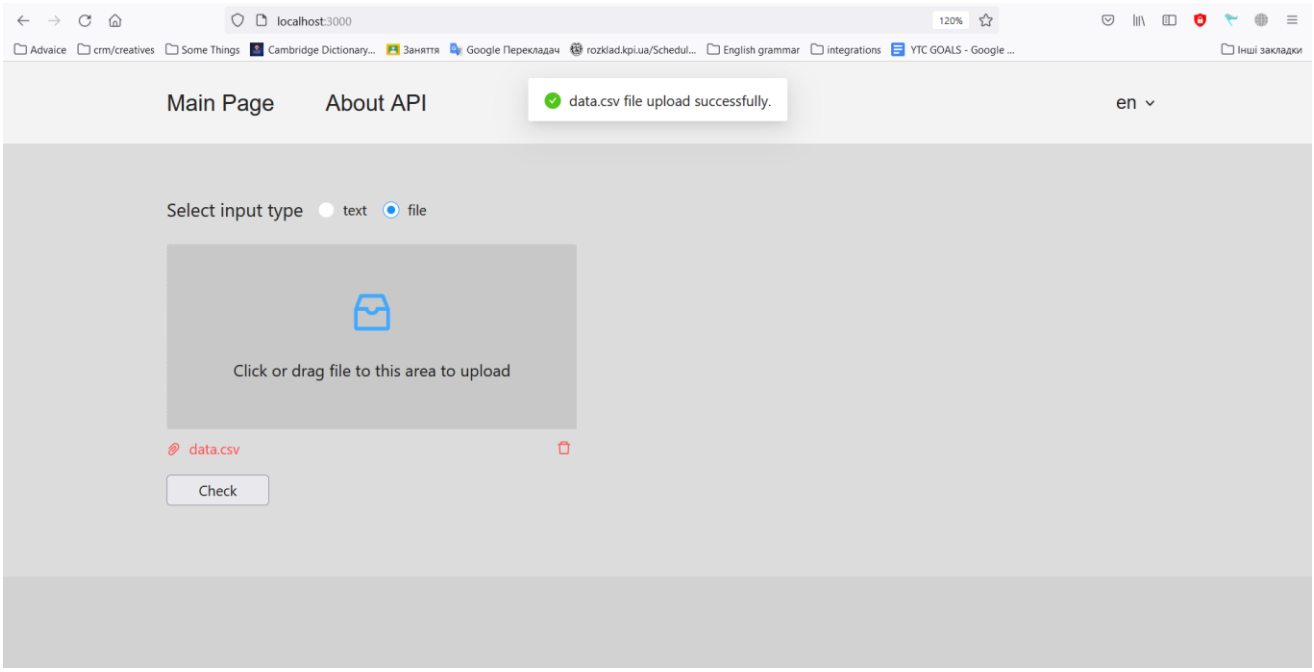


Рисунок 5.3 — Приклад успішного завантаження файлу

Після того як файл було завантажено його можна змінити натиснувши на кнопку з іконкою корзини для видалення поточного файлу після чого можна завантажити новий.

4.2 Опис Інтерфейсу сторінки про API системи

Дана сторінка містить відомості про те як взаємодіяти з системою за допомогою API, а саме:

— загальний опис та призначення ендпоінта;

— HTTP метод;

— url адреса куди необхідно надсилати запит;

— опис всіх параметрів;

— опис структури відповіді від серверу.

На рисунку 5.4 зображено документацію для запиту який використовується для класифікації однієї новини. Цей ендпоінт містить один обов’язковий параметр під назвою news. Даний параметр має текстовий формат та повинен містити текст новини для перевірки. Для даного запиту необхідно використовувати HTTP метод POST. Після обробки сервер повертає текст самої новини а також результат класифікації з ключом prediction та булевим значенням. У випадку якщо новина була класифікована як така що є достовірною то сервер повертає значення true в противному випадку повертається значення false.

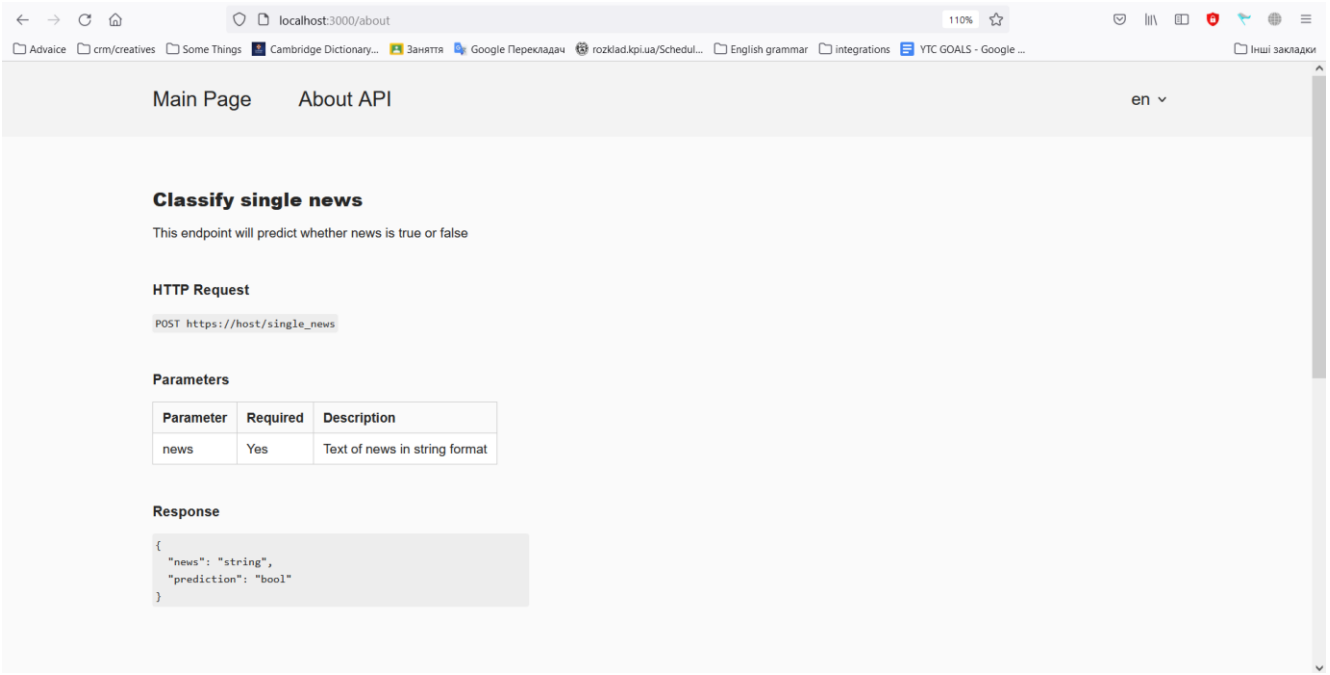


Рисунок 5.4 — Документація запиту для класифікації однієї новини

На рисунку 5.5 зображено документація для запиту який використовується для класифікації набору новин. Для цього запиту як і для попереднього використовується HTTP метод POST. Даний ендпоінт має один обов’язковий параметр з назвою file. В цьому файлі мають міститись новини для класифікації. Новини повинні бути в одній колонці з назвою text. Кожна з новин повинна бути розміщена в окремій клітинці. У разі якщо переданий файл мав правильний формат

то сервер повертає у відповідь файл з початковими даними та результатами класифікації.

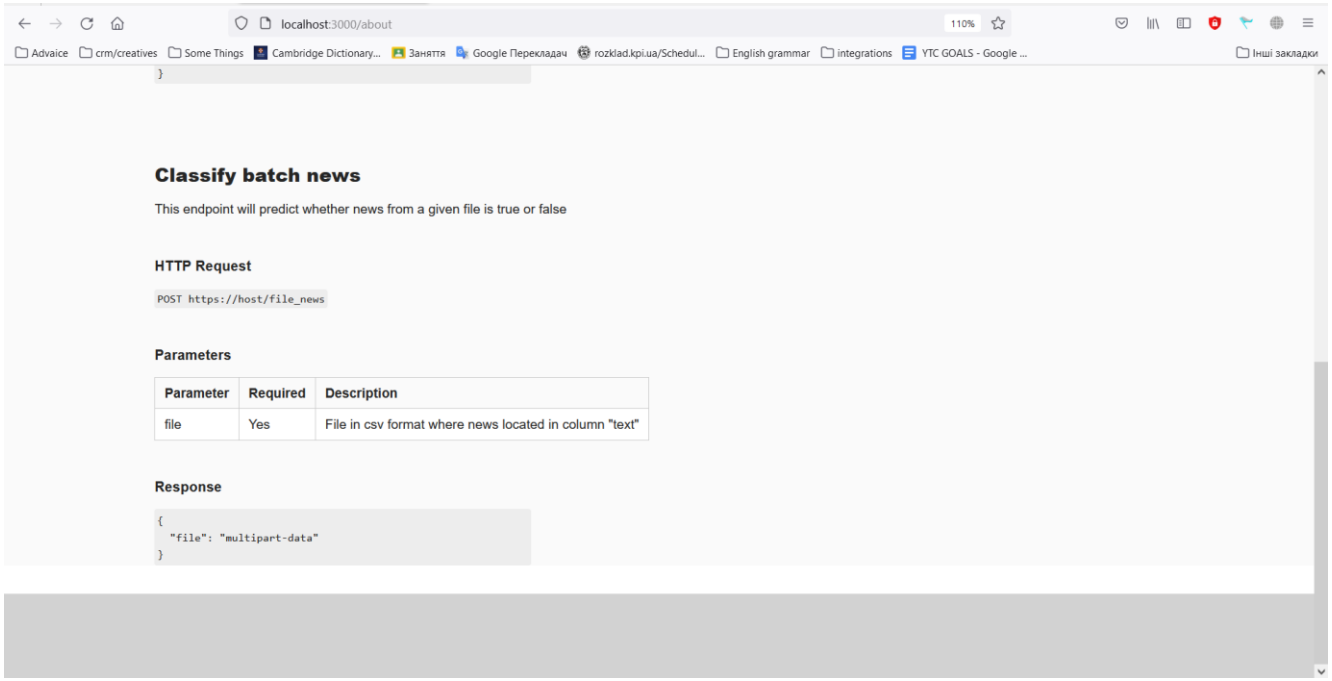


Рисунок 5.5 — Документація запиту для класифікації набору новин

Результати класифікації поміщаються в колонку з назвою prediction в якій біля кожної з вхідних новин знаходиться булеве значення яке слід інтерпретувати так само як і результати які повертає ендпоінт для класифікації однієї новини.

Висновки до розділу 4

В ході виконання роботи було створено програмний продукт, що дозволяє будь-кому перевірити новину чи набір новин на їх достовірність незалежно від того чи це проста людина чи то розробник який хоче автоматизувати процес перевірки новин які публікують користувачі в його інформаційній системі.

5 РОЗРОБКА СТАРТАПУ ПРОЄКТА

В зв’язку з актуальністю теми боротьби з фейками в сучасних інформаційних системах та потенційно великим числом можливих клієнтів та інвестицій було у відповідності до теми магістерської дисертації було розроблено стартап проєкт в якому описані всі кроки для створення програмного продукту, прораховано витрати на його розробку, підтримку та маркетинг, а також визначено цільову аудиторію та можливі ризики.

5.1 Етапи реалізації стартап проекту

Для того щоб реалізувати даний стартап проєкт було виділено такі основні кроки:

— Визначити актуальність проблеми та можливий попит на майбутній програмний продукт.

— Проаналізувати можливості систем конкурентів та виділити ті аспекти які були погано реалізовані або не були реалізовані взагалі для того щоб в подальшому створити програмний продукт який буде виділятись на загальному фоні та мати комерційний потенціал.

— Зібрати за мінімальний бюджет датасет з достовірних та фейкових новин який буде містити як мінімум 100000 елементів.

— Розробити архітектуру системи для навчання моделі яка буде класифікувати фейкові новини.

— Створити модель інфраструктури використовуючи хмарні сервіси такі як AWS або GCP яка буде вимагати мінімального об’єму фінансів, ефективно розподіляти навантаження між різними серверами та відповідно до поточного

навантаження на систему збільшувати або зменшувати кількість серверів для мінімізації витрат.

