Національний університет «Полтавська політехніка імені Юрія Кондратюка»
(повне найменування вищого навчального закладу)
Навчально-науковий інститут інформаційних технологій та робототехніки
(повна назва факультету)
Кафедра комп'ютерних та інформаційних технологій і систем
(повна назва кафедри)

Пояснювальна записка
до дипломного проекту (роботи)
магістра
(освітньо-кваліфікаційний рівень)
на тему
Розпізнавання емоційного стану людини за допомогою
алгоритмів машинного навчання

Виконав: студент 6 курсу, групи 601-ТН
спеціальності
122 Комп'ютерні науки
(шифр і назва напряму)
Бірченко В. Л.
(прізвище та ініціали)
Керівник Капітон А. М.
(прізвище та ініціали)

Полтава – 2025 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА РОБОТОТЕХНІКИ

КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

спеціальність 122 «Комп'ютерні науки»

на тему

«Розпізнавання емоційного стану людини за допомогою алгоритмів машинного навчання»

Студента групи 601-ТН Бірченка В'ячеслава Леонідовича

Керівник роботи
доктор економічних наук,
професор Капітон А. М.

Консультант
кандидат технічних наук,
доцент Скакаліна О. В.

Завідувач кафедри
кандидат фізико-математичних наук,
доцент Двірна О.А.

Полтава – 2025 року

РЕФЕРАТ

Кваліфікаційна робота магістра: 88 с., 42 рис., 3 табл., 18 джерел, 2 додат.

Об'єкт дослідження: система автоматизованого розпізнавання емоцій у текстових даних.

Мета роботи: розробка системи, яка забезпечує точне та ефективне класифікування текстів за емоційними категоріями з можливістю інтеграції в реальні прикладні додатки.

Методи дослідження: обробка природної мови (токенізація, лематизація, видалення стоп-слів), техніки векторизації тексту (CountVectorizer, TF-IDF), алгоритми машинного навчання (Logistic Regression, Support Vector Machine, Random Forest, Naive Bayes, Decision Tree).

Структура та обсяг: магістерська кваліфікаційна робота складається зі вступу, трьох розділів та висновків.

У першому розділі розглянуто теоретичні основи розпізнавання емоційного стану людини, зокрема ключові алгоритми машинного навчання, методи обробки тексту та специфіку текстів із соціальних мереж.

Другий розділ присвячено постановці задачі, вибору методів обробки текстових даних, обґрунтуванню моделей машинного навчання.

У третьому розділі представлено реалізацію системи, включаючи навчання моделей, їх інтеграцію в інтерактивний додаток, а також аналіз результатів класифікації.

Ключові слова: розпізнавання емоцій, машинне навчання, обробка тексту, NLP, TF-IDF, SVM, SGDClassifier, Streamlit, векторизація, токенізація, лематизація, класифікація, матриця плутанини, соціальні мережі, емоційний стан.

ABSTRACT

Master's Qualification Thesis: 88 pages, 42 figures, 3 tables, 18 references, 2 appendices.

Research object: a system for automated recognition of emotions in textual data.

Objective of the work: to develop a system that ensures accurate and efficient text classification into emotional categories with the potential for integration into real-world applications.

Research methods: natural language processing (tokenization, lemmatization, stop-word removal), text vectorization techniques (CountVectorizer, TF-IDF), and machine learning algorithms (Logistic Regression, Support Vector Machine, Random Forest, Naive Bayes, Decision Tree).

Structure and volume: the master's qualification work consists of an introduction, three chapters, and conclusions.

The first chapter examines the theoretical foundations of recognizing human emotional states, focusing on key machine learning algorithms, text processing methods, and the specific features of texts from social networks.

The second chapter is dedicated to formulating the problem, selecting text data processing methods, and justifying the choice of machine learning models.

The third chapter presents the system's implementation, including model training, integration into an interactive application, and the analysis of classification results.

Keywords: emotion recognition, machine learning, text processing, NLP, TF-IDF, SVM, SGDClassifier, Streamlit, vectorization, tokenization, lemmatization, classification, confusion matrix, social networks, emotional state.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ML – Machine Learning.

NLP – Natural Language Processing.

TF – Term Frequency.

IDF – Inverse Document Frequency.

TF-IDF – Term Frequency-Inverse Document Frequency.

SVM – Support Vector Machine.

SGD – Stochastic Gradient Descent

SMOTE – Synthetic Minority Oversampling Technique.

NLTK – Natural Language Toolkit.

CBOW – Continuous Bag of Words.

GloVe – Global Vectors for Word Representation.

BERT – Bidirectional Encoder Representations from Transformers.

LSTM – Long Short-Term Memory.

API – Application Programming Interface.

CSV – Comma-Separated Values.

ВСТУП

Актуальність. Розробка інструментів для автоматизованого розпізнавання емоційного стану людини за допомогою алгоритмів машинного навчання є надзвичайно актуальним завданням у сучасному інформаційному суспільстві. Зі зростанням впливу соціальних мереж, які стали платформою для вираження емоцій, обміном думками та інформацією, обсяги текстових даних, що генеруються щодня, значно зросли. Ці дані мають значну цінність для аналізу настроїв суспільства, оцінки реакцій на події та розробки маркетингових стратегій. Водночас велика кількість інформації ускладнює її обробку традиційними методами, що робить автоматизацію процесів аналізу емоцій необхідною для забезпечення швидкості та точності отримання інсайтів.

З розвитком технологій машинного навчання відкриваються нові можливості для створення моделей, здатних аналізувати текстові дані з урахуванням контексту, стилістичних особливостей і взаємозв’язків між словами. Алгоритми, такі як стохастичний градієнтний спуск, ансамблеві методи та трансформери, довели свою ефективність у задачах обробки природної мови (NLP). Вони дозволяють досягти високої точності при класифікації емоцій у текстах, що є ключовим для аналізу динамічного та багатогранного емоційного контенту в соціальних мережах.

Актуальність цієї роботи зумовлена також специфікою текстів у соціальних мережах, які є неструктурованими та часто включають сленг, скорочення, емодзі та інші невербальні елементи. Це створює додаткові виклики для автоматизованого аналізу, вимагаючи розробки підходів, які б ефективно адаптувалися до цих характеристик. Системи автоматизованого розпізнавання емоцій повинні враховувати ці особливості для забезпечення точності прогнозів, що має важливе значення для таких галузей, як маркетинг, психологія та соціальні дослідження.

Мета роботи. Метою роботи є розробка теоретичної та практичної основи для автоматизованого розпізнавання емоційного стану людини за текстовими даними із застосуванням алгоритмів машинного навчання. Особлива увага приділяється аналізу текстів із соціальних мереж, які характеризуються короткими повідомленнями, використанням сленгу та інших стилістичних маркерів. У рамках роботи розроблено модель на основі алгоритму стохастичного градієнтного спуску (SGDClassifier), яка показала високу ефективність у класифікації емоцій.

Обґрунтування проектних рішень. Для досягнення поставленої мети у роботі використовуються сучасні алгоритми машинного навчання, які поєднуються з методами обробки природної мови, такими як векторизація тексту та TF-IDF трансформація. Розроблено підхід до підготовки текстових даних, що включає видалення пунктуації, токенізацію, лематизацію та видалення стоп-слів, що дозволило покращити якість текстових даних для моделювання. Для інтерактивного застосування отриманої моделі створено додаток, який дозволяє виконувати аналіз емоцій у тексті та візуалізувати результати за допомогою радарних і стовпчикових діаграм.

Галузі застосування. Результати роботи можуть бути використані в багатьох прикладних сферах. У бізнесі такі системи можуть допомагати аналізувати відгуки клієнтів, визначати настрої ринку та формувати маркетингові стратегії. У соціальних дослідженнях автоматизоване розпізнавання емоцій дозволяє оцінювати громадську думку та вивчати реакцію суспільства на події.

