МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

БЕВЗ ДМИТРО МИКОЛАЙОВИЧ

Допускається до захисту:
В.о. завідувача кафедри
інформаційних технологій,
канд. техн. наук, доцент
_____ Оксана ЗЕЛІНСЬКА
«_____» _____ 2024р.

ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ АЛГОРИТМІВ МАШИННОГО НАВЧАННЯ НА ПРИКЛАДІ ЗАДАЧІ КЛАСИФІКАЦІЇ ТЕКСТІВ

Спеціальність 122 «Комп’ютерні науки»
Кваліфікаційна (магістерська) робота

Керівник:
П.В. Римар, старший викладач
кафедри інформаційних технологій
Науковий консультант:
Ю.С. Антонов, доцент кафедри
інформаційних технологій
к.фіз.-мат.наук, доцент

Оцінка: _____ / _____ / _____
(бали за шкалою ЄКТС / за національною шкалою)
Голова ЕК: _____
(підпис)

Вінниця – 2024

АНОТАЦІЯ

Бевз Д.М. Дослідження ефективності алгоритмів машинного навчання на прикладі задачі класифікації текстів. Спеціальність 122 «Комп'ютерні науки», освітня програма «Комп'ютерні технології обробки даних». Донецький національний університет імені Василя Стуса, Вінниця, 2024.

У кваліфікаційній роботі досліджено ефективність алгоритмів машинного навчання на прикладі задачі класифікації текстів. Для цього було розглянуто три алгоритми машинного навчання, які вирішують проблему класифікації.

На основі результатів навчання певного датасету було зроблено висновки щодо ефективності приведених алгоритмів та приведено графік на їх основі.

Ключові слова: машинне навчання, алгоритм, класифікація текстів, ефективність, наївний класифікатор байєса, логістична регресія, дерево ухвалення рішень.

Робота містить 61 сторінку, 23 рисунки, 4 таблиці та список літератури з 39 джерелами.

ABSTRACT

Bevz D.M. The effectiveness of machine learning algorithms on the example of text classification. Specialty 122 «Computer Science», Educational program «Data Science», Vasyl' Stus Donetsk National University, Vinnytsia, 2024.

In the qualification paper, the effectiveness of machine learning algorithms was investigated using the example of the problem of text classification. For this, three machine learning algorithms that solve the classification problem were considered.

Based on the training results of a certain dataset, conclusions were made regarding the effectiveness of the given algorithms and a graph based on them was given.

Keywords: machine learning, algorithm, text classification, performance, naive bayes classifier, logistic regression, decision tree.

The work contains 61 pages, 23 figures, 4 tables and a bibliography with 39 sources.

ЗМІСТ

ВСТУП

У сучасному світі люди стикаються з експоненціальним зростанням обсягів інформації, що потрапляє до їхніх рук. Аналізувати, обробляти та класифікувати ці дані за категоріями стає вкрай складно, проте, завдяки зростанню обчислювальної потужності комп'ютерів і доступу до сучасних методів, ми маємо можливість вирішувати цю проблему.

Задача класифікації текстів вимагає часу, оскільки потрібно не лише прочитати текст, але й аналізувати його зміст та визначити підходящу категорію зі списку можливих варіантів. При глобальних задачах і великій кількості даних виникають проблеми з недостатністю людських ресурсів для їх обробки.

Укладаючи фундаментальні принципи сучасного світу, науково-технічний прогрес незмінно відіграє ключову роль у вирішенні складних завдань і проблем суспільства. Одним з найбільш актуальних і швидко розвиваючихся напрямків в сфері науки і технологій є машинне навчання, яке є головним інструментом інтелектуального аналізу даних.

Поширення масового обсягу і різноманітності текстової інформації, що є неодмінною частиною сучасного цифрового світу, викликає потребу в розробці та вдосконаленні ефективних алгоритмів машинного навчання для класифікації текстів. Завдання класифікації текстів має велике значення в багатьох галузях, таких як веб-пошук, соціальні медіа, електронна комерція та багато інших.

Зростання обсягу текстових даних, пов’язане зі збільшенням використання онлайн-сервісів і простотою доступу до цих даних, призвело до зростання кількості дослідницьких робіт з класифікації тексту [32].

Класифікація тексту є основним завданням у сфері обробки природної мови, а також базовою технологією для пошуку інформації, системи запитань і відповідей, аналізу емоцій та інших складних завдань. Це одне з найперших застосувань алгоритму машинного навчання, яке досягло хороших результатів [21].

Класифікація тексту на основі машинного навчання є однією з провідних областей досліджень і має широкий спектр застосувань, зокрема виявлення спаму, виявлення ворожих висловлювань, рецензування, узагальнення рейтингів, аналіз настроїв і моделювання тем. Широко використовувані дослідження на основі машинного навчання відрізняються наборами даних, методами навчання, оцінкою ефективності та використовуваними методами порівняння [3].

Актуальність роботи полягає в знаходженні ефективності алгоритмів машинного навчання на прикладі задачі класифікації текстів.

Об'єктом дослідження є особливості розробки алгоритмів машинного навчання на прикладі задачі класифікації текстів.

Предметом дослідження є процес і результат алгоритмів машинного навчання на прикладі задачі класифікації текстів.

Мета роботи – знаходження ефективності алгоритмів машинного навчання на прикладі задачі класифікації текстів.

Завдання магістерської роботи:

- дослідити літературу яка допоможе знайти відповідні алгоритми та особливості їх розробки;
- обрати інструменти та технічну платформу для розробки алгоритмів;
- розробити кожен обраний алгоритм;
- запустити алгоритм на навчання з певними даними;
- оцінити ефективність алгоритмів машинного навчання на прикладі задачі класифікації текстів.

Методи роботи. Мова програмування python, Jupyter Book Colab Notebooks, Kaggle.

Практичне значення отриманих результатів. На основі створених алгоритмів машинного навчання на прикладі задачі класифікації текстів можна оцінити ефективність їх роботи.

Структура роботи: кваліфікаційна робота складається зі вступу, трьох

розділів, висновків, списку використаних джерел. Робота містить 63 сторінки, 23 рисунки, 4 таблиці та список літератури з 39 джерелами.

РОЗДІЛ 1
ПОСТАНОВКА ЗАДАЧІ, АНАЛІЗ ЛІТЕРАТУРИ ТА ОГЛЯД ІСНУЮЧИХ СИСТЕМ ДАНОГО НАПРЯМУ

У даному розділі наведена постановка задачі, проведено аналіз літератури за даною тематикою, а також проведено огляд існуючих програм.

1.1 Постановка задачі

- Провести огляд літератури та аналіз існуючих методів класифікації текстів та алгоритмів машинного навчання.
- Обрати інструменти та технічну платформу для розробки алгоритмів.
- Зібрати та підготувати набір даних для експериментів з класифікацією текстів.
- Реалізувати різні алгоритм для завдання класифікації текстів.
- Оцінити ефективність алгоритмів машинного навчання на прикладі задачі класифікації текстів.

1.2 Задача класифікації тексту

Машинне навчання – тема, яка є в тренді протягом багатьох років. Від сортування електронної пошти до розпізнавання зображень і навіть систем рекомендацій – усе це виконується за допомогою машинного навчання.

Однак машинне навчання в системі штучного інтелекту пов’язане з великою складністю та проблемами, такими як класифікація тексту. Конкретними застосуваннями класифікації тексту можуть бути класифікація юридичних документів, класифікація даних про продукт, класифікація коментарів у банківських переказах, пошук намірів у чат-боті, сортування електронної пошти тощо. Ці програми можуть виглядати зовсім по-різному. Однак через окуляри машинного навчання вони досить схожі [13].

Текст може бути надзвичайно багатим джерелом інформації, але вилучення з нього інформації може бути складним і трудомістким через його неструктуровану природу.

Завдяки прогресу в обробці природної мови та машинному навчанні, які обидва підпадають під широку тему штучного інтелекту, сортувати текстові дані стає легше. Він працює шляхом автоматичного аналізу та структурування тексту, швидко й економічно ефективно, тож підприємства можуть автоматизувати процеси та отримувати інформацію, що веде до кращого прийняття рішень.

Класифікація тексту – це техніка машинного навчання, яка призначає набір попередньо визначених категорій відкритому тексту. Текстові класифікатори можна використовувати для організації, структурування та класифікації майже будь-якого типу тексту: від документів до медичних досліджень і файлів, а також у всьому Інтернеті.

Наприклад, нові статті можна впорядкувати за темами; квитки підтримки можуть бути організовані в терміновому порядку; розмови в чаті можна організувати за мовами; згадування брендів можна організовувати за настроями; і так далі.

Класифікація тексту є одним із основних завдань у обробці природної мови з широким застосуванням, таким як аналіз настроїв, маркування тем, виявлення спаму та виявлення намірів.

Приклад цього можна побачити на рисунку 1.1.

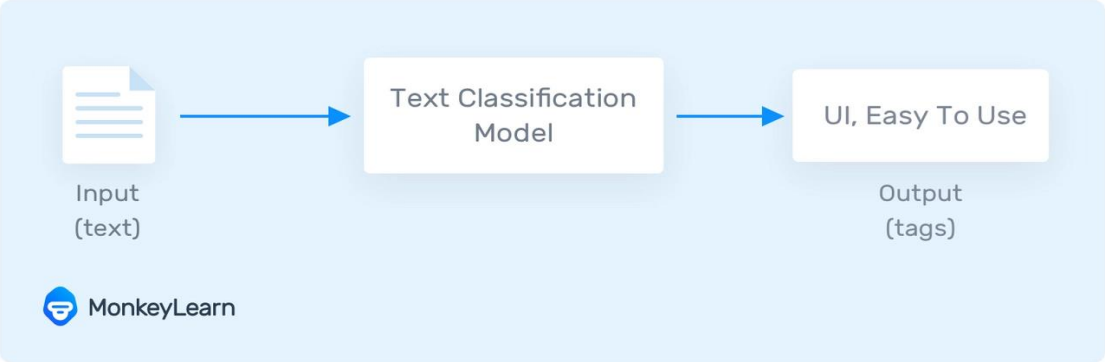


Рисунок 1.1 – Інтерфейс системи класифікації тексту

Текстовий класифікатор може прийняти цю фразу як вхідні дані, проаналізувати її вміст, потім можна призначити автоматично відповідні теги, такі як UI та Easy To Use.

За оцінками, близько 80% усієї інформації не є структурованою, а текст є одним із найпоширеніших типів неструктурованих даних. Через заплутану природу тексту аналізувати, розуміти, упорядковувати та сортувати текстові дані важко й забирає багато часу, тому більшість компаній не можуть використовувати його повною мірою.

Саме тут на допомогу приходить класифікація тексту за допомогою машинного навчання. Компанії можуть автоматично структурувати будь-який релевантний текст, використовуючи класифікатори тексту, починаючи з електронних листів, юридичних документів, соціальних мереж, чат-ботів, опитувань тощо, у швидкий і економічно ефективний спосіб. Це дозволяє компаніям економити час на аналізі текстових даних, автоматизувати бізнес-процеси та приймати бізнес-рішення на основі даних.

Деякі з основних причин для використання класифікації тексту машинного навчання:

- Масштабованість.

Аналіз і систематизація вручну є повільними та менш точними. Машинне навчання здатне ефективно проводити аналіз великих обсягів даних, таких як опитування, коментарі та електронні листи, витрачаючи при цьому мінімум ресурсів і часу, часто обробляючи інформацію за лічені хвилини. Інструменти для класифікації тексту можуть бути налаштовані так, щоб відповідати конкретним вимогам будь-якого бізнесу, незалежно від його розміру відповідно до потреб будь-якого бізнесу, великого чи малого.

- Аналіз в реальному часі.

Існують критичні ситуації, які компанії повинні якнайшвидше визначити та вжити негайних заходів (наприклад, кризи PR у соціальних мережах). Текстова класифікація за допомогою машинного навчання може постійно

стежити за згадками вашого бренду в режимі реального часу, тож ви зможете визначити важливу інформацію та мати змогу негайно вжити заходів.

- Послідовні критерії.

Люди-амататори помиляються під час класифікації текстових даних через відволікання, втому чи нудьгу, а людська суб’єктивність створює непослідовні критерії. З іншого боку, машинне навчання застосовує той самий об'єкт і критерії до всіх даних і результатів. Коли модель класифікації тексту належним чином навчена, вона працює з неперевершеною точністю [26].

1.3 Огляд існуючих аналогів

Запропоновано широкий спектр методів для розв'язання задачі класифікації текстів за допомогою автоматичних процедур. Головна мета цих методів полягає у виявленні закономірностей та класифікації великих обсягів складних структурованих даних. Можна виділити два принципово різні класи існуючих методів: методи машинного навчання та методи, що базуються на експертних знаннях, це так званий інженерний підхід.

Методи машинного навчання використовують колекцію розподілених за категоріями документів, яку попередньо проанотувала людина, для побудови класифікатора. Алгоритм машинного навчання аналізує цю колекцію та автоматично будує процедуру класифікації документів.

Методи, засновані на знаннях, використовують правила, створені експертами на основі аналізу рубрикатора та, можливо, деяких текстів, які потребують класифікації. Ці правила визначають, до якої категорії слід віднести документ.