— Проаналізувати ризики які можуть виникнути в ході реалізації стартапу та створити стратегію для їх мінімізації.

— Визначити цільову аудиторію майбутнього програмного продукту, маркетингову стратегію для його просування на ринку, платформи для реклами та оптимальну ціну за користування сервісом при якій стартап буде конкурентноздатним та зможе отримати максимальну кількість клієнтів.

— На основі всіх вище перелічених кроків зробити прогноз рівня рентабельності стартап проєкту.

5.2 Аналіз поточної ситуації на ринку

В поточний момент часу системи для визначення фейків представлені тільки однією системою Discourse Processing Lab. З цього можна зробити наступні припущення:

— Ця ніша досі не була занята іншими успішними проектами та існує можливість зайняти певну частину ринку.

— Існує можливість що успішні системи існують але їх використовують виключно для внутрішнього використання та не бажають відкривати для широкого загалу.

— Нікому не вдалося розробити систему яка виявляє фейки з достатньою точністю при якій стало б можливим знайти клієнтів та отримувати дохід за такий сервіс.

На рисунку 5.1 зображено фрагмент інтерфейсу вище згаданої системи для класифікації фейків на якому зображено те як можна класифікувати одну новину яку було введено в текстовому форматі. Все що потрібно для цього зробити це обрати спосіб (виявлення фейків), ввести новину та натиснути кнопку відправити

після чого з’являється повідомлення яке говорить про те чи новина базується на фактах чи є фейковою.

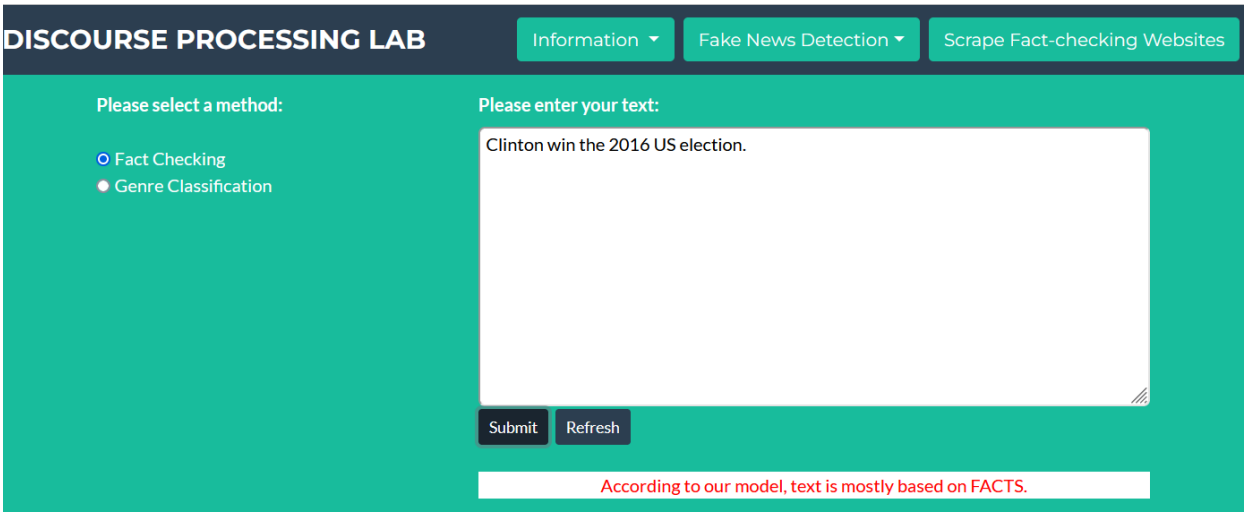


Рисунок 5.1 — Приклад функціоналу системи Discourse Processing Lab

Також існує можливість в автоматичному режимі отримати всі новини з фіксованого переліку сайтів для яких були створені скрапери (рис. 5.2). В поточний момент часу можливо завантажити тільки весь архів даних.

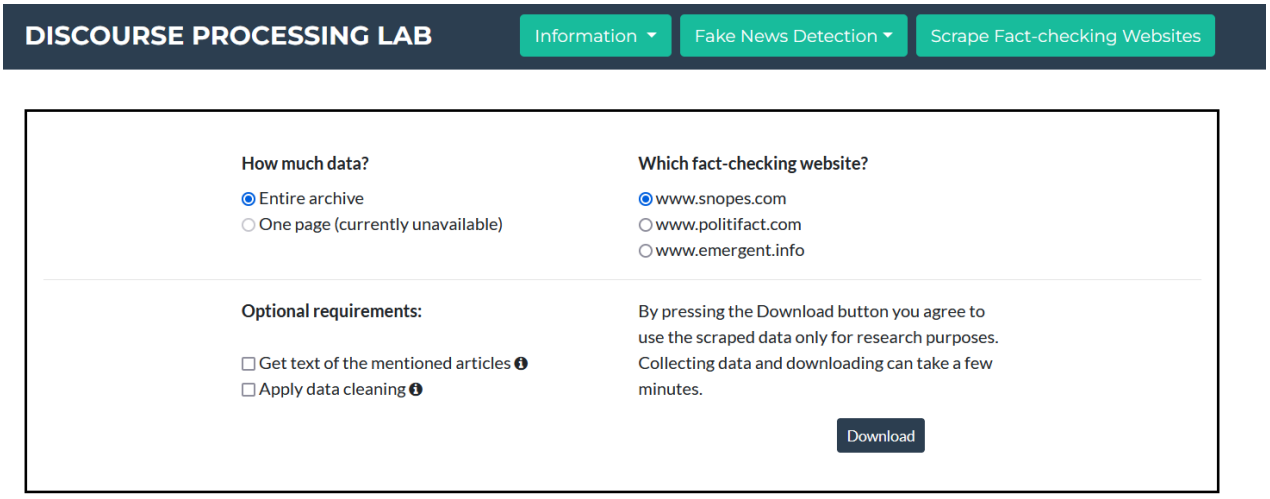


Рисунок 5.2 — Приклад інтерфейсу для скрапінгу

Також слід зазначити, що на даний момент функція для аналізу та отримання даних з певної веб сторінки недоступна.

Враховуючи всі можливості розглянутої вище системи можна виділити наступні переваги:

— Система функціонує у вигляді веб сервісу тому нею може скористатись будь-хто без необхідності займатись якимось налаштуваннями та не вимагає спеціальних знань.

— Система має достатньо непоганий дизайн та зрозумілий користувацький інтерфейс яким легко користуватись, а також наявна навігація за допомогою якої можна знайти будь-яку функцію.

— Існує можливість використання скраперів які аналізують вміст веб сторінок та завантажують з них усі новини які потім можна використовувати для тренування моделей які визначають фейки.

До недоліків даної системи можна віднести наступне:

— Немає можливості користуватись функціоналом використовуючи API тому стає неможливим створення інтеграції між даною системою та будь-якою іншою.

— Немає можливості класифікувати відразу декілька новин тому дану систему неможливо застосовувати у великих масштабах коли необхідно перевіряти тисячі або десятки тисяч новин.

— Можливості скраперів обмежені фіксованим списком який складається з трьох сайтів.

— Не працює можливість для завантаження новин з однієї вибраної певної сторінки.

5.3 Аналіз цільової аудиторії стартапу

Даний програмний продукт може бути цікавим для різного роду соціальних мереж в яких основною діяльністю користувачів є обмін різною інформацією, що відкриває широкі можливості для розповсюдження фейків з використанням

автоматизованих програм які називають ботами та звичайних людей які можуть це робити за невелику плату. Ще одним потенційним споживачем даного програмного продукту можуть бути новинні ресурси які публікують інформацію від неперевірених осіб або роблять це без попереднього жорсткого рецензування. Також є варіант застосування даної системи в сервісах таких як YouTube де можна викладати відео оскільки в коментарях під ними часто ведуться запеклі дискусії в яких беруть участь в тому числі і боти і люди які розповсюджують фейки.

5.4 Аналіз можливих ризиків

Розробку системи для розпізнавання фейків можна впевнено назвати інноваційною діяльністю у порівнянні з іншими стартапами. Через це на фінальний результат впливає велика кількість ризиків і ні про які гарантії успіху не може йти мови. За статистикою лише 10-20% стартапів досягають успіхів та приносять їх розробникам доходи, але не зважаючи на високі ризики інноваційні стартапи у разі успіху можуть принести доходи які перевищують суму інвестицій на розробку в десятки чи сотні разів.

На рисунку 5.3 зображено ризики мікросередовища, що можуть чинити значний вплив на стартап. Дані ризики можна розподілити на декілька категорій:

— Організаційні ризики включають в себе погане керування окремими підрозділами які беруть участь в розробці, погану взаємодію між командами розробників, неправильну постановку задач. Всі ці помилки можуть приводити до порушення термінів виконання робіт, конфліктів в середині команди та між різними командами а найголовніше бути причиною поганої якості фінального програмного продукту. Для мінімізації даних ризиків потрібно ретельно проводити планування етапів розробки програмного продукту, відбирати на керівні посади компетентних людей які мають досвід в цій справі та слідкувати за тим щоб в

середині команд та між різними командами панувала здорова атмосфера, а також проводити заходи які цьому сприяють.

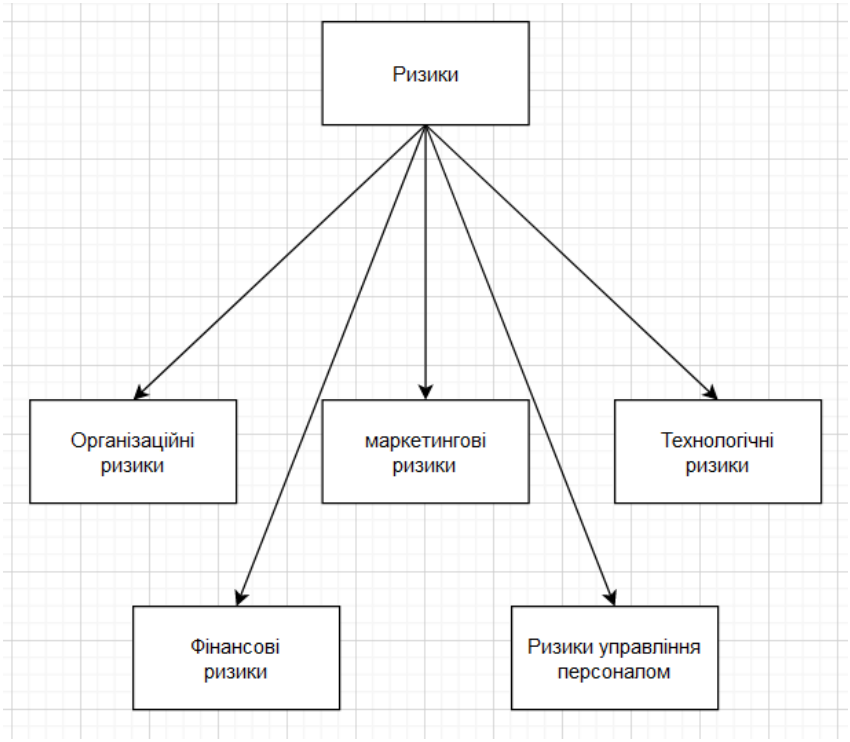


Рисунок 5.3 — Види ризиків мікросередовища стартапу

— Маркетингові ризики включають в себе неправильну оцінку цільової аудиторії програмного продукту, неправильний вибір рекламних платформ для розповсюдження реклами та недостатня якість самих рекламних матеріалів. Дані помилки з високою ймовірність призводять до того більша частина коштів буде витрачена даремно, а погана якість рекламних матеріалів може навіть викликати негативний ефект і зменшити кількість клієнтів. Для зменшення даних ризиків необхідно прикласти зусилля для визначення цільової аудиторії та рекламних платформ які найбільше її охоплюють, а також не потрібно надмірно економити на бюджеті самих рекламних матеріалів щоб не викликати негативних наслідків.

— Технологічні ризики полягають в неправильному виборі технологій для розробки програмного продукту а також в постановці задач які неможливо вирішити з поточним рівнем науково-технічного прогресу. Помилки у виборі стеку

технологій для розробки можуть призводити до збільшення часу розробки програмного продукту і як результат до збільшення витрат. У найгіршому випадку може виявитись, що з обраними технологіями неможливо створити продукт який буде відповідати всім вимогам після чого існує два варіанти: перший це починати розробку заново використовуючи новий стек технологій у разі якщо ще залишилися кошти, а другий це згортати розробку стартапу. Для того щоб мінімізувати дані ризики необхідно мати досвідчених розробників які зможуть обрати правильні технології, а також оцінити чи можливо взагалі розробити програмний продукт з бажаними характеристиками.