Таким чином, робота спрямована на підвищення ефективності автоматизованого розпізнавання емоцій за текстовими даними, що має значний потенціал для впровадження в різних галузях. У першому розділі досліджено теоретичні аспекти розпізнавання емоцій, розглянуто ключові алгоритми машинного навчання та особливості текстів у соціальних мережах. У наступних розділах представлено практичну реалізацію моделі, аналіз результатів та створення інтерактивного додатку для розпізнавання емоційного стану.

РОЗДІЛ 1
ОСНОВИ РОЗПІЗНАВАННЯ ЕМОЦІЙНОГО СТАНУ ЛЮДИНИ

1.1 Поняття емоційного стану та його вираження у тексті

Емоційний стан людини є важливим аспектом її поведінки та взаємодії з навколишнім середовищем. Він відображає реакцію на різноманітні стимули і може мати значний вплив на ухвалення рішень, спілкування та загальне самопочуття. У контексті текстових даних, таких як повідомлення в соціальних мережах, емоційний стан можна визначати за допомогою аналізу мови, виразів, лексичних маркерів та структури тексту.

Емоційний стан є інтегральною частиною людської психіки, яка відображає внутрішній психологічний та фізіологічний стан людини в певний момент часу. Емоції можуть бути позитивними, негативними або нейтральними, і вони безпосередньо впливають на те, як людина взаємодіє з оточенням. У контексті текстових даних, таких як повідомлення в соціальних мережах, емоційний стан можна виявляти через аналіз лексики, стилістики та контексту тексту, що дозволяє з високою точністю визначати емоційне забарвлення повідомлень [1].

У багатьох дослідженнях зазначається, що емоції відіграють ключову роль у цифрових комунікаціях, особливо в соціальних мережах, де повідомлення часто короткі, але емоційно насичені. Користувачі виражають свої почуття через слова, емодзі, та інші невербальні засоби. Наприклад, публікації можуть бути емоційно насиченими завдяки таким елементам, як капіталізація, або використання спеціальних символів чи знаків оклику, що підсилюють емоційний заряд тексту [2].

У текстових даних емоційний стан людини може бути класифікований через певні емоційні категорії: радість, смуток, страх, здивування тощо. Аналіз таких емоційних станів може бути особливо корисним у сферах маркетингу,

соціальної аналітики, а також для психологічних досліджень, оскільки дозволяє зрозуміти, як люди реагують на певні події або теми [1].

Емоційний стан у текстах відображається через використання емоційно забарвлених слів, які несуть у собі позитивне, негативне або нейтральне значення. Дослідники часто опираються на емоційні словники або лексикони, що містять слова з чітко визначеною емоційною полярністю. Наприклад, слова «радість», «успіх», «любов» мають позитивну конотацію, тоді як слова «страх», «невдача», «ненависть» вважаються негативними. Нейтральні слова, як правило, не несуть яскраво вираженого емоційного забарвлення [1, 2].

Додатково, контекст грає важливу роль у визначенні емоційного стану тексту. Одне й те саме слово може мати різні емоційні конотації залежно від того, як воно використовується. Наприклад, слово «сильно» може виражати як позитивні емоції (сильно люблю), так і негативні (сильно ненавиджу) (рис. 1.1). Тому аналіз текстів потребує обліку контекстуальних зв'язків між словами, а також синтаксичних відношень між елементами тексту, що дає можливість точніше розпізнавати емоційний стан людини [1].

Рисунок 1.1 – Контекстуальні зв'язки між словами

У соціальних мережах текстові повідомлення часто короткі, що додає викликів для автоматизованого визначення емоцій. Користувачі нерідко вживають скорочення, сленг, емодзі та інші невербальні елементи для вираження емоцій. Це ускладнює класичний текстовий аналіз і вимагає застосування гібридних моделей, що можуть інтегрувати текстовий і символічний контент. Наприклад, використання емодзі часто посилює емоційний контекст тексту, надаючи додаткову інформацію про емоційний стан автора [1, 2].

Окрім цього, інтонаційні та стилістичні маркери, як-от використання великих літер, повторюваних символів або знаків оклику, теж додають емоційного навантаження тексту. Такі стилістичні засоби можуть виражати захоплення, тривогу або навіть агресію, які важко розпізнати виключно через аналіз слів [2].

Таким чином, емоційний стан у тексті є багатогранним поняттям, яке може бути виражене через лексичні, синтаксичні та стилістичні засоби. Для точного аналізу необхідно враховувати не лише окремі слова, але й їхній контекст, семантичні та синтаксичні зв'язки, що дозволяє краще зрозуміти емоційний вплив тексту на його аудиторію. Автоматизовані алгоритми, що використовуються для аналізу емоційного стану, повинні бути здатні інтегрувати ці фактори для досягнення високої точності визначення емоцій [1].

1.2 Алгоритми машинного навчання для розпізнавання емоцій

Машинне навчання (ML) стало потужним інструментом для аналізу текстових даних і розпізнавання емоцій у текстах. Воно дозволяє автоматизувати процес обробки великих обсягів даних та підвищити точність класифікації емоційних станів. Основна ідея машинного навчання полягає у створенні моделей, які можуть виявляти закономірності у вхідних даних і робити прогнози на основі попереднього навчання.

Алгоритми машинного навчання працюють за двома основними підходами: контрольоване навчання (supervised learning) та неконтрольоване

навчання (unsupervised learning). У випадку розпізнавання емоцій у текстах, зазвичай використовується контрольоване навчання, коли модель навчається на попередньо анотованих текстах, що мають позначені емоційні категорії. Неконтрольоване навчання може використовуватися для кластеризації текстів без попередньої анотації, коли модель сама знаходить схожі патерни у даних [3].

Обробка текстових даних для машинного навчання складається з кількох важливих етапів: токенізація, лематизація та стемінг, векторизація, використання вбудованих векторів слів.

Токенізація – базовий етап обробки текстових даних, під час якого текст розбивається на окремі одиниці, відомі як токени. Токенами можуть бути слова, фрази або навіть символи, залежно від завдань.

Основна мета токенізації полягає в тому, щоб підготувати текст для подальшої обробки та аналізу. Наприклад, у реченні «Машинне навчання – це інструмент для аналізу даних», токенізація виділить такі токени, як «Машинне», «навчання», «це», «інструмент», «для», «аналізу», «даних» (рис. 1.2).

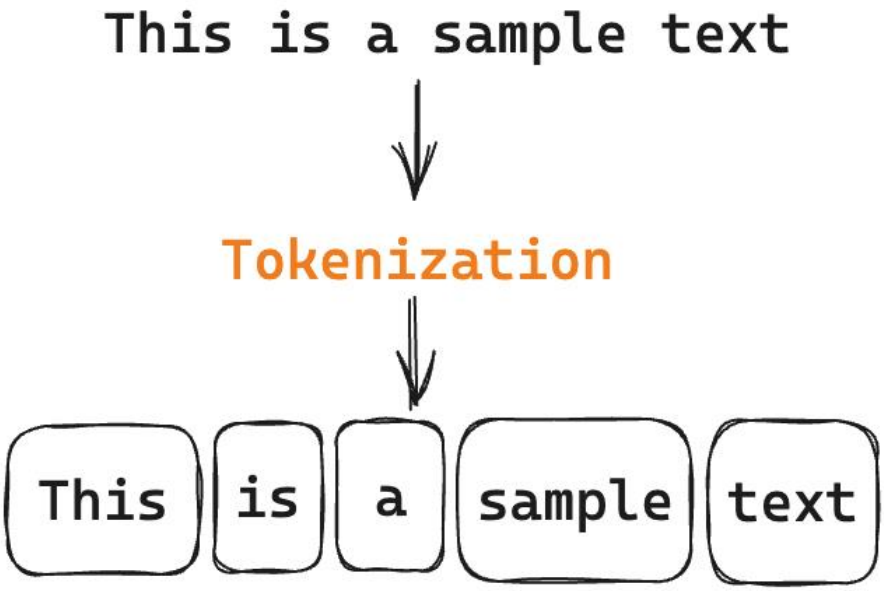

Рисунок 1.2 – Токенізація

Існують різні підходи до токенізації: розбиття за пробілами, вилучення розділових знаків або облік складних словоформ, як-от складні іменники чи фрази [4].