Незважаючи на розмаїття методів класифікації текстів, актуальним завданням є вибір методу, який буде простим у реалізації, ефективним, з низькими обчислювальними витратами під час навчання та класифікації, а також забезпечуватиме високу якість класифікації в реальних сценаріях.

Аналіз тексту є методом машинного навчання, призначеним для автоматизованого витягування значущих даних із тексту, який не має чіткої структури.[35]

Коли машина аналізує текст, вона визначає ключову інформацію, що міститься у самому тексті. Проте, коли машина проводить текстовий аналіз, вона виявляє закономірності, тенденції та інші взаємозв'язки серед тисяч текстів, що дозволяє побудувати графіки, звіти, таблиці та інші аналітичні візуалізації.

На сьогоднішній день існує безліч алгоритмів та підходів до класифікації текстів з використанням методів машинного навчання.

Детальний опис аналогів алгоритмів представлено у таблиці 1.1. Детальний опис аналогів програм представлено у таблиці 1.2.

Таблиця 1.1 – Аналоги алгоритмів

Назва	**Короткий опис**
Наївний Байєсів класифікатор	Базується на теоремі Баєса та використовує статистичні методи для призначення документу певної категорії на основі ймовірностей входження слів у цю категорію.
Метод опорних векторів (SVM)	Заснований на розбитті простору ознак на дві частини за допомогою гіперплощини.
Нейронні мережі	Рекурентні нейронні мережі (RNN) та сверточні нейронні мережі (CNN). Вони здатні враховувати контекст та залежності між словами та фразами, що сприяє покращенню точності класифікації.
Дерева рішень	Алгоритми на основі дерев рішень, такі як C4.5 та Random Forest, використовують ієрархічну структуру для класифікації текстів

Глибокі згорткові мережі (DCNN)	Цей тип алгоритмів використовує комбінацію сверточних шарів та рекурентних шарів для ефективного аналізу текстових даних.

Таблиця 1.2 – Аналоги програм

Назва	**Технологія**	**Короткий опис**
Scikit-learn	Python	Бібліотека, яка надає широкий спектр алгоритмів для класифікації текстів, включаючи наївний Баєсів класифікатор, SVM, дерева рішень та інші. Scikit-learn також надає зручні інструменти для попередньої обробки текстових даних та оцінки ефективності моделей
TensorFlow	Python	TensorFlow надає багато інструментів для класифікації текстів, зокрема рекурентні нейронні мережі (RNN) та сверточні нейронні мережі (CNN). Вона дозволяє легко побудувати та тренувати моделі для класифікації текстів з високою точністю
NLTK (Natural Language Toolkit)	Python	NLTK містить набір інструментів та ресурсів

		для токенізації, лематизації, векторизації тексту та багато іншого. Вона дозволяє реалізувати алгоритми класифікації текстів з використанням різних моделей та підходів
Apache Spark MLlib	Scala, Java, Python та R	Це розширена бібліотека машинного навчання, яка працює на основі Apache Spark - системи для обробки великих обсягів даних. MLlib містить реалізації різних алгоритмів класифікації текстів, включаючи наївний Баєсів класифікатор, SVM, логістичну регресію та багато інших. Вона забезпечує можливість розподіленого обчислення для ефективної обробки великих наборів даних.

Scikit-learn – це відкрита та безкоштовна бібліотека для Python, що спеціалізується на машинному навчанні. Вона забезпечує інструменти для створення та навчання моделей класифікації, регресії та кластеризації, включно з такими методами, як лінійна регресія, випадковий ліс та градієнтний бустинг. Ця бібліотека інтегрується з NumPy та SciPy і вважається однією з найпопулярніших в області машинного навчання. [6]

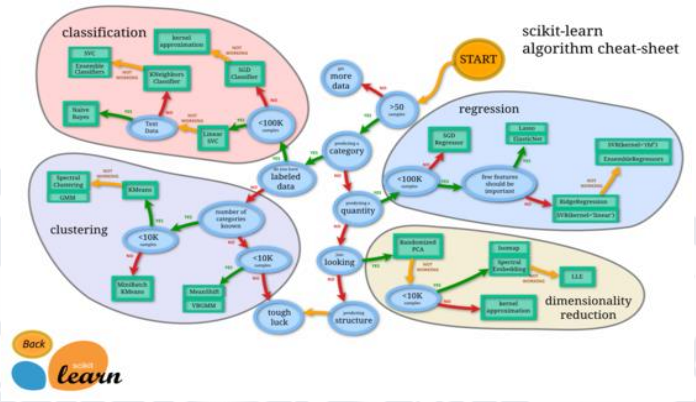


Рисунок 1.2 – Лого програми Scikit-learn

1. Scikit-learn має простий та зрозумілий інтерфейс, що дозволяє швидко і легко використовувати його для розробки моделей класифікації текстів. Вона надає гнучкість у виборі алгоритмів, підгонці параметрів та оцінці моделей.
2. Scikit-learn містить багато реалізацій алгоритмів машинного навчання для класифікації текстів, включаючи Наївний Баєсів класифікатор, Метод опорних векторів (SVM), Глибокі нейронні мережі та багато інших. Це дозволяє вибрати найбільш підходящий алгоритм для конкретної задачі.
3. Scikit-learn також надає інструменти для попередньої обробки текстових даних, таких як векторизація (наприклад, TF-IDF або Bag-of-Words), відбір ознак та нормалізація. Це дозволяє підготувати текстові дані перед подачею на вхід класифікатору.

Приклад використання Scikit-learn для задачі класифікації текстів може включати створення моделі для автоматичної класифікації позитивних та негативних відгуків на продукти. За допомогою Scikit-learn можна побудувати модель, яка використовує алгоритми, такі як Наївний Байєсів класифікатор або SVM, і навчити їх розрізняти позитивні та негативні відгуки на основі текстових ознак, таких як слова, фрази чи контекст. Ця модель може бути застосована для автоматичного аналізу та класифікації нових відгуків на продукти.

TensorFlow – це відкрита бібліотека, створена Google, яка призначена для широкого спектру завдань у сфері машинного навчання. Особливо цінною вона

є для побудови та тренування нейронних мереж, що дозволяє їм виявляти та інтерпретувати зображення, а також визначати різні види кореляцій. Ці здібності TensorFlow аналогічні до людського способу навчання та розуміння. [36]

Рисунок 1.3 – Лого програми TensorFlow

1. TensorFlow – одна з найпопулярніших бібліотек для глибокого машинного навчання та звичайного навчання. Вона розроблена командою Google Brain і має широкий спектр функцій для побудови та навчання нейронних мереж.
2. TensorFlow використовує символьне програмування, що дозволяє описувати та навчати складні глибокі нейронні мережі з багатьма шарами та різноманітними архітектурами.
3. TensorFlow надає багато модулів та інструментів для реалізації різних завдань машинного навчання, включаючи класифікацію текстів. Це включає в себе широкий набір шарів, активаційних функцій, оптимізаторів та інших компонентів, які можна використовувати для побудови моделей класифікації текстів.
4. TensorFlow підтримує графічний обчислювальний режим (Graph mode), який дозволяє оптимізувати та прискорити обчислення на графічних процесорах (GPU) або тензорних процесорах (TPU). Це робить TensorFlow дуже ефективним для тренування та застосування моделей на великих обсягах даних.

Приклад використання TensorFlow для задачі класифікації текстів може включати побудову глибокої нейронної мережі з використанням рекурентних

або згорткових шарів. Така модель може бути навчена класифікувати тексти на основі їх змісту або емоційної тональності. Наприклад, модель може бути навчена розпізнавати позитивні та негативні відгуки на продукти або класифікувати новини за їх тематикою.

NLTK (Natural Language Toolkit) – це комплекс бібліотек та програм, розроблених для обробки природної мови (NLP) у контексті англійської мови, що використовують методи символьної та статистичної обробки. Все це написано на мові програмування Python. [37]

1. NLTK – одна з найпопулярніших бібліотек для обробки природної мови (NLP). Вона надає широкий спектр інструментів та ресурсів для роботи з текстом, включаючи розпізнавання частин мови, токенізацію, стемінг, лематизацію, синтаксичний аналіз та багато іншого.
2. NLTK має велику колекцію корпусів та лексичних ресурсів, які можуть бути використані для навчання та валідації моделей NLP. Наприклад, вона містить корпуси текстів різної тематики, словники, морфологічні ресурси та інші дані, які можуть бути корисні для задач класифікації текстів.
3. NLTK надає можливості для вирішення різних задач NLP, включаючи класифікацію текстів. За допомогою NLTK можна побудувати та навчити модель класифікації текстів на основі різних алгоритмів, таких як наївний Байєсів класифікатор або метод опорних векторів.
4. NLTK також містить велику кількість прикладів та документацію, що допомагає вивчити та зрозуміти основні концепції та методики NLP. Це робить NLTK популярним вибором для студентів та дослідників, які вивчають обробку природної мови.

Приклад використання NLTK для задачі класифікації текстів може включати побудову корпусу текстових даних, попередню обробку тексту (токенізацію, лематизацію, видалення стоп-слів тощо) та навчання моделі класифікації, наприклад, за допомогою наївного Баєсового класифікатора. За допомогою NLTK можна зробити попередню обробку даних, виконати екстракцію ознак, навчити модель та оцінити її ефективність на тестовому наборі текстів.

Apache Spark – це розподілений фреймворк для машинного навчання побудований на основі Spark Core. Великою мірою завдяки розподіленій архітектурі Spark яка активно використовує оперативну пам'ять, вона в 9 разів швидша за рішення на основі дисків, які використовуються в Apache Mahout.

Рисунок 1.4 – Лого програми Apache Spark

1. Apache Spark MLlib є бібліотекою машинного навчання, яка працює на основі Apache Spark - потужного фреймворку для обробки великих обсягів даних. MLlib надає широкий спектр алгоритмів машинного навчання та інструменти для побудови моделей.
2. MLlib підтримує різні типи завдань машинного навчання, включаючи класифікацію, регресію, кластеризацію, рекомендації та інше. Це робить її універсальним інструментом для вирішення різноманітних задач аналізу даних.
3. Apache Spark MLlib підтримує розподілене навчання моделей, що дозволяє працювати з великими обсягами даних та розподіленими обчислювальними ресурсами. Це дозволяє прискорити процес навчання моделей та забезпечити масштабованість.
4. MLlib має вбудовану підтримку для обробки текстових даних, включаючи векторизацію тексту та використання згорткових нейронних мереж для аналізу тексту. Це дає змогу ефективно вирішувати задачі класифікації текстів з використанням Apache Spark MLlib.
5. Apache Spark MLlib має багатий набір інструментів для оцінки та налаштування моделей. Вона підтримує хрестову перевірку (cross-validation), метрики якості моделей, вибір гіпер параметрів та інші методи,

що допомагають забезпечити надійні та оптимальні моделі.

Приклад використання Apache Spark MLlib для задачі класифікації текстів може включати завантаження текстових даних, попередню обробку (токенізацію, векторизацію тощо), побудову моделі класифікації (наприклад, за допомогою алгоритму Random Forest або Logistic Regression) та оцінку ефективності моделі на тестових даних. В результаті можна отримати модель, яка може класифікувати нові тексти на основі вивчених закономірностей.

1.4 Наївний байєсів класифікатор

Наївний алгоритм Байєса – це метод класифікації, що використовує теорему Байєса і заснований на припущенні, що всі предиктори (ознаки) в моделі є незалежними один від одного. Іншими словами, цей метод передбачає, що присутність або відсутність будь-якої ознаки в класі не впливає на присутність інших ознак.

Наївний класифікатор Байєса – це популярний керований алгоритм машинного навчання, який використовується для таких завдань класифікації, як класифікація тексту. Він належить до сімейства алгоритмів генеративного навчання, що означає, що він моделює розподіл вхідних даних для певного класу чи категорії. Цей підхід базується на припущенні, що характеристики вхідних даних умовно не залежать від класу, що дозволяє алгоритму робити прогнози швидко й точно.

У статистиці наївні класифікатори Байєса розглядаються як прості імовірнісні класифікатори, які застосовують теорему Байєса. Ця теорема базується на ймовірності гіпотези, враховуючи дані та деякі попередні знання. Наївний класифікатор Байєса припускає, що всі ознаки у вхідних даних незалежні одна від одної, що часто не відповідає дійсності в сценаріях реального світу. Однак, незважаючи на це спрощене припущення, наївний класифікатор Байєса широко використовується через його ефективність і хорошу продуктивність у багатьох реальних програмах.