— Фінансові ризики полягають в тому, що в ході реалізації стартапу можуть виникнути непередбачувані статті витрат або заплановані етапи розробки можуть виявитись дорожчими ніж планувалося. Для того щоб мінімізувати дані ризики необхідно мати значні запаси фінансів для того щоб не збанкрутувати та не зупинитись на середині шляху, а також необхідно все ретельно планувати для того щоб уникнути непередбачуваних витрат.

— Ризики управління персоналом полягають в тому, що працівники внаслідок низького рівня кваліфікації можуть бути неспроможні самостійно виконати поставлені перед ними задачі. Для мінімізації таких ризиків необхідно проводити жорсткий відбір працівників перед прийомом на роботу, а також видавати завдання працівникам відповідно до їх рівня кваліфікації.

5.5 Прогнозування витрат на розробку програмного продукту

При розробці будь-якого програмного продукту необхідні витрати для наступних речей:

— Серверної частини, що відповідає за бізнес логіку та обробляє всі запити які надходять від користувачів системи з фронтенду.

— Розгортання та налаштування інфраструктури, що включає в себе створення серверів на яких будуть працювати серверна частина та користувацький інтерфейс, створення баз даних, налаштування SSL сертифікатів, доменів та мережевого доступу загалом.

— Розробка дизайну користувацького інтерфейсу який буде зручним у користуванні та буде гарно виглядати.

— Розробка користувацького інтерфейсу відповідно до створених дизайнерами макетів.

— Тестування системи для виявлення дефектів та багів які можуть робити частину функціоналу непридатним для користуванням або робити досвід від користування системою не таким позитивним.

— Виправлення перед релізом всіх знайдених в процесі тестування дефектів та багів.

Так як даний стартап зосереджений на застосуванні штучного інтелекту для вирішення проблеми автоматизованого визначення фейкових новин то виникають додаткові витрати, а саме:

— Витрати на створення якісного датасету який дозволить досягти високої точності моделі при виконанні поставленої перед нею задачі.

— Витрати на створення архітектури для підсистеми яка виконує роль підготовки даних та самої моделі яка відповідає за класифікацію, а також витрати на обчислювальні потужності які виникають в процесі навчання моделі.

На рисунку 5.4 зображено оціночний час який необхідний на виконання кожного з етапів розробки:

— Для розробки серверної частини необхідно приблизно 20-30 годин оскільки система не передбачає складної бізнес логіки, а основна частина роботи зосереджена на штучному інтелекті.

— Для збору датасету зі ста тисяч елементів можна скористатись готовими безкоштовними рішеннями з відкритих джерел, купити датасет чи найняти людей які розмітять датасет. Якщо скористатись останнім варіантом то це буде приблизно

коштувати 2-3 центи за кожну новину що сумарно буде вартувати від двох до трьох тисяч доларів.

Рисунок 5.4 — Час для реалізації етапів розробки

— Для створення моделі для класифікації фейків необхідно найбільше часу оскільки неможливо відразу обрати найоптимальніший метод для класифікації та вхідні дані які буде використовувати модель тому необхідно проводити багато експериментів для того щоб знайти найкращий варіант. Через це на виконання даного етапу необхідно приблизно 80-100 годин.

— Для розгортання інфраструктури необхідно приблизно 40-50 годин. Дана задача є дуже важливою оскільки за допомогою гарно написаних скриптів можна розгортати, згортати та модифікувати інфраструктуру за лічені хвилини що значно економить кошти при розробці та полегшує життя розробникам.

— Для розробки дизайну користувацького інтерфейсу необхідно приблизно 15-20 годин оскільки початкова версія системи буде складатись лише з двох сторінок: головної яка використовується для класифікації новин та сторінки з документацією про те як можна взаємодіяти з системою за допомогою API.

— Для розробки користувацького інтерфейсу необхідно приблизно 20-25 годин.

— Для тестування початкової версії системи необхідно 10-15 годин оскільки система буде мати не так багато функціоналу.

— Для виправлення дефектів та багів у разі якщо не було допущено серйозних помилок може знадобитись 10-20 годин.

Тож при оптимістичному прогнозі розробка займе 195 годин, в песимістичному 260 годин, а у реалістичному 230 годин

На рисунку 5.5 зображено вартість для виконання кожного з етапів. Для оцінки вартості кожного етапу необхідно визначити погодинну ставку працівників:

— розробники серверної частини застосунку 15$/год;

— розробники з галузі штучного інтелекту 25$/год;

— розробники користувацького інтерфейсу застосунку 12$/год;

— дизайнери користувацького інтерфейсу 20$/год;

— DevOps розробники 20$/год;

— тестери застосунку 8$/год.

Рисунок 5.5 — Оціночна вартість реалізації кожного з етапів

Таким чином мінімальна оціночна вартість розробки системи для автоматичного визначення фейкових новин буде становити 5930 доларів, але це

всього лиш оптимістичний прогноз тому враховуючи всі ризики потрібно мати як мінімум на 50% більше коштів.

5.6 Прогнозування витрат для підтримки інфраструктури

Для створення інфраструктури доцільно буде скористатись хмарними сервісами такими як AWS, GCP або Azure і на це є одразу декілька причин:

— Широкий вибір доступних сервісів які можна легко розгорнути та велика кількість докладної документації.

— Не потрібно витрачатись на дорого вартісне обладнання, його установку та підтримку.

— Можна в будь-який момент перестати користуватись сервісом і припинити платити за нього.

— Можна легко масштабувати потужності системи у разі якщо виникає така необхідність.

— Хмарні сервіси мають більшу надійність та механізми для захисту даних такі як реплікація, що копіюють інформацію в різних датацентрах убезпечуючи дані від природних катастроф які можуть все знищити як це стається коли все зберігається централізовано.

Для даного стартапу краще буде використати сервіси які надаються компанією AWS оскільки вона має наступні переваги:

— величезний вибір сервісів серед яких багато інструментів корпоративного рівня;

— багато можливостей для конфігурування;

— наявність інструментів для моніторингу стану системи та можливість керування доступом та безпекою.

— потужна підтримка екосистеми з боку розробників та величезна спільнота користувачів у якої можна отримати пораду.

Для розгортання серверів з серверною частиною та користувацьким інтерфейсом AWS надає такий сервіс як EC2 (Elastic Compute Cloud). Для того щоб взяти в оренду на місяць сервер з мінімальними для даної системи ресурсами необхідно буде заплатити 74.16 доларів (рис. 5.6).

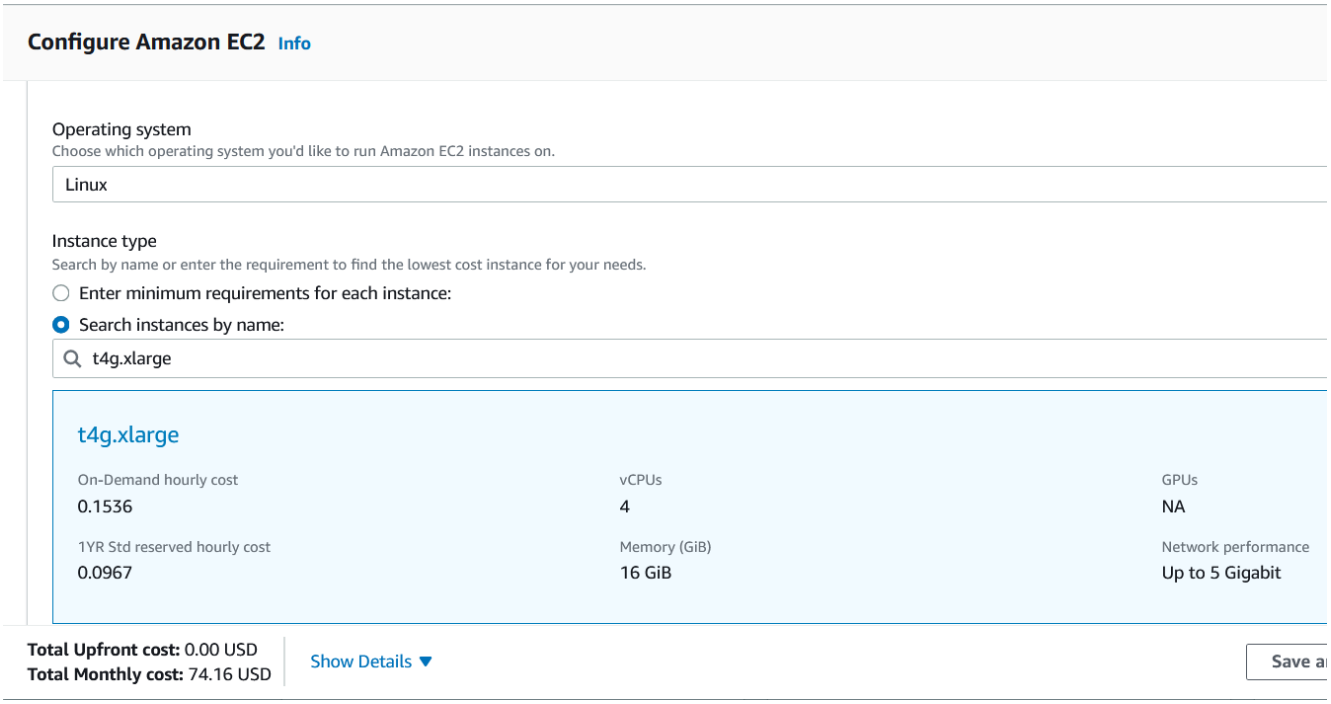


Рисунок 5.6 — Оцінка вартості серверу

Даний сервер буде мати чотири ядра, що дозволить одночасно обробляти достатньо велику кількість запитів, 16 гігабайтів оперативної пам’яті та 30 гігабайтів на диску. В ролі оперативної системи виступає Linux оскільки він надає широкі можливості для автоматизації процесу розгортання системи та налаштування CI/CD процесів.

Для того щоб мати доступ до сервер мав постійну IP адресу можна скористатись сервісом Amazon Elastic IP який дозволяє зарезервувати за своїм обліковим записом певну адресу на потрібний вам час. AWS дозволяє безкоштовно користуватись одною такою адресою прив’язаною до EC2 серверу у разі якщо сервер весь час знаходиться в робочому стані.

У випадку якщо адреса не була підключена до серверу за кожну годину буде стягуватись плата в розмірі 0.01 долару (рис. 5.7). Це було зроблено для того щоб такий обмежений ресурс як IP адреси максимально використовувався і не створювалось штучного дефіциту. Також AWS стягує плату в розмірі 0.1 долару за кожне підключення адреси до іншого серверу.

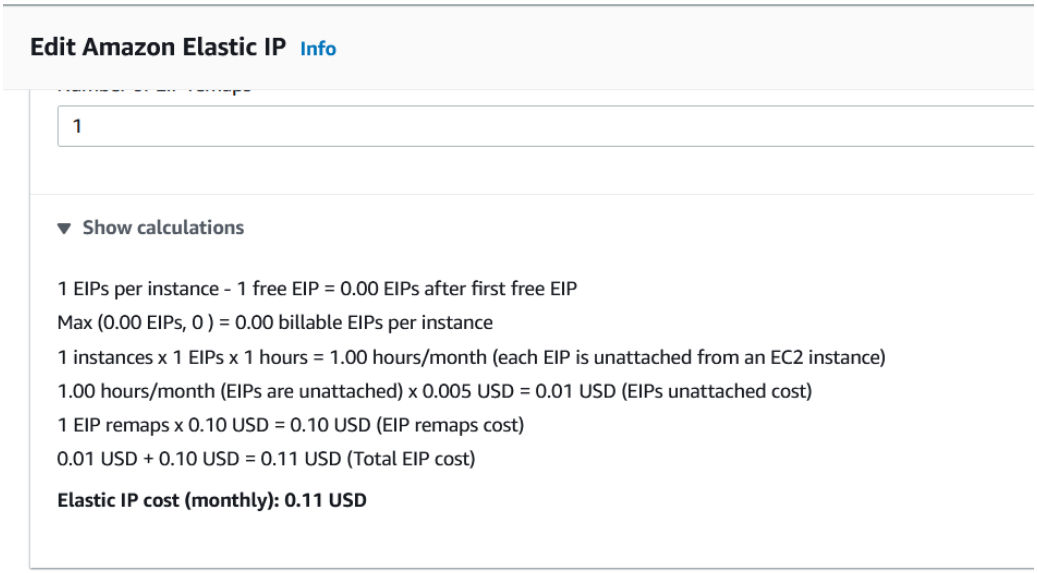


Рисунок 5.7 — Вартість користування сервісом Elastic IP

Таким чином у разі якщо сервер буде неперервно працювати, то за статичну адресу яка надається даним сервісом не потрібно буде сплачувати ніяких коштів, а за перепідключення лише по 10 центів.