Один із базових підходів – розбиття за пробілами, при якому текст розділяється на слова шляхом визначення кожного пробілу як межі між токенами. Цей метод є ефективним для мов з чітким розподілом слів, але має обмеження для мов, де пробіли не використовуються (наприклад, китайська) [4].

Інший підхід – вилучення розділових знаків, коли з тексту видаляються розділові знаки, щоб уникнути хибного інтерпретування пунктуації як окремих токенів. Це дозволяє зосередитися лише на словах або значущих символах тексту, зменшуючи шум і підвищуючи точність подальшої обробки [4].

Для складніших випадків використовується токенізація складних словоформ, яка враховує багатокомпонентні конструкції, як-от складені іменники (наприклад, «сонячна панель», «штучний інтелект») або фрази. У таких випадках токенізація передбачає об’єднання окремих слів в один токен для збереження повного сенсу виразу [4].

Інші підходи включають токенізацію за морфемами, що враховує частини слова, як-от префікси та суфікси, та підходи з використанням лексичних баз, де відомі словоформи зіставляються з лексичними ресурсами для коректної токенізації. Усі ці методи допомагають адаптувати токенізацію до різних типів текстів і цілей аналізу, забезпечуючи точніше представлення інформації для подальшої обробки та аналізу [4].

Лематизація та стемінг – два ключові процеси нормалізації тексту, що спрямовані на приведення слів до їх базової форми (рис. 1.3).

Лематизація перетворює слово до його початкової граматичної форми (леми), враховуючи морфологічні особливості мови. Наприклад, слова «бігав», «бігти», «бігає» будуть приведені до леми «біг». Це забезпечує точнішу обробку, оскільки враховується контекст слова та його граматичні характеристики [5].

Стемінг, на відміну від лематизації, є більш грубим методом, який полягає у відсіченні суфіксів та префіксів для отримання кореня слова. Наприклад, від слів «швидко», «швидкий», «швидкі» буде виділено «швидк», але стемінг не враховує граматичний контекст, що може призвести до менш точних результатів.

Стемінг зазвичай швидший за лематизацію, але остання забезпечує більшу точність у складних текстах [5].

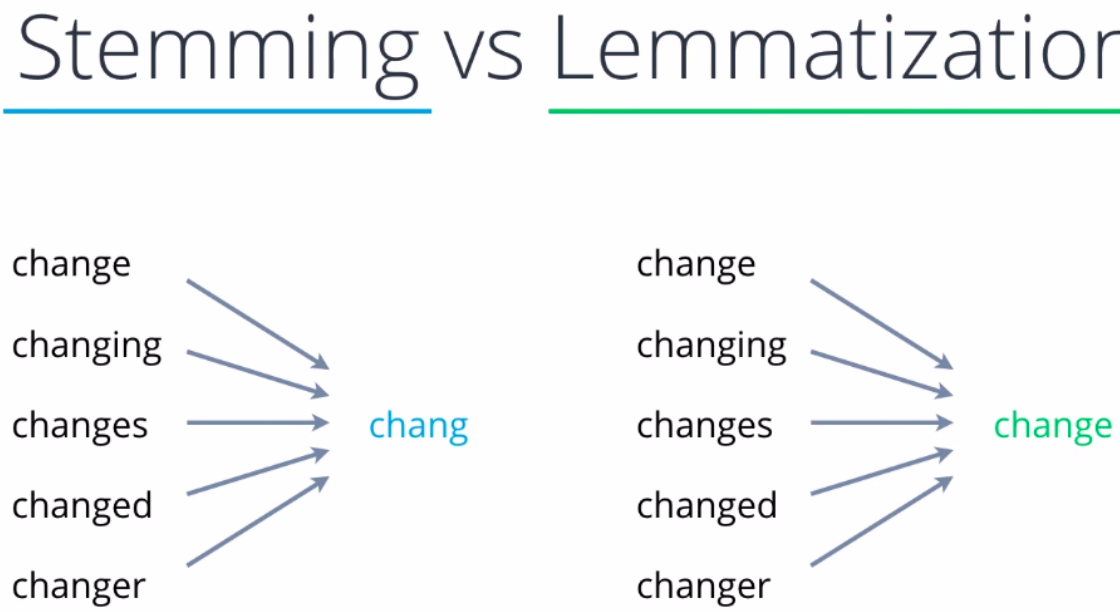

Рисунок 1.3 – Техніки нормалізації тексту

Векторизація – процес перетворення текстових даних у числовий формат, придатний для обробки алгоритмами машинного навчання. У текстах слова не мають прямого числового представлення, тому необхідно перетворити їх на вектори (числові матриці) (рис. 1.4). Один із популярних методів векторизації – TF-IDF (Term Frequency-Inverse Document Frequency), який оцінює важливість слова в окремому документі відносно всього корпусу текстів [6].

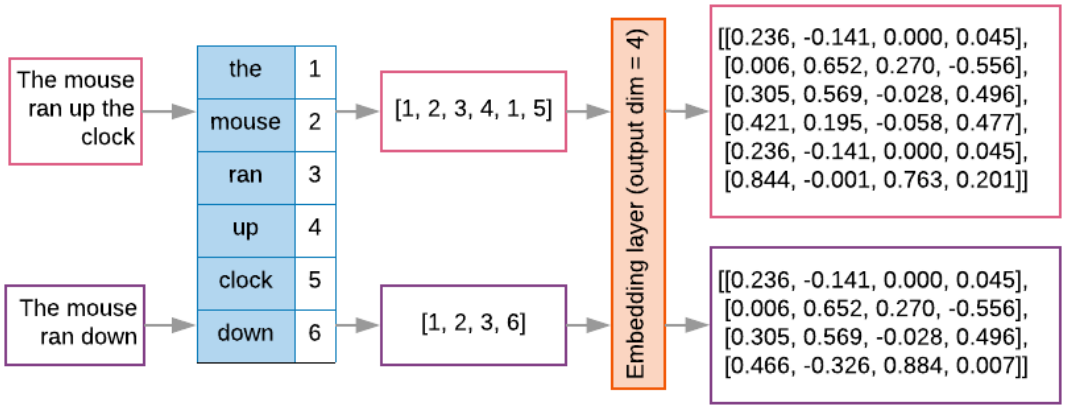

Рисунок 1.4 – Процес векторизації тексту

TF-IDF складається з двох частин: частота терміну (TF) – це кількість появ слова в документі, та зворотна частота документа (IDF) – це оцінка рідкісності слова в документі. Слова, що часто трапляються в одному документі, але рідко з'являються в інших, отримують вищу вагу, що робить TF-IDF особливо корисним для виділення ключових термінів у тексті (рис. 1.5) [7].