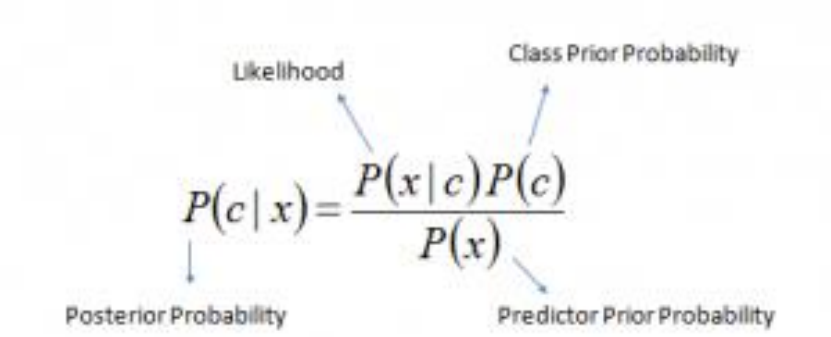
Крім того, варто зазначити, що наївні класифікатори Байєса є одними з найпростіших моделей байєсівських мереж, але вони можуть досягти високого рівня точності в поєднанні з оцінкою щільності ядра. Ця техніка передбачає використання функції ядра для оцінки функції щільності ймовірності вхідних даних, що дозволяє класифікатору покращити свою продуктивність у складних сценаріях, де розподіл даних не визначено чітко. У результаті наївний класифікатор Байєса є потужним інструментом у машинному навчанні, зокрема в класифікації тексту, фільтрації спаму та аналізі настроїв, серед іншого.

Наприклад, у випадку з наївним байєсівським класифікатором, фрукт може бути класифікований як яблуко на основі таких характеристик, як червоний колір, кругла форма та діаметр близько 3 дюймів. Цей метод припускає, що кожна з цих ознак вносить незалежний вклад у ймовірність того, що фрукт є яблуком, незалежно від того, чи існують вони окремо або залежать від інших характеристик. Саме через це припущення про незалежність ознак класифікатор отримав назву "наївний".

Модель наївного Байєса (NB) відрізняється простотою в побудові та ефективністю, особливо при роботі з великими обсягами даних. Незважаючи на свою простоту, вона часто показує кращі результати порівняно з більш складними методами класифікації.

Теорема Байєса забезпечує спосіб обчислення апостеріорної ймовірності P(c|x) із P(c), P(x) і P(x|c).

Переглянути це можна на рисунку 1.2.



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Рисунок 1.5 – Теорема Байєса

З цього можна побачити, де

P(c|x) – апостеріорна ймовірність класу (c, ціль) із заданим предиктором (x, атрибути).

P(c) – апріорна ймовірність класу.

P(x|c) – ймовірність, яка є ймовірністю даного класу предиктора.

P(x) – це пріоритетна ймовірність предиктора [16].

Типи наївного класифікатора Байєса:

- Многочлен наївного Байєса.

Ця техніка в основному застосовується для вирішення завдань класифікації документів, тобто визначення, до якої категорії належить документ, наприклад, спорту, політики, технологій тощо. Класифікатор використовує різні характеристики або прогнози, такі як частота вживання слів у документі.

- Наївний Байєс Бернуллі.

Це подібно до мультиноміального наївного Байєса, але в даному випадку предикторами є булеві змінні. Параметри, які використовуються для передбачення класу змінної, приймають лише два можливих значення - "так" або "ні", наприклад, чи присутнє слово в тексті чи ні.

- Гаусовий наївний Байєс.

Коли предиктори мають безперервні значення і не є дискретними, часто вони моделюються за допомогою розподілу Гауса.

Наївні алгоритми Байєса широко використовуються в аналізі настроїв, фільтрації спаму, системах рекомендацій і т.д. Вони є швидкими та легкими у впровадженні, але їх головним недоліком є припущення про незалежність предикторів. У більшості реальних сценаріїв предиктори можуть бути взаємозалежними, що може вплинути на ефективність класифікатора. [20].

1.5 Метод логістичної регресії

Логістична регресія – це метод статистичного аналізу для прогнозування бінарного результату, наприклад «так» або «ні», на основі попередніх спостережень за набором даних [28].

Модель логістичної регресії передбачає залежну змінну даних шляхом аналізу зв'язку між однією чи кількома існуючими незалежними змінними. Наприклад, логістичну регресію можна використати, щоб передбачити, чи виграє чи програє політичний кандидат на виборах, або чи зарахують старшокласника до певного коледжу. Ці бінарні результати дозволяють приймати прямі рішення між двома альтернативами.

Модель логістичної регресії може враховувати кілька вхідних критеріїв. У разі прийому до коледжу функція логістики може враховувати такі фактори, як середній бал студента, оцінка SAT і кількість позакласних заходів. Ґрунтуючись на історичних даних про попередні результати з тими самими вхідними критеріями, він оцінює нові випадки за ймовірністю потрапляння до однієї з двох категорій результатів.

Логістична регресія стала ключовим інструментом у галузі машинного навчання. Вона дозволяє алгоритмам, що використовуються в системах машинного навчання, класифікувати вхідні дані на основі історичних даних. З ростом обсягу доступних корисних даних алгоритми стають більш точними у передбаченні класифікацій в різних наборах даних.

Покрім того, логістична регресія може відігравати важливу роль у підготовці даних, допомагаючи впорядковувати набори даних під час процесу екстракції, трансформації і завантаження (ETL). Це дозволяє належним чином готувати дані для подальшого аналізу. [12].

Також, логістична регресія може мати суттєве значення в підготовці даних, допомагаючи розподіляти набори даних у відповідні сегменти під час процесу екстракції, трансформації і завантаження (ETL). Це дозволяє коректно підготувати інформацію для подальшого аналізу. [2].

Логістична регресія – це тип статистичної моделі, також відомий як логіт-модель, який часто використовується для проведення класифікації та прогнозної аналітики. Логістична регресія оцінює ймовірність виникнення певної події, наприклад, голосування "за" або "проти", на основі заданого набору даних незалежних змінних. Оскільки результат цієї оцінки є ймовірністю, залежна змінна обмежена в діапазоні від 0 до 1.

У логістичній регресії для перетворення шансів застосовується логіт-перетворення, що включає в себе обчислення логарифму відношення ймовірності успіху до ймовірності невдачі. Це перетворення також відоме як логарифм шансів або натуральний логарифм шансів.

Інтерпретація логістичної регресії.

Логаристичні коефіцієнти може бути важко зрозуміти в рамках аналізу даних логістичної регресії. Як результат, зведення бета-оцінок до степеня є звичайним для перетворення результатів у відношення шансів (OR), що полегшує інтерпретацію результатів. АБО представляє ймовірність того, що результат відбудеться за певної події, порівняно з ймовірністю того, що результат відбудеться за відсутності цієї події. Якщо АБО більше 1, то подія пов’язана з вищими шансами на генерування певного результату. І навпаки, якщо АБО менше 1, то подія пов’язана з нижчими шансами того, що такий результат відбудеться. Базуючись на наведеному вище рівнянні, інтерпретацію співвідношення шансів можна позначити наступним чином: шанси на успіх змінюються в exp(cB_1) разів для кожного збільшення x на одиницю c. Для прикладу скажімо, що потрібно було оцінити шанси вижити на Титаніку, враховуючи, що людина була чоловіком, а співвідношення шансів для чоловіків становить 0,0810. Це інтерпретувало б співвідношення шансів, як шанси на виживання самців зменшилися на коефіцієнт 0,0810 порівняно з самками, утримуючи всі інші змінні незмінними.

Види логістичної регресії.

Існує три типи моделей логістичної регресії, які визначаються на основі категоричної відповіді:

- Бінарна логістична регресія – це метод, де відповідь або залежна змінна приймає дихотомічний характер, тобто має лише два можливих результати, такі як 0 або 1. Цей підхід широко використовується у великій кількості задач, включаючи передбачення, чи є електронний лист спамом чи ні, чи є пухлина злоякісною чи ні.

  У контексті логістичної регресії, бінарний класифікатор є найбільш поширеним підходом і використовується для розв'язання задач бінарної класифікації .

- Мультиноміальна логістична регресія – це тип моделі логістичної регресії, в якому залежна змінна може мати три або більше можливих результатів, і ці значення не мають впорядкованої структури. Наприклад, вона може бути використана кіностудіями для передбачення, який жанр фільму виберуть кіноглядачі, щоб ефективніше просувати фільми. Модель мультиноміальної логістичної регресії допомагає студіям визначити вплив факторів, таких як вік, стать та статус особи, на вибір конкретного жанру фільму.

  За допомогою цієї моделі студія може налаштувати свою рекламну кампанію для конкретного фільму, спрямовуючи її на групу людей, які, ймовірно, виберуть цей фільм для перегляду, враховуючи їхні особисті характеристики..

- Порядкова логістична регресія – це тип моделі логістичної регресії, який використовується, коли змінна відповіді має три або більше можливих результатів, і ці значення мають певний порядок. Прикладами порядкових відповідей можуть бути шкали оцінювання від A до F або шкали оцінок від 1 до 5, де є чітка ієрархія або послідовність між різними рівнями відповідей.

  Цей тип моделі дозволяє враховувати цей порядок при аналізі та передбаченні результатів. Такі моделі можуть бути корисними в

різних галузях, де важливо визначити вплив різних факторів на порядок або ранг якогось явища або події [28].

1.6 Дерево ухвалення рішень

Дерево рішень – це метод машинного навчання, який можна використовувати як для задач класифікації, так і для задач регресії, хоча він зазвичай найкраще справляється з задачами класифікації. Це виглядає як деревоподібний класифікатор, де внутрішні вузли представляють характеристики набору даних, гілки представляють правила для прийняття рішень, а кожен листовий вузол представляє собою результат [29].

У дерева рішень є два типи вузлів: вузли рішень і листові вузли. Вузли рішень використовуються для прийняття рішень і мають кілька гілок, які представляють альтернативи. Листові вузли є результатами цих рішень і не мають гілок.

Рішення або перевірки виконуються на основі характеристик даного набору даних. Дерево рішень надає графічне представлення для отримання всіх можливих рішень на основі заданих умов. Воно може бути використане для класифікації об'єктів або прогнозування значень, залежно від конкретної задачі.

Це називається деревом рішень, оскільки, подібно до дерева, воно починається з кореневого вузла, який розширюється на подальші гілки та створює деревоподібну структуру.

Щоб побудувати дерево, ми використовуємо алгоритм CART, який розшифровується як алгоритм дерева класифікації та регресії.

Дерево рішень просто задає запитання, а на основі відповіді (Так/Ні) розбиває дерево на піддерева.

Дерево рішень є одним із найпотужніших інструментів алгоритмів машинного навчання, який використовується як для задач класифікації, так і для регресії. Цей метод створює структуру дерева, де внутрішні вузли представляють перевірку атрибутів, кожна гілка представляє результат цієї перевірки, а кожен листовий вузол містить мітку класу або значення регресії.

Він створюється шляхом рекурсивного розбиття навчальних даних на підмножини на основі значень атрибутів, доки не буде виконано критерій зупинки, такий як максимальна глибина дерева або мінімальна кількість зразків, необхідних для розбиття вузла.

Під час навчання алгоритм дерева рішень вибирає найкращий атрибут для поділу даних на основі таких показників, як ентропія або домішка Джині, яка вимірює рівень домішок або випадковості в підмножинах. Мета полягає в тому, щоб знайти атрибут, який максимізує приріст інформації або зменшення домішок після поділу.

Дерево рішень – це структура дерева, подібна до блок-схеми, де кожен внутрішній вузол позначає функцію, гілки позначають правила, а листові вузли позначають результат алгоритму. Це універсальний алгоритм машинного навчання, який є контрольований. Він використовується як для задач класифікації, так і для регресії. Це один із найпотужніших алгоритмів. Він також використовується в Random Forest для навчання на різних підмножинах навчальних даних, що робить випадковий ліс одним із найпотужніших алгоритмів машинного навчання.

Дерево рішень – це структура дерева, подібна до блок-схеми, де кожен внутрішній вузол позначає функцію, гілки позначають правила, а листові вузли позначають результат алгоритму. Це універсальний контрольований алгоритм машинного навчання, який використовується як для задач класифікації, так і для регресії. Це один із найпотужніших алгоритмів. Він також використовується в Random Forest для навчання на різних підмножинах навчальних даних, що робить випадковий ліс одним із найпотужніших алгоритмів машинного навчання.

Термінологія дерева рішень:

- Кореневий вузол: це найвищий вузол у дереві, який представляє повний набір даних. Це відправна точка процесу прийняття рішень.

- Рішення/внутрішній вузол: вузол, який символізує вибір щодо вхідної функції. Відгалуження внутрішніх вузлів з'єднує їх із листовими вузлами або іншими внутрішніми вузлами.
- Листовий/кінцевий вузол: вузол без дочірніх вузлів, який вказує на мітку класу або числове значення.
- Поділ: процес поділу вузла на два або більше підвузлів за допомогою критерію поділу та вибраної функції.
- Гілка/піддерево: Підрозділ дерева рішень починається у внутрішньому вузлі та закінчується кінцевими вузлами.
- Батьківський вузол: вузол, який ділиться на один або кілька дочірніх вузлів.
- Дочірній вузол: вузли, які виникають, коли батьківський вузол розділено.
- Домішка: вимірювання однорідності цільової змінної в підмножині даних. Це стосується ступеня випадковості або невизначеності в наборі прикладів. Індекс Джіні та ентропія є двома часто використовуваними вимірюваннями домішок у деревах рішень для завдання класифікації
- Дисперсія: дисперсія вимірює, наскільки прогнозовані та цільові змінні відрізняються в різних вибірках набору даних. Він використовується для задач регресії в деревах рішень. Середня квадратична помилка, середня абсолютна помилка, friedman_mse або відхилення половини Пуассона використовуються для вимірювання дисперсії для завдань регресії в дереві рішень.
- Інформаційний приріст: інформаційний приріст є мірою зменшення домішок, досягнутого шляхом поділу набору даних на певну функцію в дереві рішень. Критерій розбиття визначається ознакою, яка забезпечує найбільший приріст інформації. Він використовується для визначення найбільш інформативної ознаки

для розбиття на кожному вузлі дерева з метою створення чистих підмножин

- Обрізка: процес видалення гілок з дерева, які не надають жодної додаткової інформації або призводять до надмірного облаштування.