Для того щоб покращити якість користування програмним продуктом та зробити бренд впізнаваним важливо обрати правильне доменне ім’я. Використання доменних імен є фактично стандартом оскільки зараз дуже важко побачити сайт при переході на який ви побачите в пошуковому рядку IP адресу замість домену. Використання короткого і милозвучного доменного імені дозволить людям легше знаходити ваш сайт тоді як звичайна IP адреса може викликати недовіру до сервісу та відштовхнути користувачів.

Серед сервісів які надає AWS є такий за допомогою якого можна зареєструвати доменне ім’я. Даний сервіс називається Route 53 та дозволяє обирати

серед десятків доменів верхнього рівня. Для даного стартапу важливо підкреслити що він направлений на боротьбу з фейками тому доменом другого рівня можна обрати щось на кшталт fake-detection тоді як для домену верхнього рівня підійдуть .com або .org які є широко використовуваними та впізнаваними та коштують 11 та 12 доларів за рік відповідно (рис. 5.8).

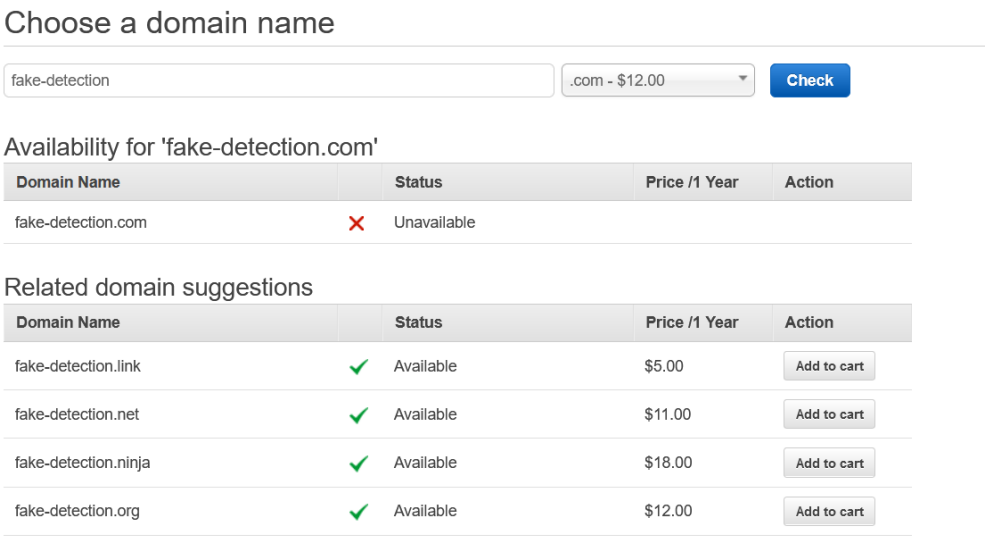


Рисунок 5.8 — Наявні для реєстрації доменні імена

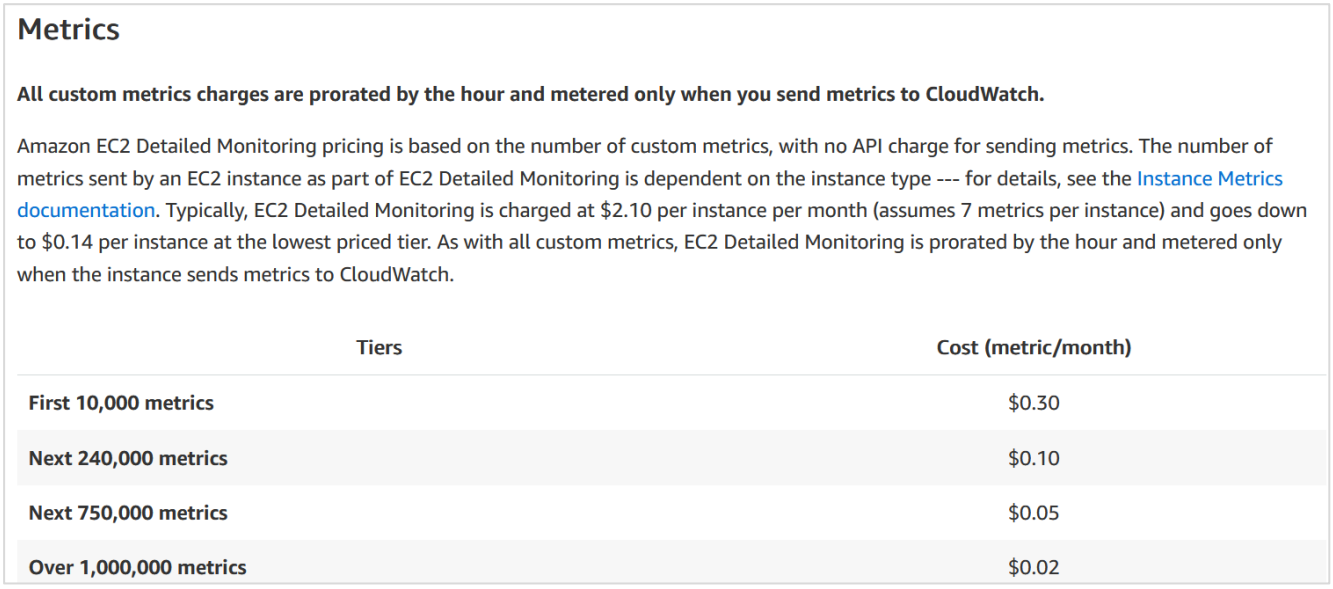
Так як подібне доменне ім’я з доменом верхнього рівня .com вже було кимось зареєстроване то доводиться зупинити свій вибір на доменному імені fake-detection.org.

Для того щоб автоматично, в режимі реального часу змінювати кількість серверів залежно від поточного навантаження можна скористатись таким сервісом від AWS як Auto Scaling. Даний сервіс працює безкоштовно і дозволяє налаштувати умови при яких за певного відсотку завантаженості серверу будуть створюватись нові сервери, а також налаштувати умови при яких кількість серверів буде зменшуватись коли навантаження на систему змінюється.

Даний сервіс є чудовим рішенням оскільки він гарантує, що сервіс буде завжди доступним для ваших користувачів, економить кошти які витрачаються на оренду серверів контролюючи їх кількість, а також даний сервіс фактично замінює

працівників яким необхідно платити за те щоб вони контролювали навантаження на сервери.

Для того щоб Auto Scaling міг працювати необхідно скористатись іншим платним сервісом під назвою CloudWatch. Даний сервіс відповідає за моніторинг більшості сервісів які надає AWS та надає такі метрики як: завантаженість процесора, об'єм вільної оперативної пам'яті, кількість вільного місці на жорсткому диску, об'єми вхідного та вихідного мережевого трафіку та інші. Завдяки всім цим метрикам стає можливим автоматичне управління кількістю робочих серверів. На рисунку 5.9 зображено ціну за моніторинг одного сервера впродовж місяця. Зазвичай це коштує $2.10 за місяць, а при мінімальному наборі метрик це коштує $0.14 за місяць.



Metrics

All custom metrics charges are prorated by the hour and metered only when you send metrics to CloudWatch.

Amazon EC2 Detailed Monitoring pricing is based on the number of custom metrics, with no API charge for sending metrics. The number of metrics sent by an EC2 instance as part of EC2 Detailed Monitoring is dependent on the instance type --- for details, see the Instance Metrics documentation. Typically, EC2 Detailed Monitoring is charged at $2.10 per instance per month (assumes 7 metrics per instance) and goes down to $0.14 per instance at the lowest priced tier. As with all custom metrics, EC2 Detailed Monitoring is prorated by the hour and metered only when the instance sends metrics to CloudWatch.

Tiers	Cost (metric/month)
First 10,000 metrics	$0.30
Next 240,000 metrics	$0.10
Next 750,000 metrics	$0.05
Over 1,000,000 metrics	$0.02

Рисунок 5.9 — Вартість сервісу CloudWatch

Для того щоб рівномірно розподілити навантаження між всіма серверами можна скористатись таким сервісом від AWS як Elastic Load Balancer. AWS надає чотири різних варіації даного сервісу:

— Application Load Balancer;

— Network Load Balancer;

— Gateway Load Balancer;

— Classic Load Balancer.

Для даного стартапу найкраще підходить другий варіант оскільки він надає ряд додаткових можливостей порівняно з останнім варіантом який тільки дає змогу розподіляти навантаження між різними серверами (рис. 5.10).

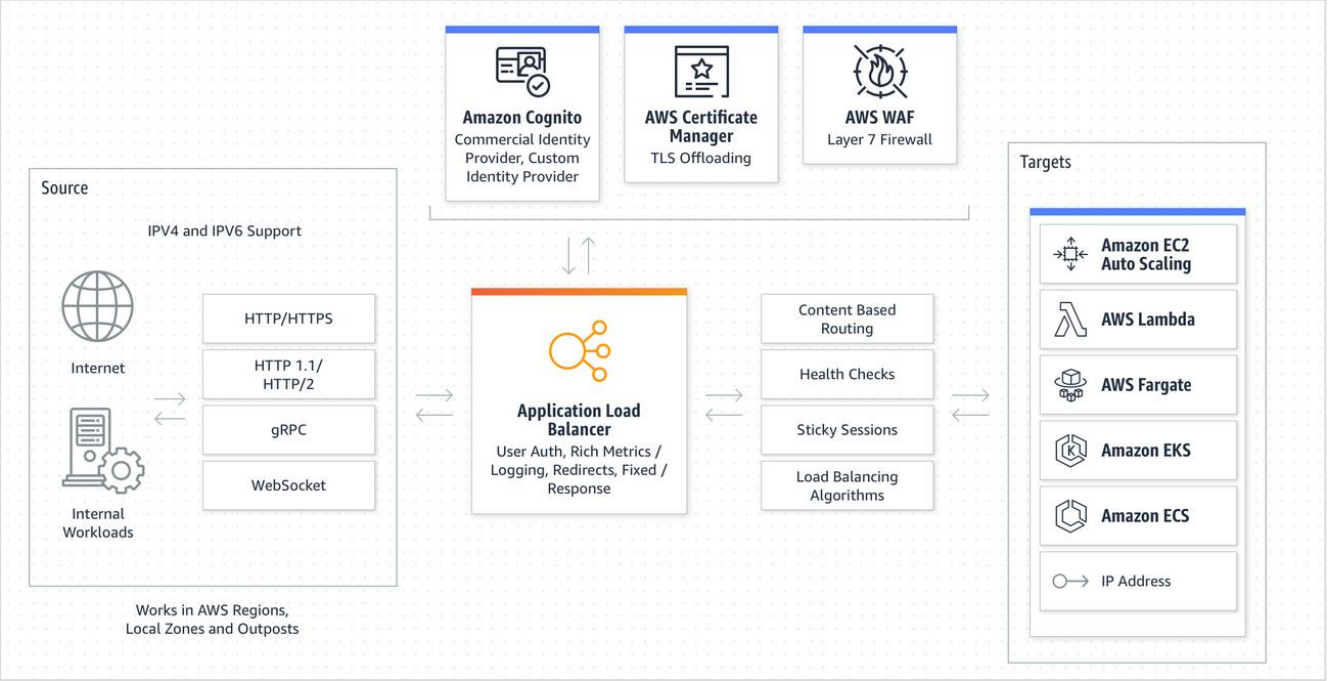


Рисунок 5.10 — Функції Network Load Balancer

Серед цих додаткових можливостей є наступні:

— Sticky Sessions: дана функція дає можливість прикріпити користувача до одного серверу який буде обробляти всі його запити. Таким чином виключається можливість того, що при кожному новому запиті буде створюватись нова сесія та гаятись час.