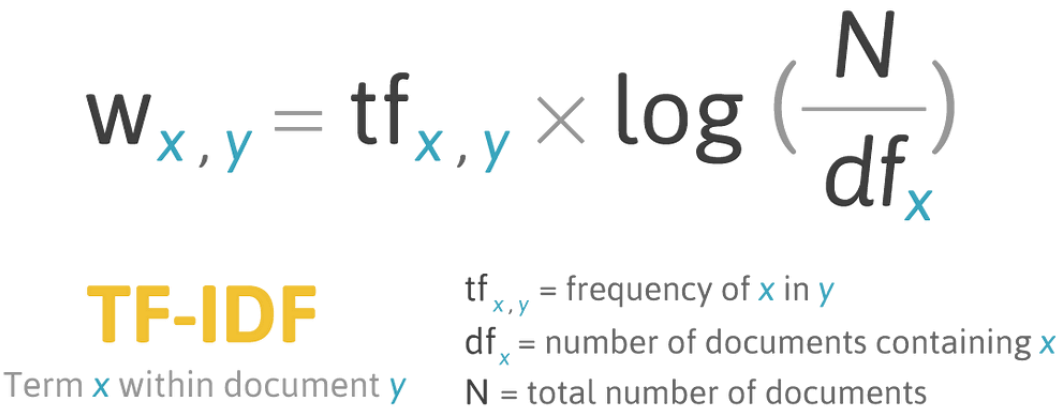

Рисунок 1.5 – Метод векторизації TF-IDF

Використання вбудованих векторів слів (Word Embeddings) – це складний метод векторизації, що дозволяє передавати не просто наявність слова у тексті, а його семантичне значення та зв'язки з іншими словами в різних контекстах. Векторні уявлення слів, або ембеддінги (embeddings), представляють кожне слово як точку в багатовимірному просторі, де семантично близькі слова

розташовані ближче один до одного. Популярними моделями для побудови таких векторів є Word2Vec, GloVe та BERT [6].

Word2Vec – це одна з перших моделей для векторизації слів, розроблена компанією Google. Word2Vec використовує дві основні архітектури: CBOW (Continuous Bag of Words) та Skip-gram. CBOW прогнозує слово за його контекстом, тоді як Skip-gram передбачає контекст для даного слова (рис. 1.6) [8].

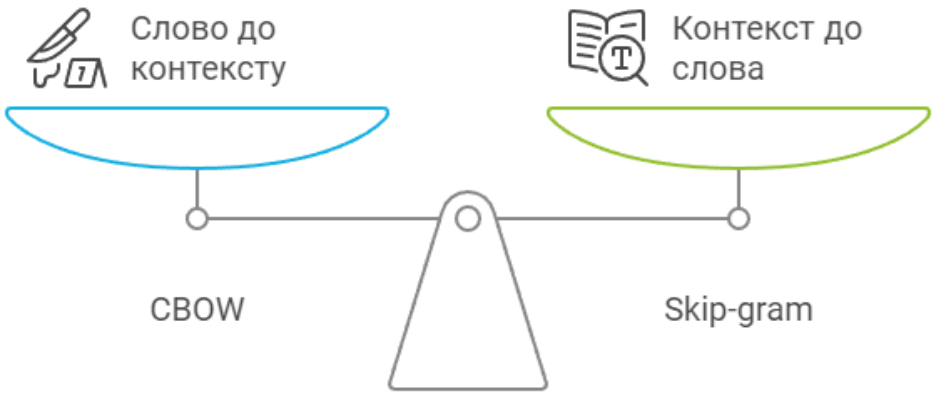

Рисунок 1.6 – Суть роботи CBOW та Skip-gram у Word2Vec

У результаті навчання на великому обсязі тексту Word2Vec вивчає, які слова часто трапляються поряд, і розміщує їх у багатовимірному векторному просторі так, що схожі слова (наприклад, «король» і «королева») будуть розташовані близько одне до одного. Однією з унікальних особливостей Word2Vec є можливість виконувати лінійні операції з векторами, що дозволяє знаходити семантичні зв'язки між словами (наприклад, «король - чоловік + жінка = королева»).

GloVe (Global Vectors for Word Representation) – модель, розроблена командою Стенфордського університету, яка враховує як локальні, так і глобальні характеристики тексту.

GloVe будується на матриці співвідношень слів і контекстів, де кожен елемент матриці відображає, як часто пара слів зустрічається разом у тексті.

Основна ідея полягає в тому, що глобальні статистики співвідношення слів є важливими для побудови точних векторних уявлень (рис. 1.7) [9].

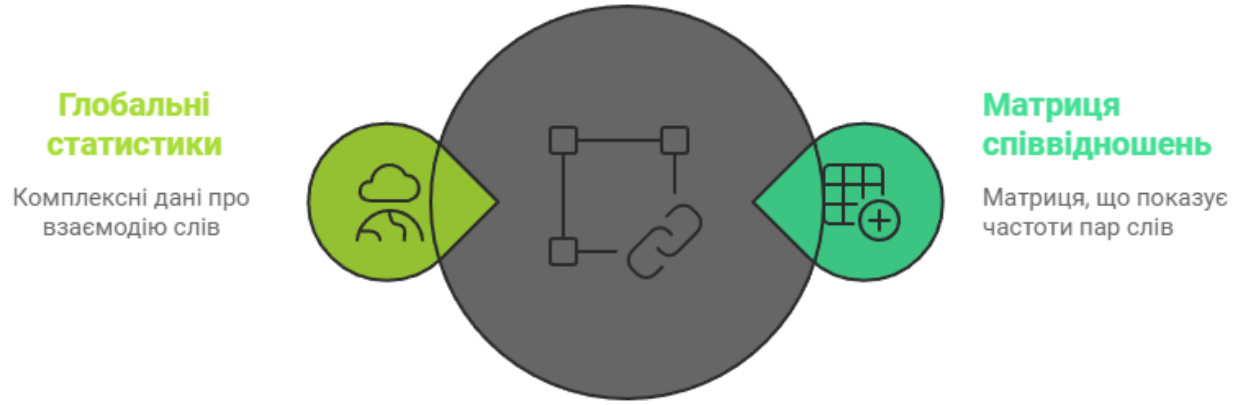

Рисунок 1.7 – Суть роботи GloVe

GloVe надає глибшу інформацію про взаємодію слів у всьому корпусі тексту, ніж Word2Vec, і підходить для задач, де важливо враховувати як локальний, так і глобальний контекст [9].

BERT (Bidirectional Encoder Representations from Transformers) – більш сучасна модель, заснована на архітектурі трансформерів. На відміну від Word2Vec і GloVe, BERT враховує двосторонній контекст, тобто розуміє значення слова з урахуванням як попередніх, так і наступних слів у реченні. Це дозволяє BERT краще обробляти складні тексти, де значення слова сильно залежить від його оточення [8, 10].

BERT навчається шляхом маскування деяких слів у реченнях і спроби моделі передбачити їх на основі контексту. Завдяки цьому підходу BERT демонструє високі результати у завданнях розпізнавання емоцій та аналізу настроїв у текстах (рис. 1.8) [1].