Побудова дерева рішень: дерево можна «вивчити», розділивши вихідний набір на підмножини на основі мір вибору атрибутів. Міра вибору атрибутів (ASM) – це критерій, який використовується в алгоритмах дерева рішень для оцінки корисності різних атрибутів для поділу набору даних. Головна мета ASM полягає в тому, щоб визначити атрибут, який призведе до створення найбільш однорідних підмножин даних після поділу, тим самим максимізуючи приріст інформації або інші критерії, що використовуються для покращення якості моделі. Вибір оптимального атрибуту для поділу грає важливу роль у побудові ефективного дерева рішень, яке може точно класифікувати або прогнозувати дані.

Цей процес повторюється для кожної похідної підмножини даних за допомогою рекурсивного розбиття. Рекурсія завершується, коли всі дані в підмножині мають однакове значення цільової змінної або коли подальше розбиття не приносить користі у покращенні прогнозів. Для побудови класифікатора дерева рішень не потрібні знання предметної області чи налаштування параметрів, і тому він підходить для дослідницького виявлення знань. Дерева рішень можуть обробляти дані великого розміру.

Схема нижче пояснює загальну структуру дерева рішень.

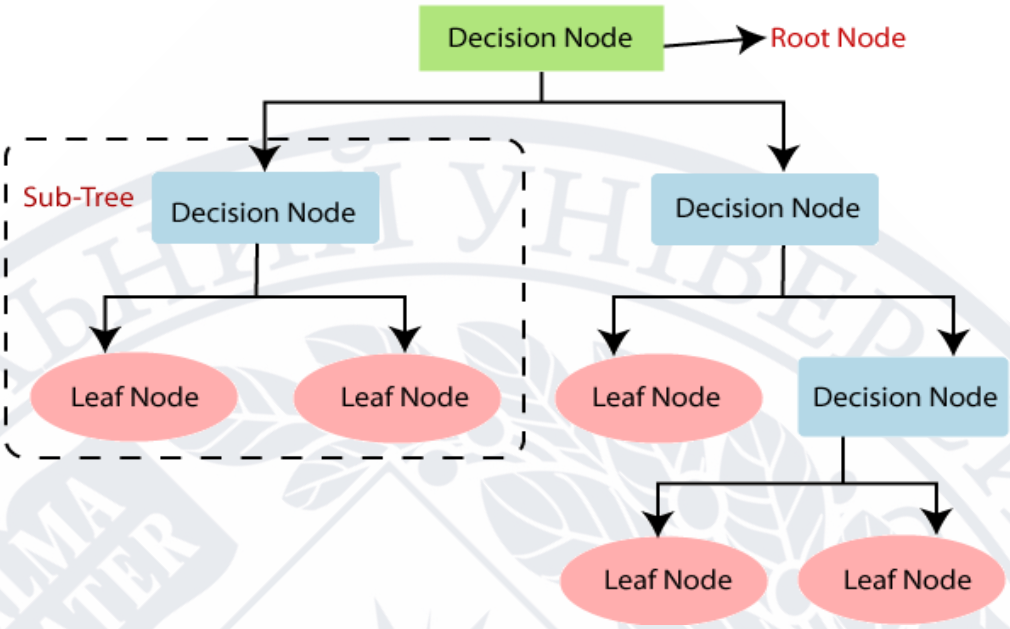


Рисунок 1.6 – Приклад дерева ухвалення рішень

Ентропія.

Ентропія є мірою ступеня випадковості або невизначеності в наборі даних. У випадку класифікацій, він вимірює випадковість на основі розподілу міток класів у наборі даних.

Ентропію для підмножини вихідного набору даних, що має K кількість класів для i-го вузла, можна визначити як:

$$H_i = -\sum_{k\epsilon K}^{n} \quad p(i,k)p(i,k)\ ,$$

де

S – зразок набору даних.

k – окремий клас із K класів

p(k) – частка точок даних, які належать до класу k, до загальної кількості точок даних у вибірці набору даних S.

$$p(k) = \frac{1}{n}\sum \qquad I(y = k).$$

Тут p(i,k) не має дорівнювати нулю.

Важливі моменти, пов’язані з ентропією:

1. Ентропія дорівнює 0, коли набір даних повністю однорідний, тобто кожен екземпляр належить до одного класу. Це найнижча ентропія, яка вказує на відсутність невизначеності у вибірці набору даних.
2. Коли набір даних порівну розподілено між декількома класами, ентропія має максимальне значення. Таким чином, ентропія найвища, коли розподіл міток класів рівномірний, що вказує на максимальну невизначеність у вибірці набору даних.
3. Ентропія використовується для оцінки якості поділу. Метою ентропії є вибір атрибута, який мінімізує ентропію результуючих підмножин шляхом поділу набору даних на більш однорідні підмножини щодо міток класу.
4. Найвищий атрибут приросту інформації вибирається як критерій поділу (тобто, зменшення ентропії після поділу за цим атрибутом), і процес повторюється рекурсивно для побудови дерева рішень.

Дерево рішень працює шляхом аналізу набору даних для прогнозування його класифікації. Він починається з кореневого вузла дерева, де алгоритм переглядає значення кореневого атрибута в порівнянні з атрибутом запису в фактичному наборі даних. На основі порівняння він продовжує слідувати гілці та переходити до наступного вузла.

Алгоритм повторює цю дію для кожного наступного вузла, порівнюючи його значення атрибутів зі значеннями підвузлів і продовжуючи процес далі.

Це повторюється, поки не досягне вузла листа дерева. Повний механізм можна краще пояснити за допомогою наведеного нижче алгоритму:

Крок 1: Потрібно почати дерево з кореневого вузла, каже S, який містить повний набір даних.

Крок 2. Знайти найкращий атрибут у наборі даних за допомогою вимірювання вибору атрибутів (ASM).

Крок 3: Розділити S на підмножини, які містять можливі значення для найкращих атрибутів.

Крок 4: Створити вузол дерева рішень, який містить найкращий атрибут.

Крок 5: Рекурсивно створювати нові дерева рішень, використовуючи підмножини набору даних, створеного на кроці -3. Продовжувати цей процес, доки не досягається стадії, на якій не зможе далі класифікувати вузли, і остаточний вузол не називатиметься листовим вузлом Класифікація та алгоритм регресійного дерева.

Переваги дерева рішень:

- Його легко зрозуміти, оскільки він слідує тому ж процесу, якому дотримується людина, приймаючи будь-які рішення в реальному житті.
- Він може бути дуже корисним для вирішення проблем, пов'язаних з прийняттям рішень.
- Він допомагає подумати про всі можливі наслідки проблеми.
- Порівняно з іншими алгоритмами вимагається менше очищення даних.

Недоліки дерева рішень:

- Дерево рішень містить багато шарів, що робить його складним.
- Можливо, проблема переобладнання, яку можна вирішити за допомогою алгоритму випадкового лісу.
- Для більшої кількості міток класу обчислювальна складність дерева рішень може зрости.

Незважаючи на те, що було розроблено різноманітні методи навчання дерева рішень із дещо різними можливостями та вимогами, навчання дерева рішень, як правило, найкраще підходить для проблем із такими характеристиками:

1. Екземпляри представлені парами атрибут-значення:

У світі вивчення дерева рішень ми зазвичай використовуємо пари атрибут-значення для представлення екземплярів. Екземпляр визначається заздалегідь визначеною групою атрибутів, як-от температура, і відповідним значенням, як-

от гаряче. В ідеалі ми хочемо, щоб кожен атрибут мав кінцевий набір різних значень, наприклад гаряче, м’яке або холодне. Це полегшує побудову дерева рішень. Однак більш просунуті версії алгоритму можуть включати атрибути з безперервними числовими значеннями, наприклад представлення температури за допомогою числової шкали.

2. Цільова функція має дискретні вихідні значення:

Позначена мета має чіткі результати. Метод дерева рішень зазвичай використовується для категоризації булевих прикладів, наприклад, так чи ні. Підходи до дерева рішень можна легко розширити для отримання функцій із подвійними мислимими значеннями результатів. Більш суттєве розширення дозволяє отримати знання про поставлені цілі за допомогою числових результатів, хоча практика дерев рішень у цій структурі є порівняно рідкісною.

3. Можуть знадобитися розділові описи:

Дерева рішень природно представляють диз'юнктивні вирази.

4. Навчальні дані можуть містити помилки:

«Методики навчання дерева рішень демонструють високу стійкість до розбіжностей, включаючи невідповідності в категоризації зразків випадків і розбіжності в деталях характеристик, які характеризують ці випадки».

5. Навчальні дані можуть містити відсутні значення атрибутів:

У деяких випадках вхідна інформація, призначена для навчання, може мати відсутні характеристики. Використання підходів дерева рішень все ще можливо, незважаючи на наявність невідомих функцій у деяких навчальних зразках. Наприклад, якщо розглядати рівень вологості протягом дня, ця інформація може бути доступною лише для певного набору тренувальних зразків [8].

1.6 Метод опорних векторів

Машини опорних векторів (SVM) представляють собою набір контрольованих методів навчання, які використовуються для вирішення завдань класифікації, регресії та виявлення викидів [25].

SVM, або машина опорних векторів, є одним із найпопулярніших алгоритмів навчання з учителем і зазвичай використовується для вирішення завдань класифікації та регресії в області машинного навчання. Однак головним застосуванням SVM є задачі класифікації.

Машина опорних векторів (SVM) – це дискримінаційний класифікатор, формально визначений розділювальною гіперплощиною. Іншими словами, враховуючи позначені навчальні дані (контрольоване навчання), алгоритм виводить оптимальну гіперплощину, яка класифікує нові приклади. У двовимірному просторі ця гіперплощина є лінією, що розділяє площину на дві частини, де кожен клас лежить по обидві сторони [22].

Метою алгоритму SVM є створення оптимальної лінії або границі рішення, яка може розділити n-вимірний простір на класи так, щоб ми могли ефективно класифікувати нові точки даних у правильні категорії в майбутньому. Ця границя рішення називається гіперплощиною.

SVM вибирає екстремальні точки або вектори, які допомагають визначити гіперплощину. Ці особливі вектори називаються опорними векторами, і сам алгоритм отримав назву "машина опорних векторів" через їх важливу роль у процесі. На прикладі діаграми, де є дві різні категорії, SVM шукає таку гіперплощину, яка найкраще розділить ці категорії та має максимальний зазор між ними:

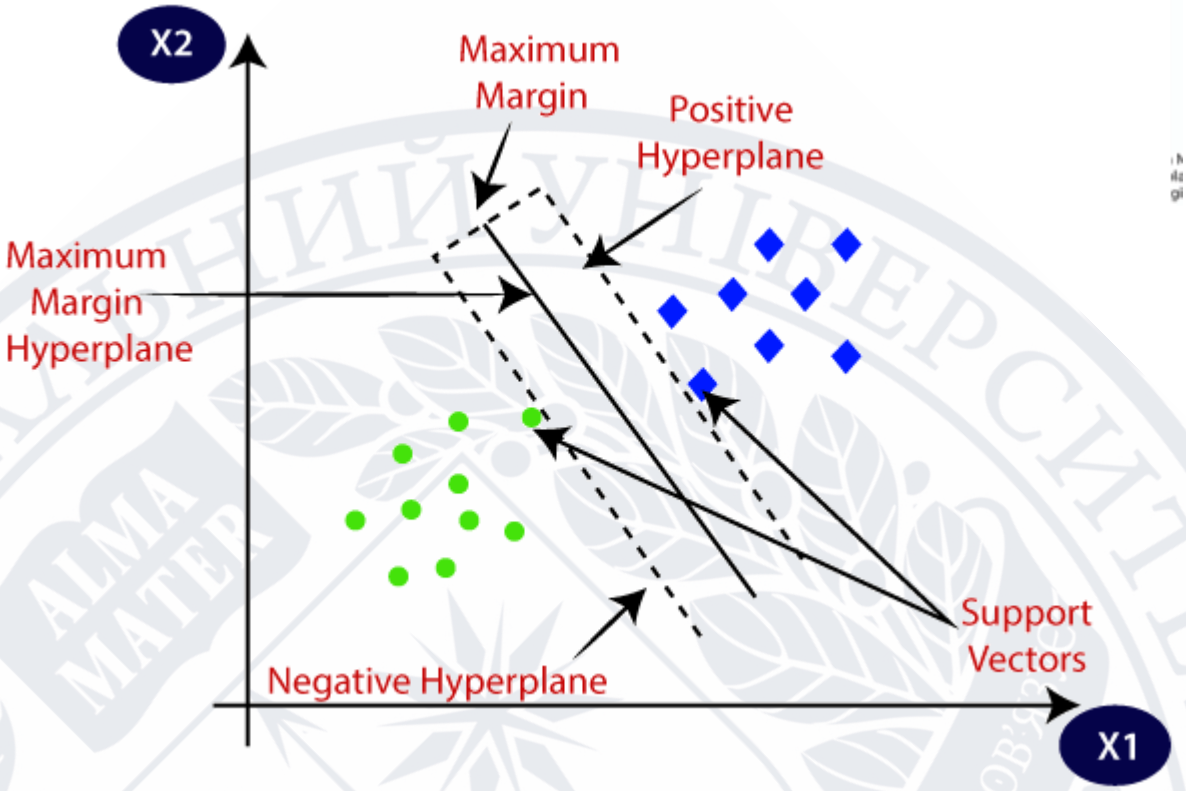


Рисунок 1.7 – Класифікція двох категорій методом опорних векторів

SVM може бути двох типів:

- Лінійний SVM: Використовується для лінійно роздільних даних, тобто таких даних, де можна провести пряму лінію, яка чітко розділить два класи. Це підходить для ситуацій, коли дані мають простий лінійний патерн розділення між класами. Лінійний SVM використовує лінійну гіперплощину для класифікації даних.
- Нелінійний SVM: Використовується для нелінійно роздільних даних, тобто даних, де класи не можна точно розділити за допомогою прямої лінії. В таких ситуаціях нелінійний SVM використовує нелінійні функції відображення для перетворення дані в вищорозмірний простір, де класи можуть бути лінійно розділені. Цей підхід дозволяє розв'язувати складніші завдання класифікації, де лінійна гіперплощина була б непридатною. [24]

Перевагами опорних векторних машин є:

- Ефективний у великих просторах.