— Low Latency: дана функція забезпечує низьку відповіді від сервера, що можу бути критичним для деяких типів застосунків.

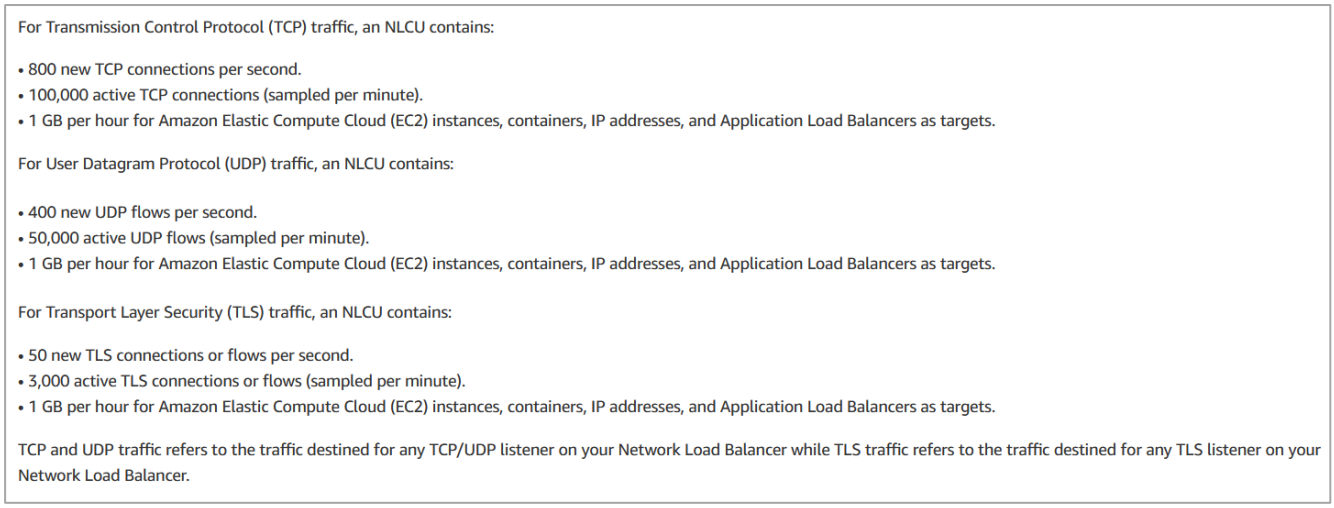
— Static IP support: дана функція забезпечує надання балансиру статичної IP адреси до якої може звертатись користувацький інтерфейс застосунку.

— Elastic IP support: дана функція дозволяє прив’язати до балансира свою власну IP адресу яку надає Elastic IP.

— DNS Fail-over: дана функція дозволяє перенаправляти трафік в інший регіон у випадку якщо сервери або балансир в регіоні де знаходиться користувач вийшли з ладу.

— Preserve source IP address: дана функція забезпечує передачу IP адреси користувача для того щоб застосунок міг використовувати її в подальшому.

На рисунку 5.11 зображено граничні об’єми трафіку який може проходити через один Network Load Balancer Capacity Unit (NLCU) для кожного з трьох протоколів (TCP, UDP, TLS). Сума оплати за кожну годину нараховується на основі того протоколу через який пройшло найбільше трафіку та як наслідок було задіяно найбільше NLCU. За кожен NLCU нараховується сума в розмірі $0.006 за одну годину.



For Transmission Control Protocol (TCP) traffic, an NLCU contains:

- 800 new TCP connections per second.
- 100,000 active TCP connections (sampled per minute).
- 1 GB per hour for Amazon Elastic Compute Cloud (EC2) instances, containers, IP addresses, and Application Load Balancers as targets.

For User Datagram Protocol (UDP) traffic, an NLCU contains:

- 400 new UDP flows per second.
- 50,000 active UDP flows (sampled per minute).
- 1 GB per hour for Amazon Elastic Compute Cloud (EC2) instances, containers, IP addresses, and Application Load Balancers as targets.

For Transport Layer Security (TLS) traffic, an NLCU contains:

- 50 new TLS connections or flows per second.
- 3,000 active TLS connections or flows (sampled per minute).
- 1 GB per hour for Amazon Elastic Compute Cloud (EC2) instances, containers, IP addresses, and Application Load Balancers as targets.

TCP and UDP traffic refers to the traffic destined for any TCP/UDP listener on your Network Load Balancer while TLS traffic refers to the traffic destined for any TLS listener on your Network Load Balancer.

Рисунок 5.11 — Об’єми трафіку для одного NLCU

Також додатково стягується оплата за кожен балансир навантаження в розмірі $0.027 за годину. Таким чином мінімальна вартість за користування балансиром навантаження становить $0.033 за годину та $23.76 за місяць неперервного користування відповідно.

В сумі для підтримки даної інфраструктури при мінімальному навантаженні доведеться витрачати приблизно $120 за місяць

5.7 Запуск маркетингової кампанії

Для проведення рекламної кампанії доцільніше всього буде скористатись послугами такої соціальної мережі як LinkedIn. Дана платформа є перспективною оскільки в ній основну масу користувачів складають люди які дотичні до сфери IT. Таким чином можна бути впевненим що рекламу побачать люди які є відповідальними за прийняття рішень в компаніях які можуть зацікавитися програмним продуктом і в результаті стати клієнтами. Завдяки цьому можна значно зменшити витрати на маркетинг оскільки не потрібно платити за показ реклами мільйонам людей, а можна обмежитись лише декількома тисячами.

На рисунку 5.12 зображено порівняльну характеристику найбільш відомих платформ які використовуються для розміщення реклами за наступними характеристиками:

— Professional Data даний критерій визначає чи володіє платформа достатньою інформацією про користувачів для того щоб реклама могла досягти потрібних людей. До такої інформації можна віднести посаду, галузі, стаж, місцезнаходження та інші

— Business Context даний критерій визначає наскільки користувачі платформи зацікавлені бізнес темами та розвитком своєї кар’єри, а також тим наскільки середовище сприятливе для ведення бізнес листування та пошуку нових партнерів

— News Feed Products даний критерій визначає наскільки добре інтегрована реклама в стрічку новин користувача та чи можна такий досвід користування платформою назвати позитивним.

Partner	Professional Data	Business Context	Newsfeed Products
LinkedIn	✓	✓	✓
Facebook			✓
Twitter			✓
NYTimes		✓	
WSJ		✓	
Dun & Bradstreet	?		

Рисунок 5.12 — Порівняльна характеристика рекламних платформ

Для того щоб почати рекламувати свій продукт для початку необхідно створити Campaign Manager account. Після чого можна приступати до створення рекламних кампаній. Від створення менеджера кампаній до запуску своєї першої кампанії необхідно пройти ряд кроків:

— Для початку необхідно створити нову або вибрати одну з існуючих груп кампаній.

— Далі необхідно вибрати ціль для кампанії. LinkedIn пропонує на вибір три варіанти це: максимальне підвищення впізнаваності бренду чи продукту, заохочення клієнтів до того щоб дізнатися якомога більше про бізнес за допомогою їх направлення на головну сторінку продукту чи залучаючи їх до інших соціальних дій та переглядів, останній варіант направлений на отримання потенційних клієнтів шляхом відстеження їх дій таких як завантаження документації продукту. Для даного стартапу найбільше підходить останній варіант який направлений на отримання клієнтів.

— Далі йде одна з найважливіших частин це вибір цільової аудиторії. LinkedIn дозволяє вибрати регіон в якому буде показуватись реклама, мову користувачів які її мають побачити, вказати критерії вибору компаній, та при

бажані вибрати конкретних користувачів завантаживши в системи список з їх контактами. Для даного стартапу буде доцільно обрати такі регіони як Європа та Сполучені Штати Америки, мову англійську, та компанії сферою діяльності яких є соціальні мережі та месенджери.

— Після того як було обрано цільову аудиторію необхідно вибрати один з типів реклами (рис. 5.13). Для того щоб продемонструвати якість роботи системи для розпізнавання фейкових новин найкращим варіантом буде обрати відео формат оскільки він легше сприймається та тому, що люди не люблять читати багато текстової інформації, а використовуючи статичні зображення буде складно пояснити всі переваги використання системи.

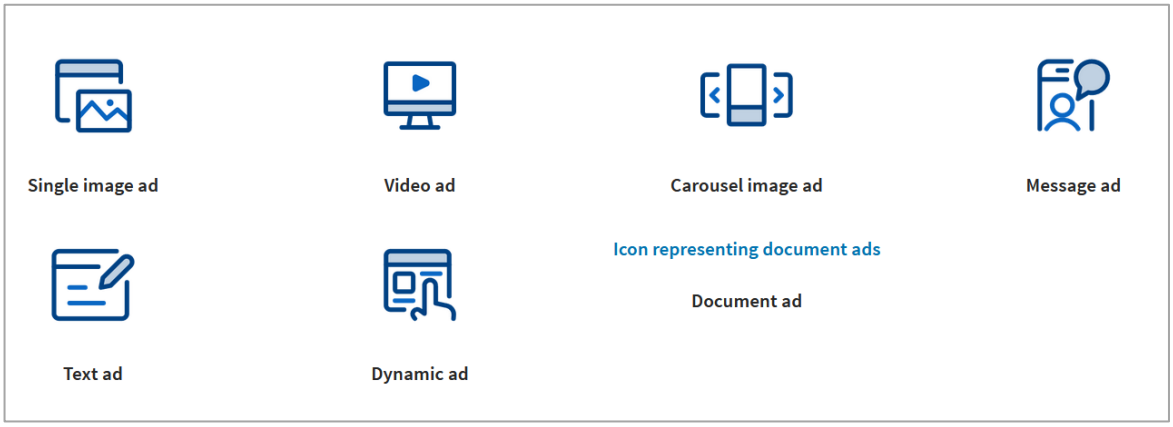


Рисунок 5.13 — Доступні типи реклами на LinkedIn

— Після цього необхідно вказати чи дозволяєте ви щоб LinkedIn розміщував рекламу на сторонніх платформах де присутня цільова аудиторія яка має облікові записи на LinkedIn. В даному випадку буде краще обмежитись публікацією реклами виключно на LinkedIn оскільки ця платформа призначення в основному для ведення ділових справ і тому це буде доцільнішим.

— Останнім обов’язковим кроком перед запуском рекламної кампанії є встановлення бюджету та графіку за яким буде запускатись реклама (рис. 5.14). На даний момент можливо запускати рекламу на постійній основі (доки вона не буде вимкнута) та у фіксованому часовому проміжку. Також можливо встановити

денний бюджет або загальний який буде використовуватись до тих пір поки не закінчяться кошти. Для даного стартапу буде доцільно запустити рекламу з денним бюджетом 50-100 доларів на термін від семи до чотирнадцяти днів та дослідити реакцію ринку на продукт.