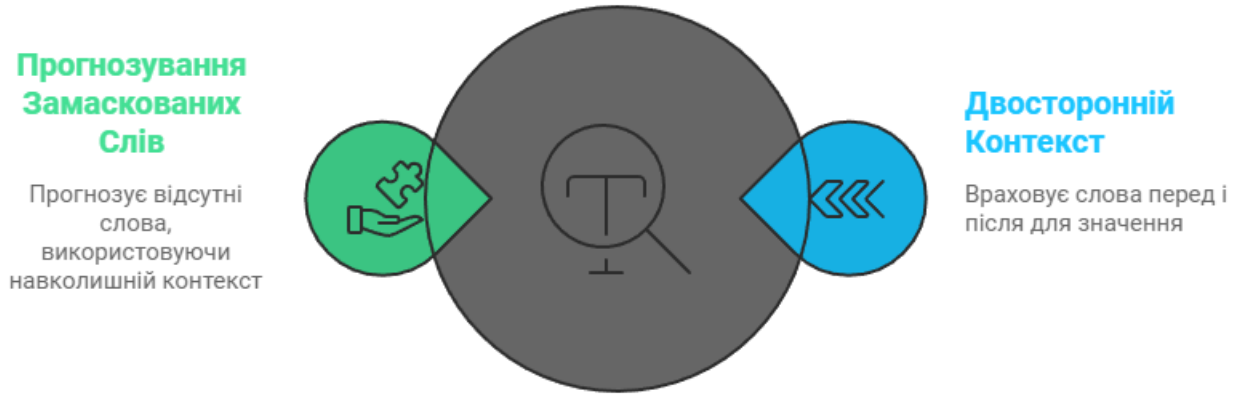

Рисунок 1.8 – Суть роботи BERT

На відміну від класичних методів векторизації, таких як TF-IDF, які розглядають слова незалежно одне від одного, вбудовані вектори слів враховують контекст і взаємодію між словами. Це дає змогу більш точно передавати значення слова в різних ситуаціях.

Наприклад, слово «коса» може означати сільськогосподарське знаряддя для косіння трави, смугу суходолу в морі або заплетене волосся залежно від контексту, і такі моделі, як BERT, дозволяють правильно інтерпретувати його значення, виходячи з оточуючих слів. Використання таких моделей забезпечує високоточний аналіз тексту та дозволяє ефективніше розпізнавати емоційний стан на основі текстових даних [1].

1.3 Підходи до автоматизованого визначення емоційного забарвлення

Для автоматизованого розпізнавання емоцій у текстах використовуються різноманітні алгоритми машинного навчання (рис. 1.9).

Наївний байєсівський класифікатор є одним із найбільш поширених і простих у використанні алгоритмів для класифікації тексту, зокрема для задач аналізу емоцій. Він базується на теоремі Байєса та передбачає, що всі ознаки (слова) в тексті є незалежними одна від одної. Це припущення є «наївним», оскільки в реальних текстах слова часто залежать одне від одного в контексті.

Незважаючи на це, наївний байєс показує високу ефективність для класифікації текстів, особливо коли доступний великий корпус даних для навчання [11].

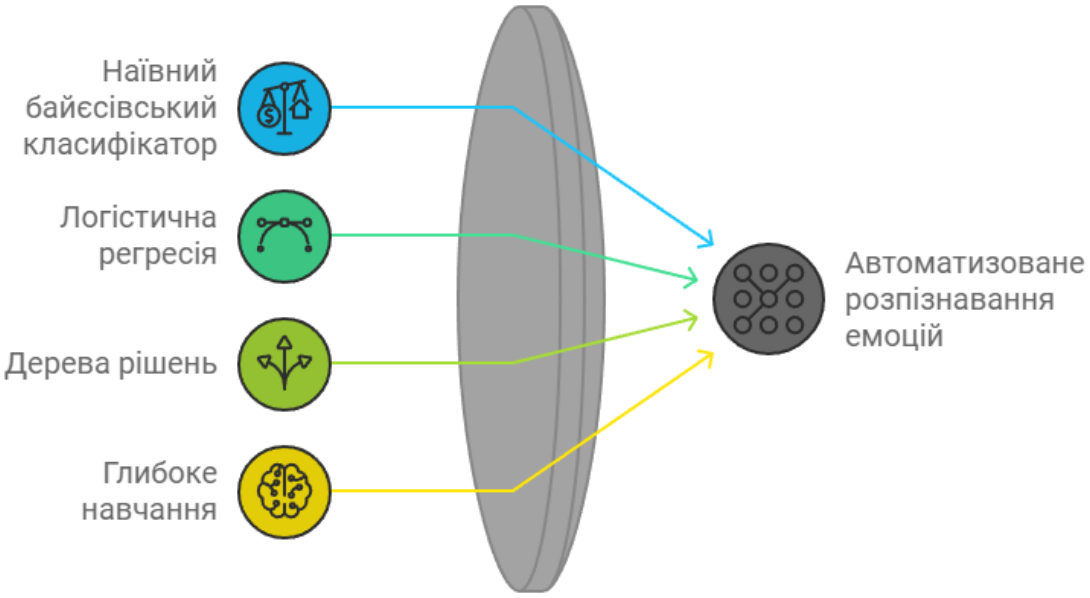

Рисунок 1.9 – Алгоритми розпізнавання емоцій

Основна перевага наївного байєсівського класифікатора – його швидкість і здатність працювати з великими обсягами даних. Він обчислює ймовірність того, що певний текст належить до конкретної емоційної категорії (наприклад, позитивної, негативної або нейтральної) на основі присутності або відсутності окремих слів. Алгоритм використовує ймовірнісну модель, яка бере до уваги частоту появи слів у різних класах емоцій і дозволяє легко додавати нові категорії для класифікації [11].

Крім того, алгоритм використовує розрахунок умовних ймовірностей – обчислює ймовірності для кожного слова в тексті, зважаючи на їх частоту в документах певного класу, і потім визначає, до якого класу найімовірніше належить текст загалом (рис. 1.10). Це дозволяє алгоритму швидко обробляти нові документи, навіть при обмеженій кількості даних.

Незважаючи на свою простоту, наївний байєс демонструє відмінні результати у задачах класифікації текстів завдяки тому, що навіть просте врахування частоти слів може бути достатнім для правильного визначення

емоційного забарвлення. Алгоритм особливо ефективний у випадках, коли тексти мають чітко визначену полярність (позитивну чи негативну) або якщо моделі навчено на добре структурованих даних [1, 2].

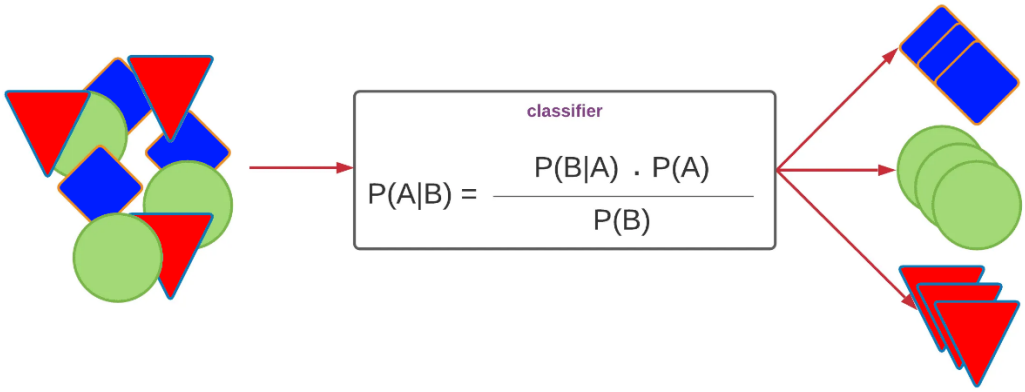

Рисунок 1.10 – Наївний байєсівський класифікатор

Логістична регресія є лінійним класифікатором, який використовується для вирішення задач бінарної або багатокласової класифікації, включаючи класифікацію емоцій у текстах. На відміну від наївного байєсівського алгоритму, логістична регресія базується на пошуку оптимальної гіперплощини, що розмежовує класи текстів на основі певних характеристик, таких як частота слів або їх значення (рис. 1.11) [12].