- Все ще ефективний у випадках, коли кількість вимірів перевищує кількість зразків.
- Підмножину тренувальних точок у функції прийняття рішення (опорні вектори), через це він є ефективним для пам’яті.
- Універсальність: для функції прийняття рішень можна вказувати різні функції ядра. Також можуть надаватись загальні ядра, але можна й вказати власні ядра.

Недоліки опорних векторних машин:

- Коли кількість функцій є набагато більшою за кількість зразків, потрібно уникати надмірної підгонки у виборі функцій ядра, а термін регуляризації матиме вирішальне значення.
- SVM не надають оцінки ймовірності безпосередньо, вони можуть обчислюватися за допомогою п’ятикратної перехресної перевірки.

Модуль scikit-learn підтримує обробку як щільних (представлених у формі numpy.ndarray і конвертованих до них за допомогою numpy.asarray), так і розріджених (в будь-якому форматі scipy.sparse) векторів вибірки як вхідних даних для методу опорних векторів (SVM). Однак, щоб використовувати SVM для передбачення на розріджених даних, вектори даних повинні бути у відповідному форматі. Для досягнення максимальної ефективності рекомендується використовувати щільні numpy.ndarray (dense) або розріджені scipy.sparse.csr_matrix (sparse) з типом даних float64. [25]

Висновок до розділу 1

У цьому розділі було описано постановку задачі роботи, проведено огляд існуючих алгоритмів та проаналізовано літературу за даною тематикою, а також були окреслені особливості обраних алгоритмів. Наступний розділ буде присвячений вибору інструментів для розробки.

РОЗДІЛ 2
ОГЛЯД ІНСТРУМЕНТІВ ДЛЯ РОЗРОБКИ

Даний розділ присвячений опису використаного стеку технологій, які були обрані для дослідження алгоритмів.

2.1 Вибір інструментарію і технічної платформи

Для реалізації алгоритмів класифікації обрано мову програмування Python.

Python – це вищорівнева, загальнопризначкова мова програмування, яка користується великою популярністю. Мову програмування Python (зокрема, останню версію Python 3) широко використовують у веб-розробці, у сфері машинного навчання та інших передових галузях програмного забезпечення.

Рисунок 2.1 – Логотип мови програмування Python

Python використовується майже всіма гігантами технологій: Google, Amazon, Facebook, Instagram, Dropbox, Uber і т.п.

Найбільша перевага Python – величезна колекція стандартної бібліотеки, яку можна використовувати для:

1. Машинне навчання
2. Програми GUI (наприклад, Kivy, Tkinter, PyQt тощо)
3. Веб-фреймворки, такі як Django (використовуються YouTube, Instagram, Dropbox)
4. Обробка зображень (наприклад, OpenCV, Pillow)
5. Веб-збирання (наприклад, Scrapy, BeautifulSoup, Selenium)
6. Тестові рамки
7. Мультимедіа
8. Наукові обчислення
9. Обробка тексту та багато іншого.

Python – найпопулярніша та високорівнева, загальнопризначкова мова програмування, яка підтримує об'єктно-орієнтовану та процедурну парадигми. Програми, які написані через Python, зазвичай мають менший обсяг коду порівняно з іншими мовами, такими як Java. Це дозволяє програмістам вводити менше коду також суворі вимоги до відступів у мові сприяють підвищенню читабельності коду усього часу. [19]

Можливості Python:

- Використання на сервері для створення веб-додатків.
- Використання разом із ПО для створення робочих процесів.
- Підключення до систем БД. Він також може читати та змінювати файли.
- Використання для обробки великих даних і виконання складної математики.
- Використання для швидкого створення прототипів або для розробки готового ПО.

Чому Python?

- Працює на різних платформах (Windows, Mac, Linux, Raspberry Pi тощо).

- Має простий синтаксис, а також подібний до англійської мови.
- Синтаксис дозволяє розробникам писати програми з меншою кількістю рядків.
- Працює в системі інтерпретатора, отже, код може бути виконаний відразу після його написання.
- Можна розглядати процедурно, об'єктно-орієнтовано чи функціонально.

Синтаксис Python порівняно з іншими мовами програмування

- Розроблений для читабельності та має подібності до англійської мови з впливом математики.
- Використовує нові рядки для завершення команди, інші мови програмування використовують крапку з комою або ж круглі дужки.
- Покладається на відступи, використовуючи пробіли, щоб визначити область: область циклів, функцій чи класів. Інші мови програмування часто використовують для цієї мети фігурні дужки [19].

Для запуску алгоритмів класифікації використовуємо Jupiter Book.

Jupyter Book — це обгортка навколо набору інструментів в екосистемі Python, які спрощують публікацію обчислювальних документів. Він використовує мову MyST Markdown у Markdown і документах блокнота. Це дозволяє користувачам писати у своїх документах насичену розмітку якості публікації.

Jupyter Book — це проект із відкритим вихідним кодом для створення красивих книг, веб-сайтів і документів, які придатні для публікації, із вихідного матеріалу, який містить обчислювальний вміст. [1]

Ось кілька особливостей Jupyter Books:

- Інтерфейс командного рядка (CLI) для швидкого створення, створення та оновлення книг.
- Записуйте вміст книги в Markdown і Jupyter Notebooks

- Перетворіть їх на сторінки Jekyll, які можна безкоштовно розмістити на GitHub
- На сторінки можуть бути автоматично додані посилання Binder, JupyterHub або Thebelab для інтерактивності.
- Сам веб-сайт заснований на Jekyll і є дуже розширюваним.
- Під капотом є багато чудових функцій HTML, таких як швидка навігація Turbolinks і копіювання клацанням у клітинках коду.[31]

Для роботи з Jupyter Book можна використати або Colab Notebooks, або Kaggle.

Рисунок 2.2 – Логотип Colab Notebooks

Colaboratory, або просто Colab, дозволяє писати та виконувати код Python у браузері. При цьому:

- не потрібно ніякого налаштування;
- безкоштовний доступ до графічних процесорів.

Надавати доступ до документів іншим людям вкрай просто.

Це відмінне рішення для студентів, фахівців з обробки даних та дослідників у галузі штучного інтелекту.

У записниках Colab є можливість поєднувати код, що виконується разом з форматованим текстом в одному документі, а також можна додавати зображення, HTML, LaTeXi т.д.. При створенні власних записників Colab, вони зберігаються в обліковому записі Google Диска. Доступ можна шарити, тобто надавати доступ до цих записників співробітникам чи друзям – так вони матимуть змогу коментувати або ж навіть редагувати їх.

З Colab можна отримати доступ до всіх популярних бібліотек Python для аналізу й візуалізації даних.

Можна імпортувати власні дані в записники Colab з облікового запису Google Диска (зокрема, з електронних таблиць), а також із GitHub і багатьох інших джерел.

Машинне навчання.

У Colab є можливість імпортувати набір даних, як зображення і навчати класифікатор, базуючись на їх основі. Оцінювати модель за допомогою лише кількох рядків коду. Записники Colab виконують код на хмарних серверах Google. Отже, можна застосувати більшість можливостей апаратного забезпечення Google, зокрема графічні й тензорні процесори, незалежно від потужності комп'ютера. Потрібен лише веб-переглядач.

Colab широко застосовується у сфері машинного навчання для таких завдань:

- початок роботи з TensorFlow;
- розробка й навчання нейронних мереж;
- експерименти з тензорними процесорами;
- поширення досліджень у сфері ШІ;
- створення навчальних посібників [6].

Також для роботи з машинним навчанням можна використовувати сервіс kaggle.

Kaggle являє собою онлайн-спільноту для науковців обробки даних та ентузіастів машинного навчання. Kaggle дозволяє більшості користувачам співпрацювати між собою, знаходити і публікувати набори даних, використовувати записники з інтегрованим графічним процесором, а також конкурувати з іншими дослідниками, які досліджують дані для вирішення завдань науки про дані [30].

Kaggle виділяється серед відомих платформ даних, визначаючи та підвищуючи стандарти науки про дані. Завдяки своєму винятковому дизайну та прагненню сприяти інноваціям, співпраці та обміну знаннями, Kaggle

перетворився на важливий центр як для відомих, так і для початківців науковців із обробки даних.

Kaggle — це платформа для змагань із наукових даних та онлайн-спільнота вчених із обробки даних і практиків машинного навчання. Належить Google, наразі це найкраща платформа, яка об'єднує спільноту машинного навчання та науки про дані [31].

Успіх Kaggle був висвітлений у засобах масової інформації, оскільки його платформа продовжує залучати мільйони дослідників даних до вирішення складних проблем. Це дозволило платформі залишатися в авангарді науки про дані, надихаючи наступне покоління вчених з обробки даних.

Рисунок 2.3 – Логотип Kaggle

Мета даної платформи (вона була заснована в 2010-му році Ентоні Голдблюмом та Джеремі Говардом і придбаною компанією Google у 2017 році) полягає в тому, щоб допомогти професіоналам та учням досягти цілей, які вони поставили у своїй подорожі наукою про дані за допомогою найпотужніших інструментів та ресурсів, які вона може надати. На даний момент користувачів цієї платформи понад 8 мільйонів.

Однією з підплатформ, яка зробила Kaggle таким популярним ресурсом, є їх змагання. Подібно до того, як HackerRank відіграє цю роль для розробників програмного забезпечення та комп'ютерних інженерів, «Змагання Kaggle»

мають важливе значення для дослідників даних; можна дізнатися більше у Посібнику зі змагань Kaggle, а також навчитися крок за кроком аналізувати набір даних Підручнику зі змагань Kaggle.

У змаганнях з обробки даних, таких як Kaggle або DataCamp, компанії та організації діляться великою кількістю складних завдань із науки про дані зі щедрими винагородами, у яких дослідники даних, від початківців до досвідчених, змагаються за їх виконання. Kaggle також надає Kaggle Notebook, який, так само як і DataCamp Workspace, дозволяє редагувати та запускати код для завдань з обробки даних у браузері, тому локальному комп'ютеру не потрібно виконувати всю важку роботу, і не потрібно самостійно налаштувати нове середовище розробки.

Kaggle надає потужні ресурси в хмарі та дозволяє використовувати максимум 30 годин GPU та 20 годин TPU на тиждень. Можна завантажувати свої набори даних у Kaggle, а також завантажувати набори даних інших користувачів. Крім того можна перевірити набори даних і блокноти інших людей та почати обговорення на них. Уся активність оцінюється на платформі, і рахунок збільшується, коли користувач допомагає іншим та ділитеся корисною інформацією. Щойно користувач почнете заробляти очки, він потрапить до таблиці лідерів із мільйонами користувачів Kaggle.

Kaggle підходить для різних груп людей: від студентів, які цікавляться наукою про дані та штучним інтелектом, до найдосвідченіших науковців у світі. Якщо новачок, він може скористатися курсами, які надає Kaggle. Приєднавшись до цієї платформи, він матиме змогу розвиватися в спільноті людей різного рівня знань і матимете можливість спілкуватися з багатьма досвідченими науковцями з обробки даних. Коли почне заробляти бали та медалі Kaggle, які є доказом вашого прогресу, цілком можливо, що він навіть матиме змогу залучити хедхантерів і рекрутерів і відкриєте нові можливості роботи.