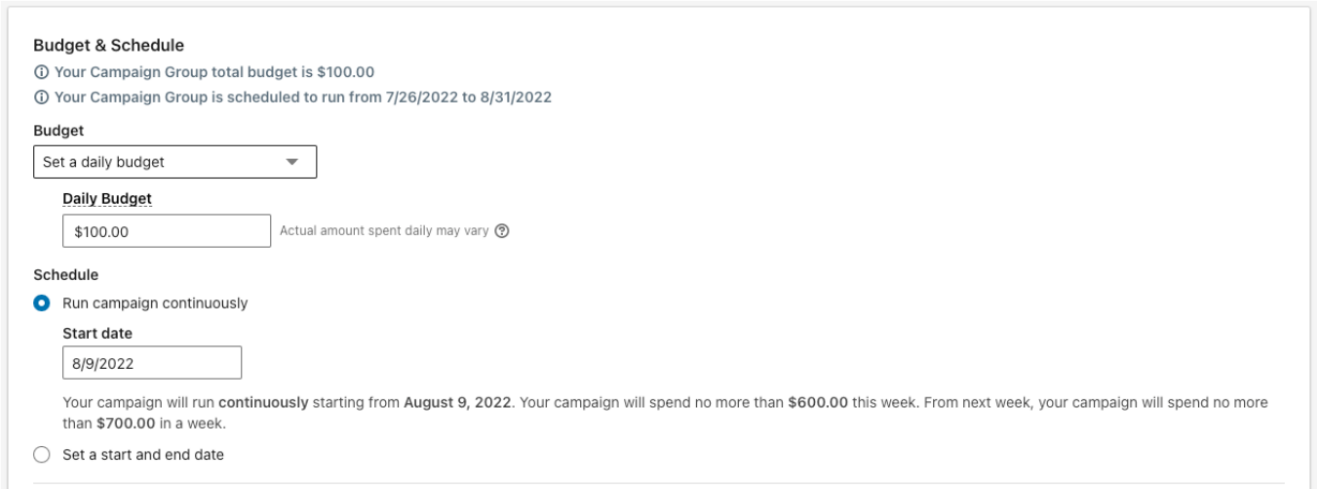


Рисунок 5.14 — Налаштування бюджету та графіку

Після того як всі ці кроки було зроблено можна за бажанням додати до свого веб-сайту спеціальний тег який буде відслідковувати користувачів які перейшли до нього за посиланням яке було в рекламі. Таким чином можна відслідковувати параметр в маркетингу який називається conversion rate що показує те наскільки реклама є ефективною та наскільки добре вона є привабливою для користувачів.

Після цього необов’язкового кроку все що залишається так це перевірити чи все гаразд з налаштуваннями та запустити рекламу.

5.8 Обґрунтування рівня рентабельності стартапу

Для того щоб почати отримувати кошти зі свого стартапу необхідно визначити ціни за надання своїх послуг користувачам, а також створити механізм який буде виключати користування послугами без попередньої оплати за них.

Причому ціни необхідно встановити таким чином, щоб покрити постійні витрати на підтримання інфраструктури та розробку програмного продукту, а також залучити максимально можливу кількість клієнтів.

Для того щоб контролювати використання послуг окремим клієнтом можна створити механізм який використовує JWT токени в своїй роботі. Кожен такий токен може містити всю необхідну інформацію про користувача якому він належить та гарантувати що ніхто інший не буде володіти таким токеном. Перевагою використання токенів є те, що їх неможливо визначити простим перебором оскільки вони як правило складаються з рядка довжиною більше 100 символів, а у випадку якщо токен потрапив у треті руки, то власник облікового запису якому він належить завжди може його відкликати та створити новий.

Для того щоб кожен з нових користувачів міг на власному досвіді переконатись у тому на скільки добре працює даний програмний продукт буде доцільним надати бонус у вигляді безкоштовної перевірки першої тисячі новин. Така стратегія має прискорити ріст кількості користувачів.

У разі якщо стартап досягне успіху і зможе знайти великих клієнтів то буде гарною ідеєю зробити знижки для користувачів які перевіряють за місяць більше ніж певне число новин, наприклад:

— Якщо користувач перевіряє від нуля до одного мільйона новин в місяць то плата за перевірку однієї новини становитиме $0.00001 та $10 за один мільйон відповідно.

— Для користувачів які перевіряють від одного до десяти мільйонів новин в місяць плата за перевірку однієї новини становитиме $0.000009 та $9 за один мільйон відповідно.

— Для великих компаній які будуть перевіряти в місяць більше ніж десять мільйонів новин буде діяти знижка в розмірі 20% і вони будуть платити за перевірку однієї новини $0.000008 та $8 за один мільйон відповідно.

Плата за користування послугами буде стягуватись відразу після кожного перевіреного мільйону новин або в кінці місяця якщо це невеликий клієнт який

перевіряє за місяць менше одного мільйона новин. У разі якщо в клієнта не достатньо коштів для оплати послуг його токен буде відразу заблокований і він не зможе користуватись сервісом доки не оплатить рахунок. Також наявність знижки в поточному місяці буде напряму залежати від того як багато він користувався послугами сервісу в попередньому місяці. Таким чином знижка для певного користувача може з’являтись та зникати і її неможливо буде отримати один раз і назавжди.

Якщо орієнтуватись на те що стартап зможе знайти клієнтів середнього розміру, то для того щоб компенсувати витрати на підтримку архітектури необхідно як мінімум 5 клієнтів які будуть перевіряти в місяць не менше трьох мільйонів новин.

Висновки до розділу 5

В ході даної роботи було проведено аналіз щодо можливості створення стартапу на базі теми даної магістерської дисертації. Було сформовано основні кроки щодо його реалізації серед яких: аналіз ситуації на ринку, аналіз ризиків, витрати на реалізацію та підтримку програмного продукту, запуск маркетингової кампанії та обґрунтування рентабельності стартапу.

ВИСНОВКИ

В ході розробки системи автоматичного визначення фейкових новин в сучасному інформаційному просторі було виконано наступне:

— досліджено наявну літературу на базі практики стосовно такої області штучного інтелекту як Natural language processing (NLP), а також способів імплементації таких моделей в існуючих системах;

— обрано стек технологій який дозволить в найкоротші строки натренувати моделі для класифікації та застосувати їх в програмному продукті, а також обрано технології для розгортання системи та побудови користувацького інтерфейсу;

— зібрано з відкритих джерел датасет з прикладів достовірних та фейкових новин для подальшого навчання моделей для класифікації;

— проведено попередню підготовку даних перед використанням їх для тренування моделей з метою покращення точності роботи моделей;

— проведено тренування моделей з використанням різних алгоритмів з подальшим вибором найкращої з них;

— розроблено зручний та зрозумілий користувацький інтерфейс;

— розгорнуто систему з використанням хмарного сервісу EC2 (Elastic Compute Cloud) який надає компанія Amazon;

— перевірено коректність роботи кожного окремого модуля, а також їх взаємодію між собою в рамках однієї системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Speech and language processing : handbook / D. Jurafsky et al. 2nd ed. 2014. 1024 p.

2. Indurkhya N. Handbook of natural language processing (chapman & hall/crc: machine learning & pattern recognition) : handbook. 2nd ed. Chapman and Hall/CRC, 2010. 702 p.

3. The handbook of computational linguistics and natural language processing : handbook / ed. by A. Clark, C. Fox, S. Lappin. Wiley-Blackwell, 2013. 804 p.

4. Bird S., Klein E., Loper E. Natural language processing with python: analyzing text with the natural language toolkit : handbook. O'Reilly Media, 2009. 903 p.

5. Ghavami P. Big data analytics methods: modern analytics techniques for the 21st century: the data scientist’s manual to data mining, deep learning & natural language processing : handbook. CreateSpace Independent Publishing Platform, 2016. 304 p.

6. Robbins J. Learning web design: A beginner's guide to HTML, CSS, javascript, and web graphics : handbook. 4th ed. O'Reilly Media, 2012. 624 p.

7. Kyrnin J., Meloni J. C. HTML, CSS, and javascript all in one: covering HTML5, CSS3, and ES6 : handbook. 3rd ed. Sams Publishing, 2018. 800 p.

8. Santana R. React 17 Design Patterns and Best Practices: design, build, and deploy production-ready web applications using industry-standard practices : handbook. 3rd ed. 2021. 394 p.

9. Gordon Z. React explained: your step-by-step guide to React : handbook. 2019. 366 p.

10. Duckett J. HTML and CSS: design and build websites : handbook. John Wiley & Sons, 2011. 490 p.

11. Frain B. Responsive Web Design with HTML5 and CSS: Develop future-proof responsive websites using the latest HTML5 and CSS techniques : handbook. 3rd ed. Packt Publishing, 2020. 410 p.

12. Myers M. A Smarter Way to Learn HTML & CSS: Learn it faster. Remember it longer : handbook. 2015. 231 p.

13. Ramalho L. Fluent Python : handbook. O'Reilly Media, 2015. 769 p.

14. Beazley D., Jones B. Python cookbook: recipes for mastering Python 3 : handbook. 3rd ed. O'Reilly Media, 2013. 706 p.

15. Slatkin B. Effective Python: 90 specific ways to write better Python : handbook. 2nd ed. Addison-Wesley Professional, 2020. 480 p.

16. Ravindran A. Django design patterns and best practices: industry-standard web development techniques and solutions using python : handbook. 2nd ed. 2018. 282 p.

17. William S. Vincent. Django for APIs: Build web APIs with Python & Django : handbook. 2018. 190 p.

ДОДАТОК А

Лістинг програми

Система автоматичного визначення фейкових новин в сучасному інформаційному просторі

УКР.НТУУ«КПІ»_НН ІАТЕ_ІПЗЕ_ ТВ-11мп

Аркушів 9

Київ – 2022

LogisticRegression

```
# Importing libraries
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings( "ignore" )

# to compare our model's accuracy with sklearn model
from sklearn.linear_model import LogisticRegression
# Logistic Regression
class LogitRegression() :
    def __init__( self, learning_rate, iterations ) :
        self.learning_rate = learning_rate
        self.iterations = iterations

    # Function for model training
    def fit( self, X, Y ) :
        # no_of_training_examples, no_of_features
        self.m, self.n = X.shape
        # weight initialization
        self.W = np.zeros( self.n )
        self.b = 0
        self.X = X
        self.Y = Y

        # gradient descent learning

        for i in range( self.iterations ) :
            self.update_weights()
        return self

    # Helper function to update weights in gradient descent

    def update_weights( self ) :
        A = 1 / ( 1 + np.exp( - ( self.X.dot( self.W ) + self.b ) ) )

        # calculate gradients
        tmp = ( A - self.Y.T )
        tmp = np.reshape( tmp, self.m )
        dW = np.dot( self.X.T, tmp ) / self.m
        db = np.sum( tmp ) / self.m

        # update weights
        self.W = self.W - self.learning_rate * dW
        self.b = self.b - self.learning_rate * db

        return self

    # Hypothetical function  h( x )

    def predict( self, X ) :
        Z = 1 / ( 1 + np.exp( - ( X.dot( self.W ) + self.b ) ) )
        Y = np.where( Z > 0.5, 1, 0 )
        return Y


# Driver code

def main() :

    # Importing dataset
    df = pd.read_csv( "diabetes.csv" )
    X = df.iloc[:,:-1].values
```

```
    Y = df.iloc[:,-1:].values

    # Splitting dataset into train and test set
    X_train, X_test, Y_train, Y_test = train_test_split(
      X, Y, test_size = 1/3, random_state = 0 )

    # Model training
    model = LogitRegression( learning_rate = 0.01, iterations = 1000 )

    model.fit( X_train, Y_train )
    model1 = LogisticRegression()
    model1.fit( X_train, Y_train)

    # Prediction on test set
    Y_pred = model.predict( X_test )
    Y_pred1 = model1.predict( X_test )

    # measure performance
    correctly_classified = 0
    correctly_classified1 = 0

    # counter
    count = 0
    for count in range( np.size( Y_pred ) ) :

        if Y_test[count] == Y_pred[count] :
            correctly_classified = correctly_classified + 1

        if Y_test[count] == Y_pred1[count] :
            correctly_classified1 = correctly_classified1 + 1

        count = count + 1

    print( "Accuracy on test set by our model       :  ", (
      correctly_classified / count ) * 100 )
    print( "Accuracy on test set by sklearn model   :  ", (
      correctly_classified1 / count ) * 100 )


if __name__ == "__main__" :
    main()
```