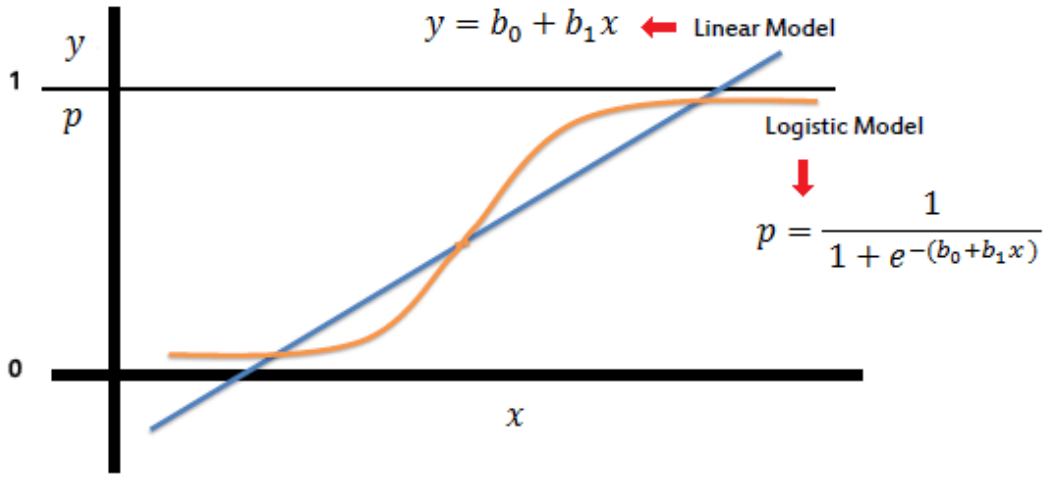

Рисунок 1.11 – Логістична регресія

У контексті аналізу емоцій логістична регресія добре працює для задач, де необхідно чітко розділяти класи емоцій, наприклад, позитивні, негативні та нейтральні тексти. Вона використовує сигмоїдну функцію для передбачення ймовірності належності тексту до одного з класів, що робить її ефективною для класифікації текстів із двома можливими результатами.

Однією з основних переваг логістичної регресії є її інтерпретованість, оскільки вона надає чітке уявлення про вплив кожного слова або фічі на класифікаційне рішення. Це важливо для розуміння того, які слова або словосполучення в тексті найбільше впливають на визначення емоційного стану. Крім того, логістична регресія добре масштабується для обробки великих наборів даних [2].

Дерева рішень – популярні алгоритми для класифікації, що створюють дерево, де кожен вузол відповідає певній ознаці (наприклад, наявності конкретного слова), а листки дерева – класи (емоційні категорії). Дерева рішень добре працюють для задач класифікації, коли важливо визначити не лише, чи присутні в тексті певні слова, але й у якому порядку або контексті вони зустрічаються (рис. 1.12) [13].

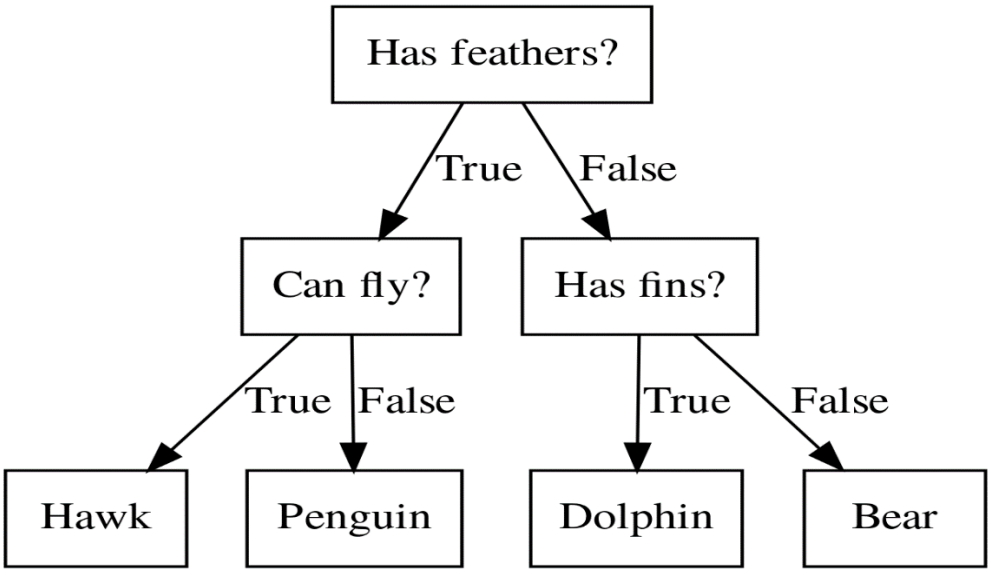

Рисунок 1.12 – Дерево рішень

Однак, для підвищення точності прогнозів використовуються ансамблеві методи, такі як Random Forest та Gradient Boosting. Random Forest – ансамблевий метод, що створює кілька дерев рішень і комбінує їх прогнози для отримання більш надійного результату. Він підходить для багатошарових емоційних класифікацій, де враховуються різні рівні емоційності тексту (рис. 1.13) [14].

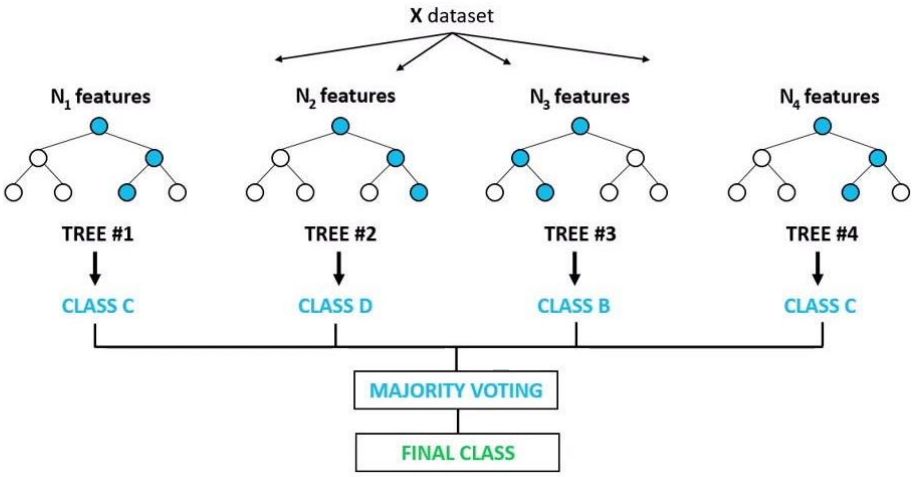

Рисунок 1.13 – Random Forest

Gradient Boosting використовує послідовність дерев рішень, кожне з яких намагається виправити помилки попередніх дерев. Це дозволяє досягти високої точності, особливо для складних текстів, де емоційне забарвлення може бути змішаним або важким для розпізнавання (рис. 1.14) [14].

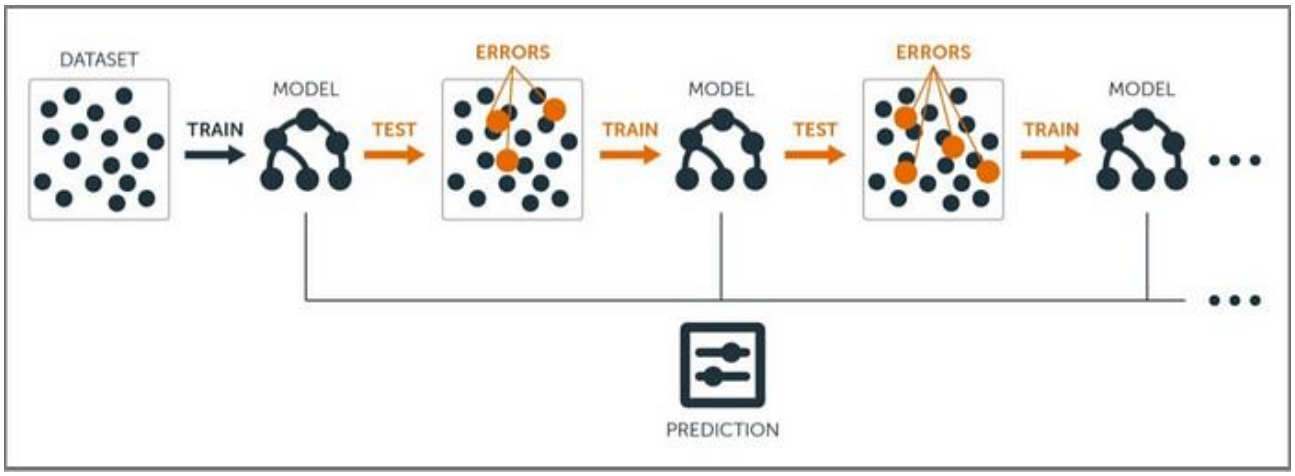

Рисунок 1.13 – Gradient Boosting

Ансамблеві методи є потужними інструментами для аналізу емоційних станів, оскільки вони можуть враховувати широкий спектр ознак тексту і знижують ризик перенавчання (overfitting). Вони добре працюють у задачах, де класифікація емоцій вимагає врахування складних патернів і взаємозв'язків між словами [1, 2].