І останнє, але не менш важливе: коли користувач подає заявку на роботу в галузі обробки даних, згадка про досвід роботи з Kaggle безперечно має позитивний вплив. Зрозуміло, що всі ці переваги також стосуються досвідчених

спеціалістів із обробки даних. Незалежно від того, наскільки він досвідчений, ця платформа пропонує постійне навчання та можливості для вдосконалення, і, звичайно, грошові винагороди, які можуть прийти під час змагань, не менш цікаві [30].

2.2 Програмні модулі, їх взаємозв'язки та опис

Для реалізації алгоритму потрібно спочатку підключити бібліотеки і модулі, які допоможуть реалізувати класифікацію.

Для того, щоб отримати доступ до датасету потрібно спочатку зайти до папки із завантаженими файлами, а потім обрати потрібний файл:

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
news_data = pd.read_csv('/kaggle/input/bbc-fulltext-and-category/bbc-text.csv').
```

Також використовуються функції для роботи з датасетом:

news_data.shape - виводить розмір датасету;

news_data.columns – виводить які колонки є в дата сеті;

news_data.category.unique() – яка виводить які унікальні категорії є в дадасеті;

news_data.head().append(news_data.tail()) – виведення перших та останніх 5 рядків даних, які знаходяться в дата сеті.

Для побудови графіка по категоріям використовується фрагмент коду:

```
# Plot category data
plt.figure(figsize=(10,6))
sns.countplot(news_data.category)
plt.show()
```

Інші функції використовуються для роботи з текстом:

self.tokenize() – розбиття речень на слова;

self.remove_stopwords() – видалення стоп слів;

self.remove_non_words() – видалення символів, які не відносяться до тексту

self.lemmatize_words() – групування відмінювання слів, для того, щоб їх використовувати як одне.

Використовують Word2Vec для того, щоб зробити векторний простір.

Word2Vec – це прогностична модель для вивчення вставок слів із необробленого тексту. Він приймає як вхідні дані великий корпус слів і створює векторний простір, як правило, з кількох сотень вимірів, і навіть кожному слову, що є унікальним у корпусі назначається відповідний вектор у просторі.

TF-IDF (TF - term frequency, IDF - inverse document frequency) – є статистичною мірою, яка використовується для оцінювання важливості слова в самому контексті документа. Вага деякого слова прямо пропорційна частоті вживання саме цього слова в документі і є обернено пропорційною частоті вживання слова у всіх документах колекції.

Міра TF-IDF часто використовується в задачах аналізу текстів та інформаційного пошуку, наприклад, як один із критеріїв релевантності документа пошуковому запиту, при розрахунку міри близькості документів під час кластеризації [29].

Висновок до розділу 2

У даному розділі було описано стек технологій, які використовувались для роботи з алгоритмами, а також описані основні функції. Наступний розділ буде присвячений опису розробленого продукту.

РОЗДІЛ 3
РОБОТА З АЛГОРИТМАМИ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ

Даний розділ присвячений опису алгоритмів та роботі з ними.

3.1 Тестування алгоритмів

Для тестування алгоритму потрібні дані, на яких можна його навчити. Такі дані можна взяти із сайту kaggle.

Kaggle Datasets дозволяє публікувати та ділитися наборами даних приватно чи публічно. Надає ресурси для зберігання та обробки наборів даних. Kaggle підтримує різноманітні формати публікації наборів даних [23].

З kaggle можна завантажити датасет або взяти посилання на нього. Задача зроблена на основі навчених та ненавчених моделей.

Навчені моделі — це моделі, які були треновані на великій кількості даних і вже мають здатність робити передбачення або приймати рішення на основі цього навчання. Тренування моделі включає в себе процес адаптації моделі до даних за допомогою алгоритму оптимізації, щоб мінімізувати помилку передбачення. Ось основні метрики, які використовуються для оцінювання навчених моделей:

- Точність (Accuracy): Відсоток випадків, коли модель правильно класифікує або передбачає.
- Прецизійність (Precision): Здатність моделі не класифікувати як позитивний зразок той, який є негативним.
- Виклик (Recall): Здатність моделі знайти всі позитивні зразки.
- F1-бал (F1-Score): Гармонійне середнє прецизійності та виклику, корисне коли потрібно збалансувати обидві метрики.

Ненавчені моделі — це моделі, які не були експоновані до жодних даних і тому не мають здатності робити обґрунтовані передбачення. Їхні результати зазвичай дуже низькі і близькі до випадкового вибору. Ось як можна охарактеризувати їхні передбачення:

- Випадкові передбачення: Якщо датасет збалансований, ненавчена модель матиме точність близько 1/кількість_класів (для 5 класів це 20%).
- Відсутність узагальнення: Модель не може узагальнювати з даних, тому її передбачення не мають статистичної ваги.
- Відсутність навичок класифікації: Модель не може розрізняти між класами, тому всі класи для неї однакові.

Порівняльний аналіз між навченими та ненавченими моделями дозволяє нам зрозуміти важливість навчання та якість даних. Навчання моделей дає змогу побачити, як алгоритми машинного навчання адаптуються до патернів у даних і покращують свої передбачення. Ненавчені моделі слугують базовою лінією або контрольною точкою для вимірювання цього покращення.

Висновки. Загалом, порівняння показує, що навчання моделі є критично важливим для її здатності робити точні та відповідальні передбачення. Моделі, які не пройшли процес навчання, не можуть ефективно вирішувати задачі машинного навчання, такі як класифікація текстів.

Задача класифікації текстів вирішуватиметься на прикладі датасету з текстами зі статтями BBC. Датасет містить 2225 текстів, розділених на 5 категорій, зображених на рис.3.1.

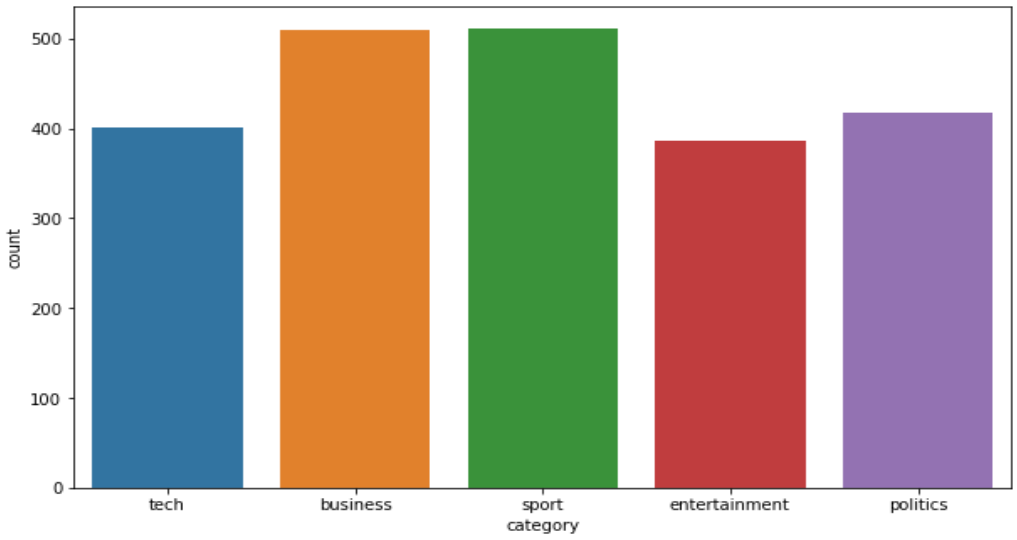


Рисунок 3.1 – Гістограма розподілення кількості текстів

Із рисунку видно, що датасет містить такі категорії: business, politics, entertainment, sport, tech.

Так як, алгоритм запускається в Colab Notebooks, то потрібно кожен фрагмент коду запускати окремо.

Тобто, спочатку потрібно запустити фрагменти підключення бібліотеки й отримуємо такий результат:

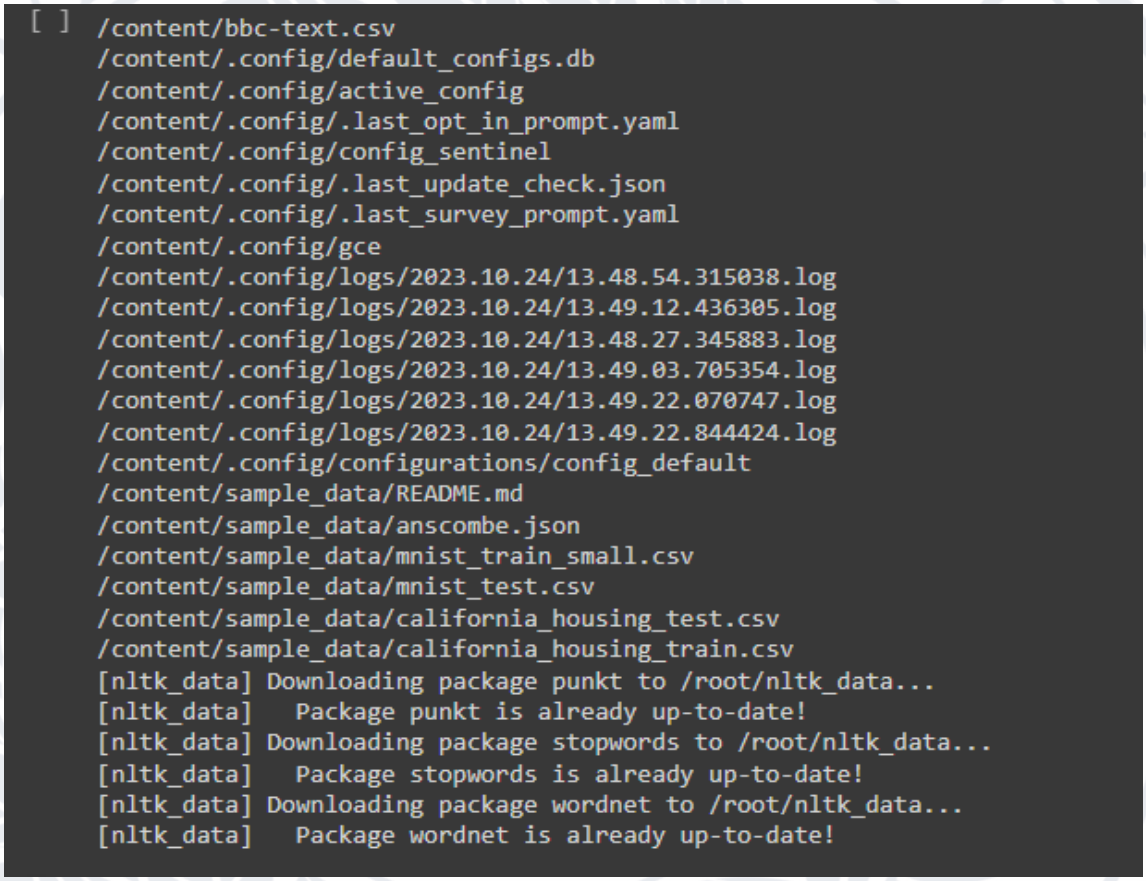


Рисунок 3.2 — Завантаження модулів і бібліотек

Потім потрібно завантажити датасет та подивитись інформацію про нього:

```
[ ] news_data = pd.read_csv('/content/bbc-text.csv')

print(f"Shape : {news_data.shape}, \n\nColumns: {news_data.columns}, \n\nCategories: {news_data.category.unique()}")

# print sample data
news_data.head().append(news_data.tail())
```

```
Shape : (2225, 2),

Columns: Index(['category', 'text'], dtype='object'),

Categories: ['tech' 'business' 'sport' 'entertainment' 'politics']
```

	category	text
0	tech	tv future in the hands of viewers with home th...
1	business	worldcom boss left books alone former worldc...
2	sport	tigers wary of farrell gamble leicester say ...
3	sport	yeading face newcastle in fa cup premiership s...
4	entertainment	ocean s twelve raids box office ocean s twelve...
2220	business	cars pull down us retail figures us retail sal...
2221	politics	kilroy unveils immigration policy ex-chatshow ...
2222	entertainment	rem announce new glasgow concert us band rem h...
2223	politics	how political squabbles snowball it s become c...
2224	sport	souness delight at euro progress boss graeme s...

Рисунок 3.3 — Завантаження датасету

Наступним фрагментом можна отримати гістограму в якій буде показано розподіл текстів по групам:

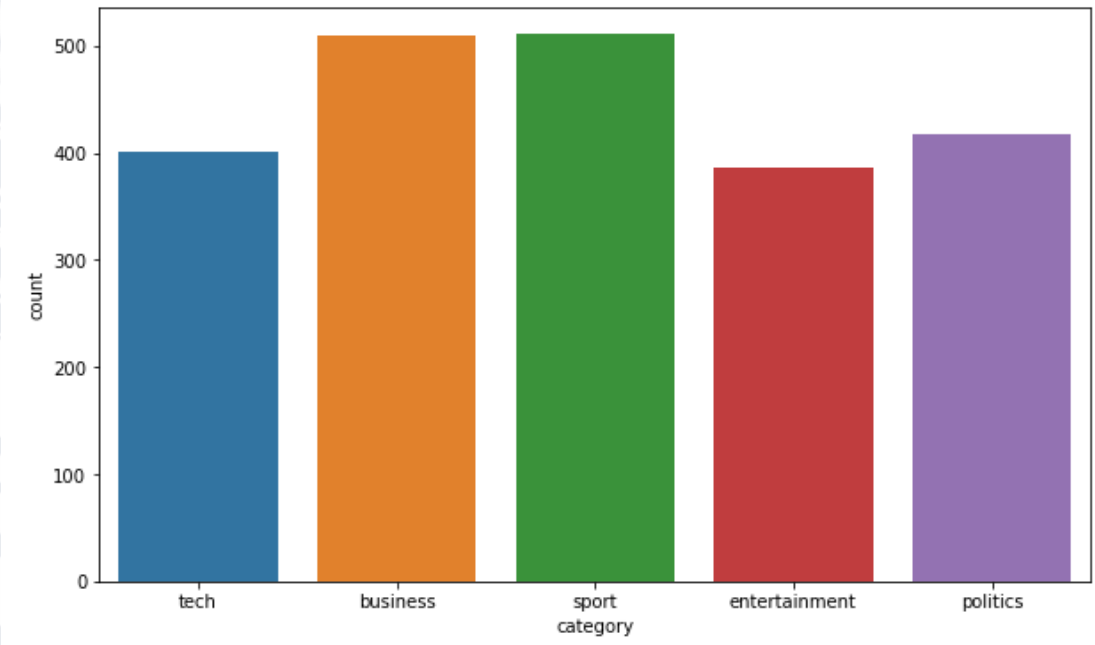


Рисунок 3.4 — Гістограма розподілу текстів по групам

Потім запускається фрагмент обробки даних і виведення результату обробки:

```
Tokenization is done.
Remove stopwords done.
Removed non english characters is done.
Lemmatization on the words.
```

Рисунок 3.5 — Результат обробки тексту

Далі потрібно створити прогностичну модель за допомогою Word2Vec у вигляді векторного простору.

Після цього потрібно запустити функцію класифікації із створеним вектором, яка розділить текст на дві групи: для навчання і для тестування, після цього буде виведено результат:

```
Vectorization is completed.
(1780, 26750) (445, 26750)
```

Рисунок 3.6 — Результат навчання

З цього результату перший вектор відповідає за кількість текстів з дат сету, що виділені для навчання, другий за кількість текстів для перевірки.

Після цього потрібно запустити навчання та тестування і через певний час отримуємо такий результат:

```
Naive Bayes :
Accuracy: 0.969         Precision: 0.969        Recall: 0.969   F1-Score: 0.969

SVC :
Accuracy: 0.975         Precision: 0.974        Recall: 0.976   F1-Score: 0.975

Decision Tree :
Accuracy: 0.769         Precision: 0.766        Recall: 0.769   F1-Score: 0.767

SGD Classifier :
Accuracy: 0.980         Precision: 0.978        Recall: 0.980   F1-Score: 0.979
```

Рисунок 3.7 — Результат навчання і тестування

Тепер потрібно проаналізувати отримані результати роботи алгоритму на основі датасету.

3.2 Порівняння результатів класифікації текстів

Після запуску алгоритму можна отримаємо такий результат:

```
Naive Bayes :
Accuracy: 0.971         Precision: 0.971        Recall: 0.966   F1-Score: 0.968

SVC :
Accuracy: 0.982         Precision: 0.981        Recall: 0.980   F1-Score: 0.980

Decision Tree :
Accuracy: 0.796         Precision: 0.804        Recall: 0.786   F1-Score: 0.789

SGD Classifier :
Accuracy: 0.987         Precision: 0.986        Recall: 0.984   F1-Score: 0.985
```

Рисунок 3.8 — Результат роботи алгоритму

У даному результаті є дані про кожен виконаний алгоритм:

Accuracy (точність) — це один із показників для оцінки класифікації моделей. Неформально точність — це доля правильних прогнозів, зроблених нашою моделлю. Формально точність має таке визначення:

$$Accuracy = \frac{Number\ of\ correct\ prediction}{Total\ number\ of\ prediction},$$

Precision — намагається відповісти на наступне питання: яка доля позитивних ідентифікацій була насправді правильною?

Точність визначається таким чином:

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

де TP - істинні позитивні результати,

FP - хибні спрацювання.

Recall — намагається відповісти на наступне запитання: яку частку фактично позитивних результатів було визначено правильно?

Математично відгук визначається наступним чином:

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

де TP - істинні позитивні результати,

FN - хибнонегативні результати.

F1-Score — поєднує точність і запам’ятовування, використовуючи їх середнє гармонічне значення. Максимізація оцінки F1 передбачає одночасну максимізацію як точності, так і запам’ятовування.

Математична формула для оцінки F1 наступна:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall},$$

На відміну від середнього арифметичного, середнє гармонічне має тенденцію бути ближчим до меншого числа в парі. Таким чином, оцінка F1 буде високою, лише якщо і точність, і запам’ятовування високі, забезпечуючи хороший баланс обох значень.

Далі потрібно побудувати графіки для всіх значень:

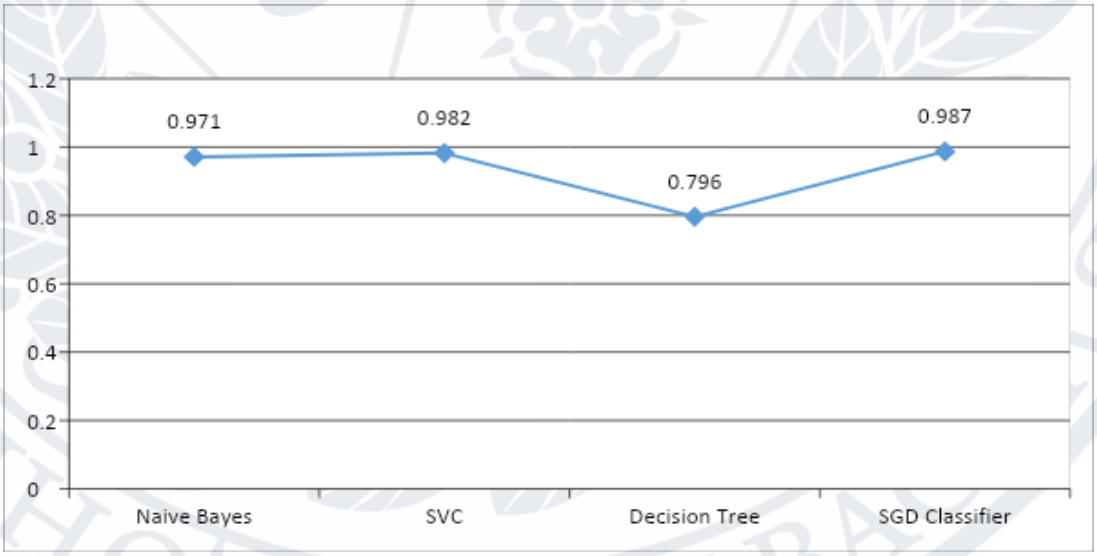


Рисунок 3.8 — Графік значення accuracy для всіх алгоритмів

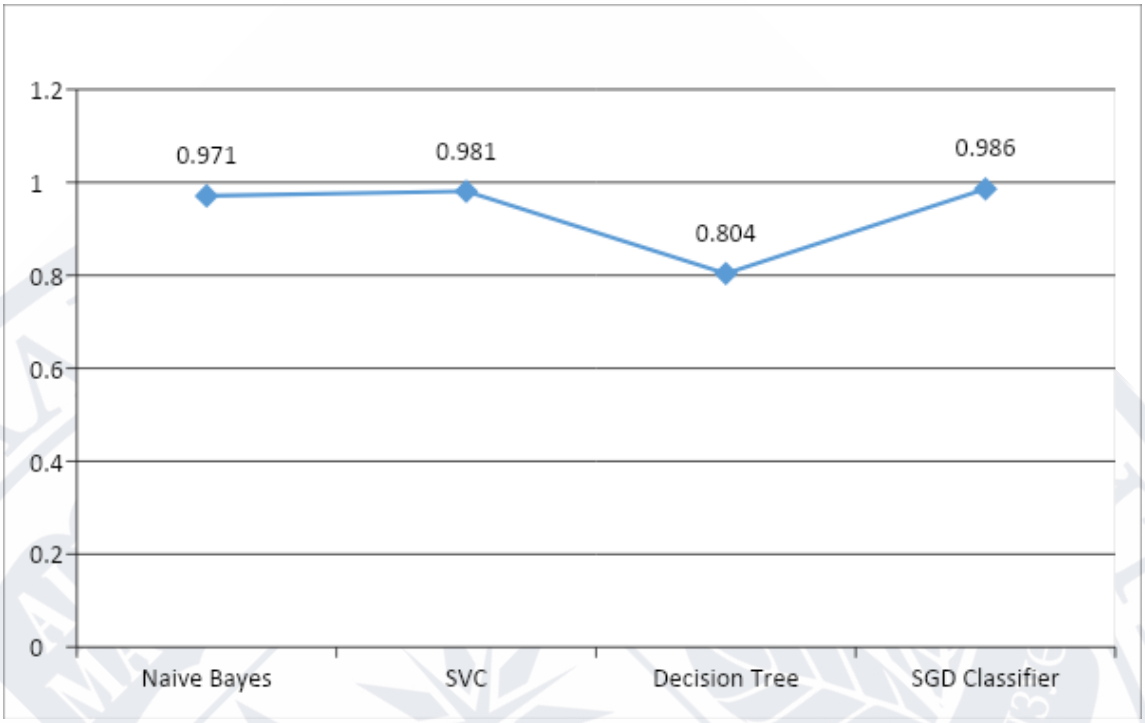


Рисунок 3.9 — Графік значення precision для всіх алгоритмів

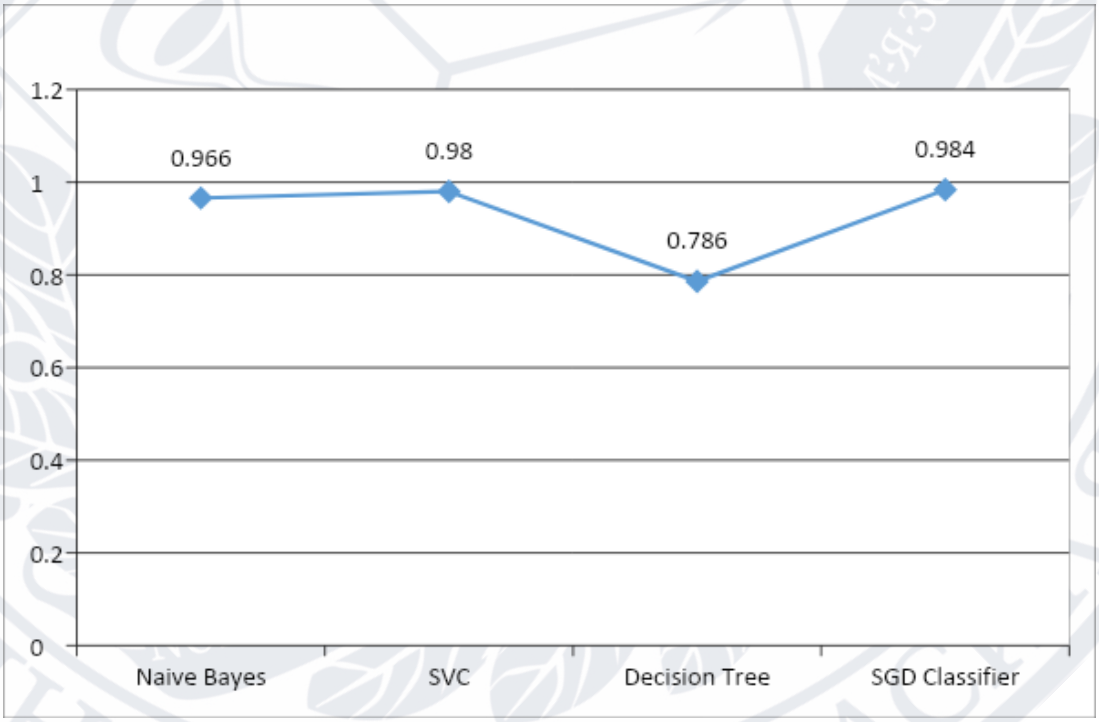


Рисунок 3.10 — Графік значення recall для всіх алгоритмів

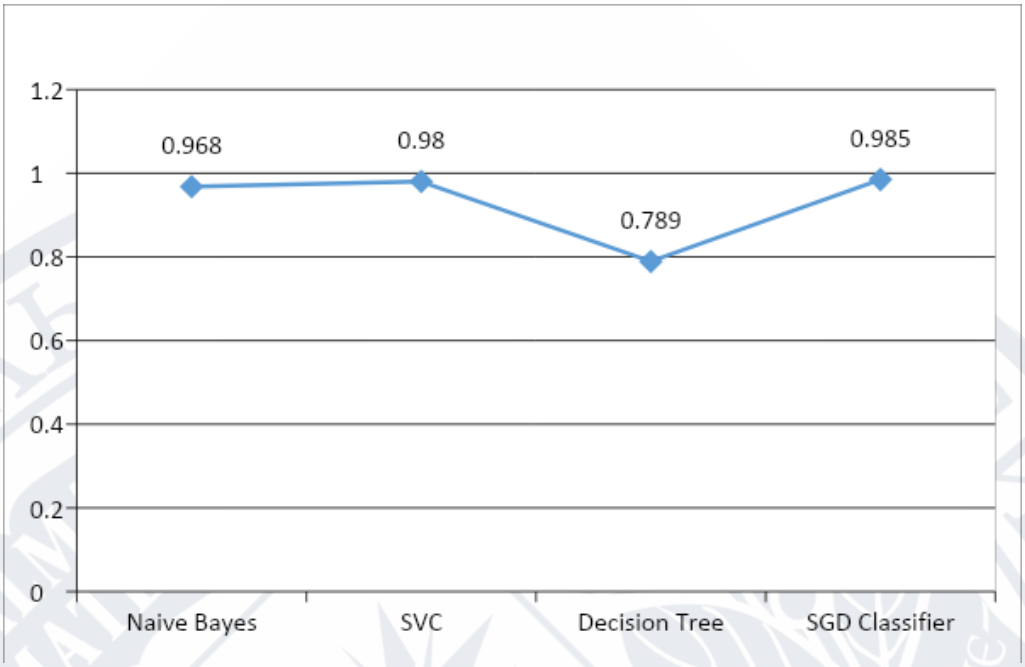


Рисунок 3.11 — Графік значення f1-score для всіх алгоритмів

Після запуску алгоритмів та отримання графіків значення потрібно зробити порівняльну таблицю для всіх значень:

Таблиця 3.1 – Порівняльна таблиця значень

	Naive Bayes	SVC	Decision Tree	SGD Classifier	Максимальне	Назва алгоритму
Accuracy	0,971	0,982	0,796	0,987	0,987	SGD Classifier
Precision	0,971	0,981	0,804	0,986	0,986	SGD Classifier
Recall	0,966	0,98	0,786	0,984	0,984	SGD Classifier
F1-Score	0,968	0,98	0,789	0,985	0,985	SGD Classifier

Отже, можемо зробити висновок, що для даного дата сету, доцільно використовувати алгоритм логістичної регресії. Таким чином користувачі

системи можуть самі завантажувати дата сети та оцінювати моделі машинного навчання для того щоб обрати найбільш ефективну модель.

Наступним кроком стане порівняння навчених моделей з ненавченими. Щоб проаналізувати результати та порівняти навчені моделі з ненавченими, зазвичай потрібно слідували таким крокам:

1. Підготовка тестового набору даних:
- Завантажити датасет BBC News Classification з Kaggle.
- Розділити датасет на тренувальну і тестову вибірки. Для ненавчених моделей використовуємо тільки тестову вибірку.
2. Ініціалізація та запуск ненавчених моделей:
- Ініціалізувати чотири моделі: Naive Bayes, SVC, Decision Tree, та SGD Classifier без тренування.
- Застосувати ненавчені моделі до тестової вибірки, щоб отримати передбачення. Оскільки моделі не навчені, вони працюють як базові класифікатори з дефолтною поведінкою (наприклад, випадкові передбачення).
3. Оцінка результатів ненавчених моделей:
- Виміряти точність, прецизійність, виклик та F1-бал для кожної з ненавчених моделей.
- Оскільки вони ненавчені, очікується вкрай низькі значення метрик.
4. Порівняння навчених та ненавчених моделей:
- Створити таблицю, де для кожного алгоритму поруч представлені метрики навченої (з наявної таблиці) та ненавченої версії.
- Проаналізувати різницю в продуктивності між навченими та ненавченими моделями.