SupportVectorMachine

```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split as tts
from sklearn.metrics import accuracy_score, recall_score, precision_score
from sklearn.utils import shuffle


# >> FEATURE SELECTION << #
def remove_correlated_features(X):
    corr_threshold = 0.9
    corr = X.corr()
    drop_columns = np.full(corr.shape[0], False, dtype=bool)
    for i in range(corr.shape[0]):
        for j in range(i + 1, corr.shape[0]):
            if corr.iloc[i, j] >= corr_threshold:
                drop_columns[j] = True
    columns_dropped = X.columns[drop_columns]
    X.drop(columns_dropped, axis=1, inplace=True)
    return columns_dropped
```

```
def remove_less_significant_features(X, Y):
    sl = 0.05
    regression_ols = None
    columns_dropped = np.array([])
    for itr in range(0, len(X.columns)):
        regression_ols = sm.OLS(Y, X).fit()
        max_col = regression_ols.pvalues.idxmax()
        max_val = regression_ols.pvalues.max()
        if max_val > sl:
            X.drop(max_col, axis='columns', inplace=True)
            columns_dropped = np.append(columns_dropped, [max_col])
        else:
            break
    regression_ols.summary()
    return columns_dropped


##############################


# >> MODEL TRAINING << #
def compute_cost(W, X, Y):
    # calculate hinge loss
    N = X.shape[0]
    distances = 1 - Y * (np.dot(X, W))
    distances[distances < 0] = 0  # equivalent to max(0, distance)
    hinge_loss = regularization_strength * (np.sum(distances) / N)

    # calculate cost
    cost = 1 / 2 * np.dot(W, W) + hinge_loss
    return cost


# I haven't tested it but this same function should work for
# vanilla and mini-batch gradient descent as well
def calculate_cost_gradient(W, X_batch, Y_batch):
    # if only one example is passed (eg. in case of SGD)
    if type(Y_batch) == np.float64:
        Y_batch = np.array([Y_batch])
        X_batch = np.array([X_batch])  # gives multidimensional array

    distance = 1 - (Y_batch * np.dot(X_batch, W))
    dw = np.zeros(len(W))

    for ind, d in enumerate(distance):
        if max(0, d) == 0:
            di = W
        else:
            di = W - (regularization_strength * Y_batch[ind] * X_batch[ind])
        dw += di

    dw = dw/len(Y_batch)  # average
    return dw


def sgd(features, outputs):
    max_epochs = 5000
    weights = np.zeros(features.shape[1])
    nth = 0
    prev_cost = float("inf")
    cost_threshold = 0.01  # in percent
    # stochastic gradient descent
    for epoch in range(1, max_epochs):
        # shuffle to prevent repeating update cycles
        X, Y = shuffle(features, outputs)
        for ind, x in enumerate(X):
            ascent = calculate_cost_gradient(weights, x, Y[ind])
            weights = weights - (learning_rate * ascent)

        # convergence check on 2^nth epoch
        if epoch == 2 ** nth or epoch == max_epochs - 1:
            cost = compute_cost(weights, features, outputs)
            print("Epoch is: {} and Cost is: {}".format(epoch, cost))
            # stoppage criterion
            if abs(prev_cost - cost) < cost_threshold * prev_cost:
                return weights
```

```
            prev_cost = cost
            nth += 1
    return weights


########################


def init():
    print("reading dataset...")
    # read data in pandas (pd) data frame
    data = pd.read_csv('./data/data.csv')

    # drop last column (extra column added by pd)
    # and unnecessary first column (id)
    data.drop(data.columns[[-1, 0]], axis=1, inplace=True)

    print("applying feature engineering...")
    # convert categorical labels to numbers
    diag_map = {'M': 1.0, 'B': -1.0}
    data['diagnosis'] = data['diagnosis'].map(diag_map)

    # put features & outputs in different data frames
    Y = data.loc[:, 'diagnosis']
    X = data.iloc[:, 1:]

    # filter features
    remove_correlated_features(X)
    remove_less_significant_features(X, Y)

    # normalize data for better convergence and to prevent overflow
    X_normalized = MinMaxScaler().fit_transform(X.values)
    X = pd.DataFrame(X_normalized)

    # insert 1 in every row for intercept b
    X.insert(loc=len(X.columns), column='intercept', value=1)

    # split data into train and test set
    print("splitting dataset into train and test sets...")
    X_train, X_test, y_train, y_test = tts(X, Y, test_size=0.2, random_state=42)

    # train the model
    print("training started...")
    W = sgd(X_train.to_numpy(), y_train.to_numpy())
    print("training finished.")
    print("weights are: {}".format(W))

    # testing the model
    print("testing the model...")
    y_train_predicted = np.array([])
    for i in range(X_train.shape[0]):
        yp = np.sign(np.dot(X_train.to_numpy()[i], W))
        y_train_predicted = np.append(y_train_predicted, yp)

    y_test_predicted = np.array([])
    for i in range(X_test.shape[0]):
        yp = np.sign(np.dot(X_test.to_numpy()[i], W))
        y_test_predicted = np.append(y_test_predicted, yp)

    print("accuracy on test dataset: {}".format(accuracy_score(y_test, y_test_predicted)))
    print("recall on test dataset: {}".format(recall_score(y_test, y_test_predicted)))
    print("precision on test dataset: {}".format(recall_score(y_test, y_test_predicted)))


# set hyper-parameters and call init
regularization_strength = 10000
learning_rate = 0.000001
init()
```

DecisionTree

```
# Data wrangling
import pandas as pd
```

```
# Array math
import numpy as np

# Quick value count calculator
from collections import Counter


class Node:
    """
    Class for creating the nodes for a decision tree
    """
    def __init__(
        self,
        Y: list,
        X: pd.DataFrame,
        min_samples_split=None,
        max_depth=None,
        depth=None,
        node_type=None,
        rule=None
    ):
        # Saving the data to the node
        self.Y = Y
        self.X = X

        # Saving the hyper parameters
        self.min_samples_split = min_samples_split if min_samples_split else 20
        self.max_depth = max_depth if max_depth else 5

        # Default current depth of node
        self.depth = depth if depth else 0

        # Extracting all the features
        self.features = list(self.X.columns)

        # Type of node
        self.node_type = node_type if node_type else 'root'

        # Rule for spliting
        self.rule = rule if rule else ""

        # Calculating the counts of Y in the node
        self.counts = Counter(Y)

        # Getting the GINI impurity based on the Y distribution
        self.gini_impurity = self.get_GINI()

        # Sorting the counts and saving the final prediction of the node
        counts_sorted = list(sorted(self.counts.items(), key=lambda item: item[1]))

        # Getting the last item
        yhat = None
        if len(counts_sorted) > 0:
            yhat = counts_sorted[-1][0]

        # Saving to object attribute. This node will predict the class with the most frequent class
        self.yhat = yhat

        # Saving the number of observations in the node
        self.n = len(Y)

        # Initiating the left and right nodes as empty nodes
        self.left = None
        self.right = None

        # Default values for splits
        self.best_feature = None
        self.best_value = None

    @staticmethod
    def GINI_impurity(y1_count: int, y2_count: int) -> float:
        """
        Given the observations of a binary class calculate the GINI impurity
        """
        # Ensuring the correct types
```

```
        if y1_count is None:
            y1_count = 0

        if y2_count is None:
            y2_count = 0

        # Getting the total observations
        n = y1_count + y2_count

        # If n is 0 then we return the lowest possible gini impurity
        if n == 0:
            return 0.0

        # Getting the probability to see each of the classes
        p1 = y1_count / n
        p2 = y2_count / n

        # Calculating GINI
        gini = 1 - (p1 ** 2 + p2 ** 2)

        # Returning the gini impurity
        return gini

    @staticmethod
    def ma(x: np.array, window: int) -> np.array:
        """
        Calculates the moving average of the given list.
        """
        return np.convolve(x, np.ones(window), 'valid') / window

    def get_GINI(self):
        """
        Function to calculate the GINI impurity of a node
        """
        # Getting the 0 and 1 counts
        y1_count, y2_count = self.counts.get(0, 0), self.counts.get(1, 0)

        # Getting the GINI impurity
        return self.GINI_impurity(y1_count, y2_count)

    def best_split(self) -> tuple:
        """
        Given the X features and Y targets calculates the best split
        for a decision tree
        """
        # Creating a dataset for spliting
        df = self.X.copy()
        df['Y'] = self.Y

        # Getting the GINI impurity for the base input
        GINI_base = self.get_GINI()

        # Finding which split yields the best GINI gain
        max_gain = 0

        # Default best feature and split
        best_feature = None
        best_value = None

        for feature in self.features:
            # Droping missing values
            Xdf = df.dropna().sort_values(feature)

            # Sorting the values and getting the rolling average
            xmeans = self.ma(Xdf[feature].unique(), 2)

            for value in xmeans:
                # Spliting the dataset
                left_counts = Counter(Xdf[Xdf[feature]<value]['Y'])
                right_counts = Counter(Xdf[Xdf[feature]>=value]['Y'])

                # Getting the Y distribution from the dicts
                y0_left, y1_left, y0_right, y1_right = left_counts.get(0, 0), left_counts.get(1, 0), right_counts.get(0, 0), right_counts.get(1, 0)

                # Getting the left and right gini impurities
                gini_left = self.GINI_impurity(y0_left, y1_left)
```

```
            gini_right = self.GINI_impurity(y0_right, y1_right)

            # Getting the obs count from the left and the right data splits
            n_left = y0_left + y1_left
            n_right = y0_right + y1_right

            # Calculating the weights for each of the nodes
            w_left = n_left / (n_left + n_right)
            w_right = n_right / (n_left + n_right)

            # Calculating the weighted GINI impurity
            wGINI = w_left * gini_left + w_right * gini_right

            # Calculating the GINI gain
            GINIgain = GINI_base - wGINI

            # Checking if this is the best split so far
            if GINIgain > max_gain:
                best_feature = feature
                best_value = value

                # Setting the best gain to the current one
                max_gain = GINIgain

    return (best_feature, best_value)

def grow_tree(self):
    """
    Recursive method to create the decision tree
    """
    # Making a df from the data
    df = self.X.copy()
    df['Y'] = self.Y

    # If there is GINI to be gained, we split further
    if (self.depth < self.max_depth) and (self.n >= self.min_samples_split):

        # Getting the best split
        best_feature, best_value = self.best_split()

        if best_feature is not None:
            # Saving the best split to the current node
            self.best_feature = best_feature
            self.best_value = best_value

            # Getting the left and right nodes
            left_df, right_df = df[df[best_feature]<=best_value].copy(), df[df[best_feature]>best_value].copy()

            # Creating the left and right nodes
            left = Node(
                left_df['Y'].values.tolist(),
                left_df[self.features],
                depth=self.depth + 1,
                max_depth=self.max_depth,
                min_samples_split=self.min_samples_split,
                node_type='left_node',
                rule=f"{best_feature} <= {round(best_value, 3)}"
                )

            self.left = left
            self.left.grow_tree()

            right = Node(
                right_df['Y'].values.tolist(),
                right_df[self.features],
                depth=self.depth + 1,
                max_depth=self.max_depth,
                min_samples_split=self.min_samples_split,
                node_type='right_node',
                rule=f"{best_feature} > {round(best_value, 3)}"
                )

            self.right = right
            self.right.grow_tree()

def print_info(self, width=4):
```

```
        """
        Method to print the infromation about the tree
        """
        # Defining the number of spaces
        const = int(self.depth * width ** 1.5)
        spaces = "-" * const

        if self.node_type == 'root':
            print("Root")
        else:
            print(f"|{spaces} Split rule: {self.rule}")
        print(f"{' ' * const}   | GINI impurity of the node: {round(self.gini_impurity, 2)}")
        print(f"{' ' * const}   | Class distribution in the node: {dict(self.counts)}")
        print(f"{' ' * const}   | Predicted class: {self.yhat}")

    def print_tree(self):
        """
        Prints the whole tree from the current node to the bottom
        """
        self.print_info()

        if self.left is not None:
            self.left.print_tree()

        if self.right is not None:
            self.right.print_tree()

    def predict(self, X:pd.DataFrame):
        """
        Batch prediction method
        """
        predictions = []

        for _, x in X.iterrows():
            values = {}
            for feature in self.features:
                values.update({feature: x[feature]})

            predictions.append(self.predict_obs(values))

        return predictions

    def predict_obs(self, values: dict) -> int:
        """
        Method to predict the class given a set of features
        """
        cur_node = self
        while cur_node.depth < cur_node.max_depth:
            # Traversing the nodes all the way to the bottom
            best_feature = cur_node.best_feature
            best_value = cur_node.best_value

            if cur_node.n < cur_node.min_samples_split:
                break

            if (values.get(best_feature) < best_value):
                if self.left is not None:
                    cur_node = cur_node.left
            else:
                if self.right is not None:
                    cur_node = cur_node.right

        return cur_node.yhat
```