Глибоке навчання, особливо у вигляді нейронних мереж, стало основою сучасного аналізу тексту, включаючи класифікацію емоцій. LSTM (Long Short-Term Memory) та BERT (Bidirectional Encoder Representations from Transformers) є двома найпопулярнішими моделями для задач обробки тексту.

LSTM – це тип рекурентної нейронної мережі (RNN), яка здатна зберігати контекст та враховувати довгострокові залежності у тексті. Це особливо важливо для аналізу емоцій, коли значення слова залежить від попередніх або наступних слів. Наприклад, у реченні «Не зовсім задоволений результатом», LSTM може правильно зрозуміти, що слово «задоволений» у цьому контексті означає негативну емоцію через слово «не зовсім» [15].

BERT є ще більш просунутим методом, який базується на трансформерах і враховує контекст в обох напрямках – і вперед, і назад. Це дозволяє BERT правильно інтерпретувати значення слова у складних і неоднозначних текстах, де контекст може змінювати емоційне забарвлення. BERT є ефективним для задач аналізу емоцій у великих текстах і демонструє високу точність завдяки можливості враховувати контекст не тільки в межах одного речення, але й на рівні всього тексту [10].

Нейронні мережі мають високу здатність до генералізації та можуть розпізнавати складні патерни емоційних взаємозв'язків між словами, що робить їх одними з найпотужніших інструментів для аналізу емоцій у текстах [1, 2].

Методи обробки природної мови (Natural Language Processing, NLP) є важливими для ефективного використання машинного навчання у розпізнаванні емоцій у текстах. Вони включають синтаксичний та семантичний аналіз текстів, що дозволяє виділяти не лише емоційно забарвлені слова, але й зв'язки між ними.

Одним з важливих аспектів є врахування контекстуальних зв'язків між словами, що дозволяє краще розпізнавати емоційний зміст тексту. Наприклад, моделі типу BERT навчаються на двосторонньому контексті, що допомагає зрозуміти значення слова не лише у прямому, але й у зворотному контексті. Це дозволяє моделі точніше визначати емоції в тексті, особливо у випадках, коли одні й ті самі слова можуть мати різні емоційні конотації залежно від контексту [1].

Використання машинного навчання для розпізнавання емоцій у текстах має численні переваги, такі як автоматизація обробки великих масивів даних, підвищення точності аналізу та можливість налаштування моделей під конкретні завдання.

Таким чином, машинне навчання є потужним інструментом для аналізу емоційного стану на основі текстових даних. Правильний вибір методів і алгоритмів дозволяє досягти високої точності в класифікації емоцій і отримати цінну інформацію про емоційний вплив текстів на аудиторію [1, 2].

1.4 Особливості текстових даних у соціальних мережах

Текстові дані у соціальних мережах відрізняються від інших типів текстів своєю неформальністю, динамічністю та різноманіттям. Вони формуються під впливом культурних, соціальних і технологічних чинників, що робить їх особливими у контексті автоматизованого аналізу емоційного стану. Розуміння цих особливостей є ключовим для розробки ефективних моделей машинного навчання, що здатні коректно інтерпретувати емоційний зміст повідомлень.

У соціальних мережах користувачі часто висловлюються короткими, фрагментованими повідомленнями, які мають лаконічний характер. Це може бути обмежено кількістю символів або простим бажанням користувачів передати думки швидко і доступно. Така лаконічність створює виклики для розпізнавання емоцій, оскільки навіть окремі слова або символи можуть нести глибоке

емоційне навантаження. Наприклад, повідомлення «Знову ці понеділки...» може виражати негативне ставлення, навіть якщо текст сам по собі нейтральний.

Користувачі соціальних мереж активно використовують сленг, скорочення та інші нестандартні форми вираження, що ускладнює автоматизований аналіз тексту. Такі елементи, як «лол» (сміх), «омг» (подив) можуть суттєво змінити емоційний зміст повідомлення. Крім того, емодзі та інші графічні символи стали невід'ємною частиною онлайн-спілкування. Вони передають емоційний контекст або уточнюють зміст повідомлення.

У соціальних мережах повідомлення часто написані без контексту або у формі, де значення залежить від попередніх дописів або обговорення. Одна й та ж фраза може бути сприйнята по-різному залежно від контексту, у якому вона вживається. Наприклад, коментар «Нарешті!» може виражати як радість, так і полегшення, або навіть сарказм, залежно від контексту. Це створює додаткові труднощі для автоматизованого визначення емоційного забарвлення, оскільки модель повинна брати до уваги не лише окремі слова, але й їхнє місце у загальному обговоренні або діалозі.

У текстах соціальних мереж важливу роль відіграють стилістичні засоби вираження емоцій: великі літери, знаки оклику, повторення символів або слів, що підсилюють певний емоційний тон. Наприклад, фраза «Я ТАК ЩАСЛИВИЙ!!!» має набагато сильніше емоційне забарвлення, ніж звичайне «Я так щасливий». Використання цих стилістичних елементів часто сигналізує про підвищену емоційність повідомлення, що робить їх важливими маркерами при аналізі.

Крім того, соціальні мережі є місцем, де активно використовуються сарказм та іронія. Це може викликати труднощі в автоматизованому аналізі емоцій, оскільки стандартні алгоритми часто розглядають слова лише в прямому значенні, не враховуючи прихованого підтексту. Наприклад, фраза «Прекрасно, просто чудово!» може означати і розчарування, і захоплення залежно від контексту.

Таким чином, текстові дані в соціальних мережах є унікальними за своєю природою і вимагають спеціалізованих методів обробки. Ефективні системи автоматизованого розпізнавання емоцій повинні враховувати специфіку цих даних, зокрема їхню неформальність, багатошаровість та динамічність, щоб коректно і точно визначати емоційний стан користувачів.

1.5 Висновки до Розділу 1

Машинне навчання відіграє важливу роль у процесі класифікації емоцій у текстах, забезпечуючи автоматизацію аналізу та підвищуючи точність результатів. Методи обробки тексту, такі як токенізація, лематизація, стемінг і векторизація, дозволяють ефективно перетворювати текстові дані у формат, зручний для машинного навчання. Алгоритми, такі як наївний байєсівський класифікатор, логістична регресія, дерева рішень, а також сучасні нейронні мережі (LSTM, BERT), показують високу ефективність у задачах розпізнавання емоційного забарвлення текстів.

Окрему увагу було приділено особливостям текстових даних у соціальних мережах, які відрізняються своєю неструктурованістю, використанням сленгу, емодзі та хештегів, а також динамічністю і швидкою зміною контенту. Для успішного розпізнавання емоцій у таких текстах системи машинного навчання повинні враховувати ці специфічні характеристики та бути здатними адаптуватися до нових трендів і стилістичних особливостей онлайн-спілкування.