```
(           Naive Bayes    SVC  Decision Tree  SGD Classifier
Accuracy          0.971  0.982          0.796           0.987
Precision         0.971  0.981          0.804           0.986
Recall            0.966  0.986          0.784           0.984
F1-Score          0.968  0.989          0.789           0.985,
           Naive Bayes       SVC  Decision Tree  SGD Classifier
Accuracy      0.217191  0.187890       0.188601        0.211543
Precision     0.190981  0.206712       0.177135        0.223373
Recall        0.206221  0.195413       0.203773        0.214461
F1-Score      0.212323  0.199133       0.203033        0.198064)
```

Рисунок 3.12 — Результат навчених та ненавчених моделей

На основі симуляції, можна порівняти результати навчених та ненавчених моделей. Результати можна спостерігати в таблиці 3.2.

Таблиця 3.2 – Порівняльна таблиця результату навчених та ненавчених моделей

	Naive Bayes	SVC	Decision Tree	SGD Classifier
Accuracy	0,971	0,982	0,796	0,987
Precision	0,971	0,981	0,804	0,986
Recall	0,966	0,98	0,786	0,984
F1-Score	0,968	0,98	0,789	0,985
Accuracy	0.217	0.188	0.189	0.212
Precision	0.191	0.207	0.177	0.223
Recall	0.206	0.195	0.204	0.214
F1-Score	0.212	0.199	0.203	0.198

Ці дані показують, що навчені моделі суттєво перевершують ненавчені, що очікувано, оскільки ненавчені моделі не мають здатності узагальнювати зі зразків даних. Ненавчені моделі, як правило, показують результати, які близькі до випадкового вибору, що в цьому випадку близько 20% для кожної метрики, але в симуляції дозволили невелику варіацію навколо цього рівня.

Максимальне значення серед ненавчених моделей, на основі симуляції, яку провели, становить: SGD Classifier - Прецизійність (Precision): 0.223.

Це симульоване значення, яке відображає найвищу прецизійність серед ненавчених моделей у прикладі. Такий результат близький до 22.3%, що є досить високим для ненавченої моделі.

Висновок до розділу 3

У даному розділі було описано алгоритми та оцінено ефективність їх роботи. Також було описано всі екрани та проаналізовано використані алгоритми.

ВИСНОВКИ

У світі переважна більшість інформації існує у вигляді текстів. Через це зростає необхідність у створенні сучасних, автоматизованих систем для обробки текстових даних. Для того, щоб вирішувати конкретну задачу класифікації текстів, потрібно буде витратити досить багато ресурсів, що не є прийнятим особливо, коли розглядатимуться комплексні задачі. Саме тому виникає необхідність у створенні універсального методу задля вирішення поставленої задачі.

Автоматизована класифікація тексту вважається життєво важливим методом керування та обробки величезної кількості документів у цифрових формах, які широко поширені та постійно зростають. Загалом класифікація тексту відіграє важливу роль у вилученні та узагальненні інформації, пошуку тексту та відповідях на запитання [11].

У роботі виконано такі завдання:

- досліджено літературу яка допоможе знайти відповідні алгоритми та особливості їх розробки;
- обрано інструменти та технічну платформу для розробки алгоритмів;
- розроблено кожен обраний алгоритм;
- запущено алгоритм на навчання з певними даними;
- побудовано графіки і таблиці для чотирьох значень для кожного алгоритму;
- оцінено ефективність алгоритмів машинного навчання на прикладі задачі класифікації текстів.

На прикладі дата сету, що складається із текстів статей розділених по категорія було побудовано моделі машинного навчання, показано як оцінюється їх ефективність.

На основі зроблених тестів та побудови графіків для 4 значень, що визначають ефективність класифікації тексту можна зробити висновок, що для даного дата сету найкраще використовувати метод логістичної регресії.

Використовуючи дану розроблену систему класифікації текстів, кожен може дозволити собі створити моделі машинного навчання, які будуть обрані на основі алгоритмів, оцінити ефективності моделей та обрати конкретну модель для вирішення задачі. Для створення такої системи було проаналізовано алгоритми машинного навчання та описано їх переваги та недоліки.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Announcing the new Jupyter Book [Електронний ресурс] – Режим доступу до ресурсу: https://blog.jupyter.org

2. Arthur Mello. Logistic regression: the basics. Towards Data Science, Jan 13, 2020.

3. Ashokkumar Palanivinayagam,Claude Ziad El-Bayeh, Robertas Damaševičius. A Review of Twenty Years of Machine-Learning-Based Text Classification: A Systematic Review. Algorithms 2023, 16(5), 236.

4. BBC articles fulltext and category [Електронний ресурс] – Режим доступу до ресурсу : https://www.kaggle.com

5. Books with Jupyter [Електронний ресурс] – Режим доступу до ресурсу: https://chrisholdgraf.com

6. Colab Notebooks: [Електронний ресурс] – Режим доступу до ресурсу: https://colab.research.google.com

7. Conceptual Understanding of Logistic Regression for Data Science Beginners [Електронний ресурс] – Режим доступу до ресурсуL: https://www.analyticsvidhya.com

8. Decision Tree [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org

9. Decision Tree Classification Algorithm [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com

10. Decision Tree Classification in Python Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.datacamp.com

11. Emmanouil K. Ikonomakis, Sotiris Kotsiantis, V. Tampakas. Text Classification Using Machine Learning Techniques. WSEAS TRANSACTIONS ON COMPUTERS 4(8):966-974, August 2005.

12. George Lawton, Ed Burns, Linda Rosencrance. Logistic regression. TechTarget.

13. How to tackle text classification challenges in machine learning (1/3) [Електронний ресурс] – Режим доступу до ресурсу: https://www.owt.swiss

14. Kaggle [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com

15. Machine Learning [Електронний ресурс] – Режим доступу до ресурсу: https://developers.google.com

16. Naive Bayes Classifier Explained: Applications and Practice Problems of Naive Bayes Classifier [Електронний ресурс] – Режим доступу до ресурсу: https://www.analyticsvidhya.com

17. Naive Bayes Classification Tutorial using Scikit-learn [Електронний ресурс] – Режим доступу до ресурсу: https://www.datacamp.com

18. Natural Language Processing: Naive Bayes Classification in Python [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com

19. Python Introduction [Електронний ресурс] – Режим доступу до ресурсу: https://www.w3schools.com

20. Rohith Gandhi. Naive Bayes Classifier. Towards Data Science, May 5, 2018.

21. Ruiguang Li, Ming Liu, Dawei Xu, Jiaqi Gao, Fudong Wu & Liehuang Zhu. A Review of Machine Learning Algorithms for Text Classification. Part of the Communications in Computer and Information Science book series (CCIS,volume 1506)

22. Savan Patel. Chapter 2 : SVM (Support Vector Machine) — Theory. Machine Learning 101, May 3, 2017.

23. Serafeim Loukas. Text Classification Using Naive Bayes: Theory & A Working Example. Towards Data Science, Oct 12, 2020

24. Support Vector Machine Algorithm [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com

25. Support Vector Machines [Електронний ресурс] – Режим доступу до ресурсу: https://scikit-learn.org

26. Text Classification: What it is And Why it Matters [Електронний ресурс] – Режим доступу до ресурсу: https://monkeylearn.com

27. Understanding Logistic Regression in Python Tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://www.datacamp.com

28. What is logistic regression? [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com.

29. Decision-making tree [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/%D0%94%D0%B5%D1%80%D0%B5%D0%B2%D0%BE_%D1%83%D1%85%D0%B2%D0%B0%D0%BB%D0%B5%D0%BD%D0%BD%D1%8F_%D1%80%D1%96%D1%88%D0%B5%D0%BD%D1%8C

30. Wikipedia. Free encyclopedia [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org

31. What is Kaggle? [Електронний ресурс] – Режим доступу до ресурсу: https://www.datacamp.com

32. What Is Kaggle? How to Compete in Kaggle Competitions [Електронний ресурс] – Режим доступу до ресурсу: https://builtin.com

33. Yehia Ibrahim Alzoubi, Ahmet E. Topcu, Ahmed Enis Erkaya. Machine Learning-Based Text Classification Comparison: Turkish Language Context. Appl. Sci. 2023, 13(16), 9428.

34. Сперкач М. О., Юзьвак Д. Ю. РОЗВ'ЯЗАННЯ ЗАДАЧІ КЛАСИФІКАЦІЇ ТЕКСТІВ МЕТОДАМИ ОБРОБКИ ПРИРОДНОЇ МОВИ ТА МАШИННОГО НАВЧАННЯ. Національний технічний університет України Київський політехнічний інститут імені Ігоря Сікорського. ЖУРНАЛ НАУКОВИЙ ОГЛЯД № 4(57), 2019. УДК: 004.93.

35. НЛП і текстовий аналіз [Електронний ресурс] – Режим доступу до ресурсу: https://ts2.space/uk/%D0%BD%D0%BB%D0%BF-%D1%96-%D1%82%D0%B5%D0%BA%D1%81%D1%82%D0%BE%D0%B2%D0%B8%D0%B9-%D0%B0%D0%BD%D0%B0%D0%BB%D1%96%D0%B7-

%D0%B2%D0%B8%D0%BB%D1%83%D1%87%D0%B5%D0%BD%D0%BD%D1%8F-%D1%80%D0%BE%D0%B7%D1%83/

36. Scikit-learn [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Scikit-learn

37. TensorFlow [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/TensorFlow

38. Набір інструментів природної мови [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/Natural_Language_Toolkit

39. Quinlan J. R. C4.5 Programs for machine learning. Morgan Kaufmann, San Mateo, Californie, 1993.