RandomForest

```
class RandomForest:
    '''
    A class that implements Random Forest algorithm from scratch.
    '''
    def __init__(self, num_trees=25, min_samples_split=2, max_depth=5):
        self.num_trees = num_trees
        self.min_samples_split = min_samples_split
```

```
        self.max_depth = max_depth
        # Will store individually trained decision trees
        self.decision_trees = []

    @staticmethod
    def _sample(X, y):
        '''
        Helper function used for boostrap sampling.

        :param X: np.array, features
        :param y: np.array, target
        :return: tuple (sample of features, sample of target)
        '''
        n_rows, n_cols = X.shape
        # Sample with replacement
        samples = np.random.choice(a=n_rows, size=n_rows, replace=True)
        return X[samples], y[samples]

    def fit(self, X, y):
        '''
        Trains a Random Forest classifier.

        :param X: np.array, features
        :param y: np.array, target
        :return: None
        '''
        # Reset
        if len(self.decision_trees) > 0:
            self.decision_trees = []

        # Build each tree of the forest
        num_built = 0
        while num_built < self.num_trees:
            try:
                clf = DecisionTree(
                    min_samples_split=self.min_samples_split,
                    max_depth=self.max_depth
                )
                # Obtain data sample
                _X, _y = self._sample(X, y)
                # Train
                clf.fit(_X, _y)
                # Save the classifier
                self.decision_trees.append(clf)
                num_built += 1
            except Exception as e:
                continue

    def predict(self, X):
        '''
        Predicts class labels for new data instances.

        :param X: np.array, new instances to predict
        :return:
        '''
        # Make predictions with every tree in the forest
        y = []
        for tree in self.decision_trees:
            y.append(tree.predict(X))

        # Reshape so we can find the most common value
        y = np.swapaxes(a=y, axis1=0, axis2=1)

        # Use majority voting for the final prediction
        predictions = []
        for preds in y:
            counter = Counter(x)
            predictions.append(counter.most_common(1)[0][0])
        return predictions
```

ДОДАТОК Б

Презентація програми

Система автоматичного визначення фейкових новин в сучасному інформаційному просторі

УКР.НТУУ«КПІ»_НН ІАТЕ_ІПЗЕ_ ТВ-11мп

Аркушів 13

Київ – 2022

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут атомної та теплової енергетики

Кафедра інженерії програмного забезпечення в енергетиці

Тема: Система автоматичного визначення фейкових новин
в сучасному інформаційному просторі

Роботу виконав студент 2 курсу групи ТВ-11мп
Кузьменчук Дмитро Олександрович

Науковий керівник:
ктн., доц. Ходаковський Олексій Володимирович

Київ - 2022

АКТУАЛЬНІСТЬ

- Широке використання фейків в сучасному житті
- Негативний вплив фейкових новин
- Використання фейків Російською Федерацією для виправдання своїх злочинів
- Близько 60% інформації люди отримують з інтернету.
- Кожного дня людина знаходить як мінімум пару фейкових новин.
- Близько 40% людей хоча б раз поширювали фейкові новини

МЕТА ДОСЛІДЖЕННЯ

Мета – дослідити ефективність існуючих алгоритмів штучного інтелекту для визначення фейкових новин, розробити свій алгоритм та реалізувати його у вигляді програмного продукту.

Очікуваний результат – розробка та створення системи яка дозволить автоматизувати процес виявлення фейків. Зниження розповсюдження фейкових новин в сучасних інформаційних системах.

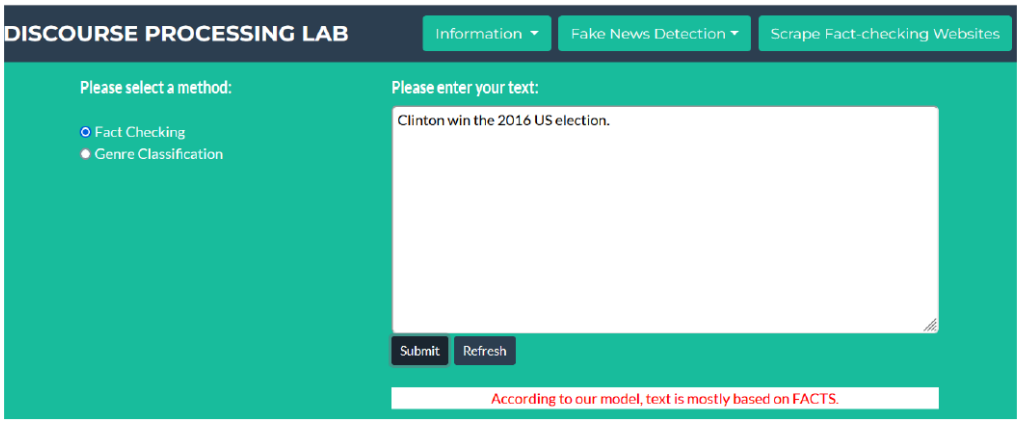
Об'єкт дослідження – аналіз природних мов засобами штучного інтелекту.

Предмет дослідження – виявлення фейкових новин засобами штучного інтелекту.

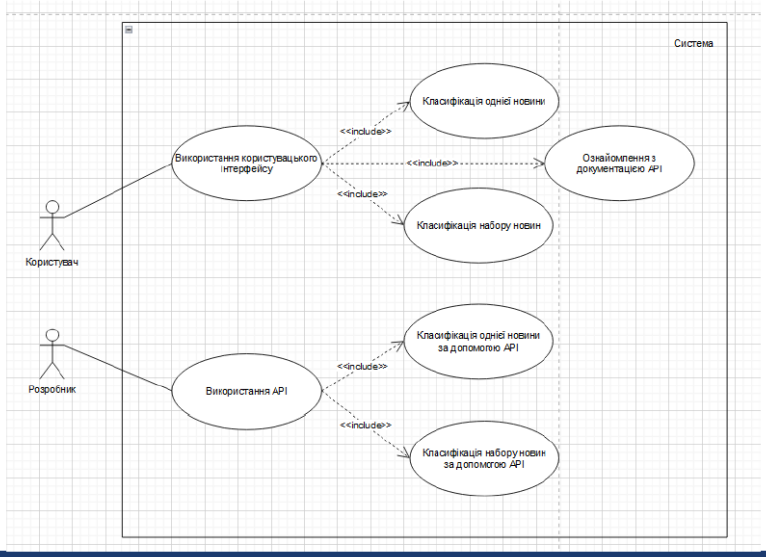
НАУКОВІ ЗАВДАННЯ

- Провести аналіз існуючих систем.
- Провести аналіз існуючих методів для аналізу природніх мов.
- Обрати інструменти для тренування моделі що буде класифікувати фейки.
- Зібрати датасет для тренування моделі.
- Визначити метод який найкраще підходить для вирішення поставленої задачі.

АНАЛІЗ ІСНУЮЧИХ СИСТЕМ



ДІАГРАМА ПРЕЦЕДЕНТІВ



ЗАСОБИ РОЗРОБКИ

СТВОРЕННЯ МОДЕЛІ ДЛЯ КЛАСИФІКАЦІЇ

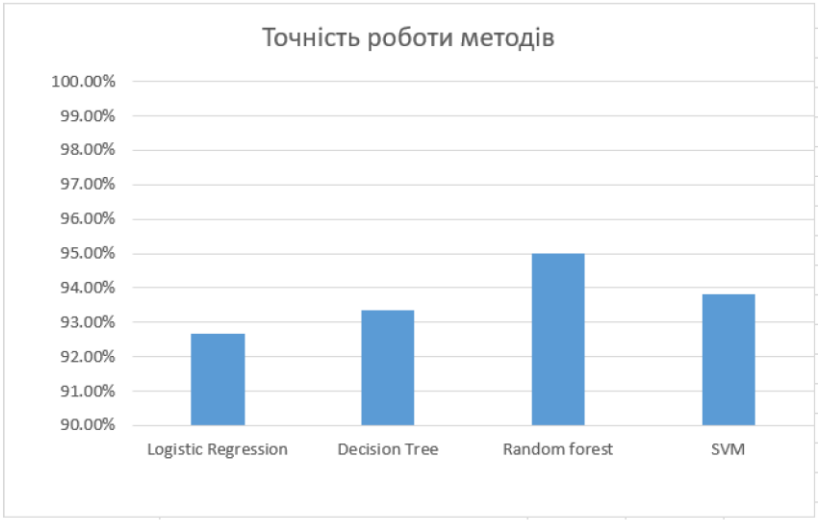
Підготовка даних для навчання

1. Видалення знаків пунктуації та спец символів.
2. Видалення URL з тексту.
3. Приведення до нижнього регістру.
4. Видалення стоп слів.
5. Токенізація.
6. Приведення спільнокореневих слів до єдиної форми

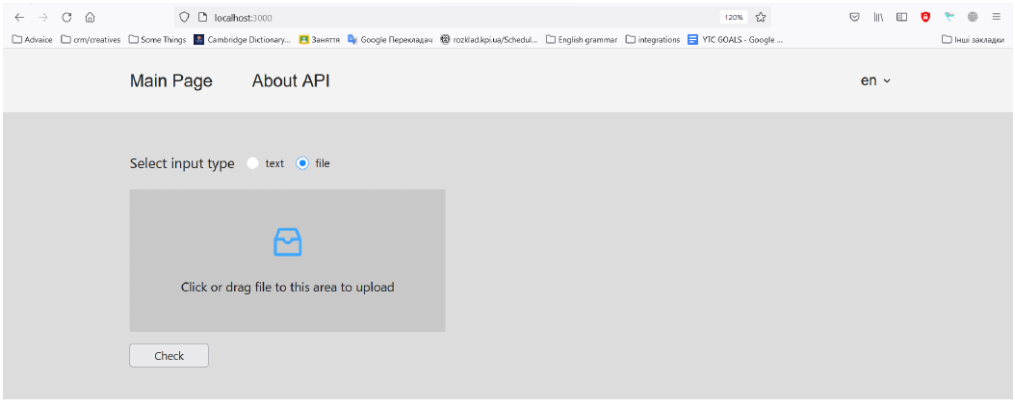
Навчання моделі

1. Створюємо з тренувальної вибірки розміром N підвибірку з повтореннями розміром n.
2. Обираємо з простору ознак M підвибірку розміром m.
3. Будуємо дерево прийняття рішень використовуючи критерій Джинні.
4. Повторюємо попередні кроки для створення інших дерев.

ОТРИМАНІ РЕЗУЛЬТАТИ



ГОЛОВНА СТОРІНКА СИСТЕМИ



РЕЗУЛЬТАТИ РОБОТИ СИСТЕМИ

	A	B	C
1	text	prediction	
2	Donald Trump sent his own plane to transport 200 stranded marines	most likely a fake	
3	FBI director received millions from Clinton's Foundation, his brother law firm does Clinton's taxes	most likely a fake	
4	Pope Francis shocks world, endorses Donald Trump for president	most likely a fake	
5	Ghost of Kyiv' killed in fighting, has shot down 40 Russian jets	probably fake	
6	3 Reasons Why You Should Stop Eating Peanut Butter Cups	fake	
7	Coronavirus Bioweapon: How China Stole Coronavirus From Canada And Weaponized It	most likely a fake	
8	Evidence Surfaces That The FBI Planned And Executed January 6 Capitol Riot	most likely a fake	
9	FEMA Arrives in Tornado - Stricken Kentucky - With VACCINATIONS	fake	
10	French President Tweets Rain Forest Fire Photo	most likely a fake	
11	Israeli Defense Minister: If Pakistan sends ground troops into Syria on any pretext, we will destroy this country with a nuclear attack	uncertain	
12	Leonardo DiCaprio donates $10 million to his grandmother's homeland Ukraine	most likely a fake	
13	Military Arrests SCJ Sonia Sotomayor	most likely a fake	
14	No Reprieve For Consumers As Congress Gives Themselves A Raise & Inflation Expected To Continue Soaring	probably fake	
15	Obama Signs Executive Order Banning The Pledge Of Allegiance In Schools Nationwide	most likely a fake	
16	Bill and Hillary Clinton were reported to be using a Pizza restaurant as a front for a pedophile sex ring.	probably fake	
17	Rupaul claims Trump touched him inappropriately in the 1990s	fake	
18	US Bacon Reserves Hit 50 Year Low	most likely a fake	

ВИСНОВКИ

1. Проаналізовано сферу застосування фейкових новин в сучасних інформаційних системах.
2. Досліджено використання різних методів штучного інтелекту для визначення фейкових новин.
3. Проведено аналіз отриманих результатів для визначення найкращого методу.
4. Розроблено систему яка дозволяє автоматизувати процес визначення фейкових новин.

Дякую за увагу