Таким чином, розділ охоплює базові концепції, методи та виклики, пов'язані з автоматизованим аналізом емоційного стану людини на основі текстових даних, зокрема у соціальних мережах, і закладає основу для подальшого поглибленого дослідження та розробки ефективних алгоритмів.

РОЗДІЛ 2
МЕТОДИКА ТА ЗАСОБИ АНАЛІЗУ ТЕКСТОВИХ ДАНИХ

2.1 Постановка задачі розпізнавання емоцій

Розпізнавання емоційного стану людини за текстовими даними є однією з актуальних проблем у сучасній сфері штучного інтелекту. Зростання обсягів текстів у соціальних мережах зумовлює необхідність автоматизації процесу аналізу емоцій, що має важливе значення для бізнесу, психології, маркетингу, соціології та інших галузей. Розробка моделей, здатних точно визначати емоційний стан людини, дозволяє отримувати цінну інформацію для прийняття рішень у різних контекстах, таких як аналіз суспільної думки, підтримка клієнтів або моніторинг психоемоційного стану.

Мета роботи полягає у створенні теоретичної та практичної основи для автоматизованого розпізнавання емоційного стану людини на основі текстових даних із використанням сучасних методів машинного навчання. Основним завданням є розробка та впровадження системи, яка забезпечує високу точність класифікації емоцій та може бути інтегрована в реальні додатки, зокрема для аналізу даних із соціальних мереж або у чат-ботах.

Вхідна інформація складається з текстових даних, таких як коментарі, публікації чи відгуки, отриманих із соціальних мереж (наприклад, Twitter). Дані можуть бути взяті з відкритих джерел або з використанням існуючих розмічених наборів, класифікованих за емоційними категоріями. Перед подачею до моделей дані проходять етапи очищення, нормалізації та токенізації.

Вихідна інформація включає емоційні категорії, присвоєні кожному текстовому фрагменту (наприклад, «радість», «сум», «гнів»), а також статистичні звіти про розподіл емоцій у проаналізованих текстах.

Вимоги до забезпечення включають:

1. Математичне забезпечення:

a. Застосування методів обробки тексту: токенізація, лематизація, видалення шумів та зупинкових слів.

b. Використання алгоритмів машинного навчання, таких як Logistic Regression, SVM, Random Forest, XGBoost, Naive Bayes та Decision Tree.

2. Програмне забезпечення:

a. Застосування бібліотек для обробки тексту (NLTK, spaCy).

b. Використання інструментів для машинного навчання, зокрема scikit-learn, та фреймворків для візуалізації (seaborn, matplotlib).

c. Використання інструментів для створення інтерактивного додатку (streamlit).

Засоби реалізації. Для обробки тексту, навчання моделей і створення інтерактивного додатку використовується мова програмування Python та його бібліотеки (pandas, scikit-learn, Streamlit тощо). Дані подаються в попередньо обробленому вигляді, а моделі навчаються на основі векторизації тексту з використанням TF-IDF.

Етапи проектування включають в себе:

1. Попередня обробка даних: збір текстових даних, очищення, нормалізація, токенізація та створення розміченого корпусу.

2. Навчання моделей: розробка і тестування алгоритмів Logistic Regression, SVM, Random Forest, Naive Bayes та Decision Tree, порівняння їхньої продуктивності за метриками точності та повноти.

3. Тестування та аналіз: оцінка ефективності моделей на окремій тестовій вибірці, аналіз помилок та оптимізація параметрів.

4. Розробка додатку: створення інтерактивного інтерфейсу для демонстрації роботи моделей і візуалізації результатів класифікації.

В результаті реалізована система повинна бути здатною автоматично класифікувати текстові повідомлення за емоційними категоріями з високою точністю. Крім того, необхідна реалізація інтерактивного додатку, який дозволяє користувачам перевіряти емоційне забарвлення текстів у реальному часі.

Завдяки виконанню поставлених завдань, реалізована система зможе забезпечити ефективне автоматизоване розпізнавання емоцій, адже ефективне виконання етапів проектування забезпечить створення моделі, що відповідає високим вимогам до точності, швидкодії та масштабованості.

2.2 Вибір інструментів для попередньої обробки тексту

У процесі розпізнавання емоцій у текстових даних ключовим етапом є вибір та застосування відповідних інструментів для попередньої обробки тексту, що дозволяє підготувати дані до подальшого аналізу та забезпечити високу якість результатів класифікації. Попередня обробка тексту включає низку методів очищення, нормалізації та структуризації текстових даних.

Одним із перших кроків є завантаження бібліотек, що надають можливості для роботи з текстовими даними. Для цього було використано такі популярні бібліотеки, як pandas та numpy, вони забезпечують зручність обробки та аналізу даних у табличному форматі. Крім того, бібліотека nltk надає інструменти для роботи з текстом, включаючи токенізацію, видалення стоп-слів і лематизацію. Допоміжними засобами для обробки тексту виступають модулі string і re, які використовуються для видалення пунктуації та числових символів за допомогою регулярних виразів.

На етапі очищення тексту ключовим завданням є зменшення «шуму» в даних, що досягається за допомогою спеціальної функції, яка видаляє всі символи пунктуації та числа. Такий підхід дозволяє уникнути зайвих елементів у текстових даних, які не несуть змістовного навантаження.

Наступним етапом є токенізація тексту, яка полягає в розділенні тексту на окремі слова або токени. Для цього текст приводиться до нижнього регістру, після чого він розбивається на слова за допомогою регулярних виразів. Токенізація є важливим кроком, оскільки вона забезпечує базову одиницю текстового аналізу – слово.

Видалення стоп-слів є ще одним важливим кроком у процесі обробки тексту. Стоп-слова – поширені слова, які не несуть значного смислового навантаження (наприклад, «the», «and», «it»). Для їх видалення використовується список стоп-слів, наданий бібліотекою nltk. Крім того, цей список вручну розширено специфічними словами, характерними для даних, але не суттєвими для визначення емоцій, такий підхід дозволяє зосередитися на аналізі більш інформативних слів.

Для подальшого зменшення розмірності текстових даних та забезпечення їх нормалізації використовується лематизація. Лематизація – процес приведення слова до його базової форми. Наприклад, слова «running» та «ran» будуть зведені до основної форми «run», що дозволяє зменшити кількість різних форм слів, що використовуються в тексті, зберігаючи при цьому їх лексичне значення. Для реалізації лематизації застосовується лематизатор з бібліотеки nltk.

Аналіз довжини тексту є важливим аспектом дослідження текстових даних. Для цього обчислюється кількість символів і слів у кожному тексті. Результати аналізу візуалізуються за допомогою графіків, що дозволяє зрозуміти розподіл довжини текстів у даних. Такий аналіз також проводиться з урахуванням емоційних категорій, що допомагає виявити залежності між довжиною тексту та його емоційним забарвленням.

Для ідентифікації ключових слів, характерних для кожної емоції, здійснюється аналіз частоти вживання слів. Цей підхід дозволяє визначити найпоширеніші слова, які асоціюються з певною емоційною категорією. Для кожної емоції обчислюється частота використання слів, і результати візуалізуються у вигляді графіків, що дає змогу отримати інтуїтивне розуміння того, які слова найчастіше зустрічаються в текстах, що належать до тієї чи іншої емоції. Завдяки цьому за необхідності загальні слова, що зустрічаються для кожної емоції можуть бути додані до списку стоп-слів.

Таким чином, обрані інструменти та методи дозволяють ефективно здійснювати попередню обробку текстових даних. Їх застосування забезпечує високу якість підготовки даних, що є ключовим фактором для успішного

виконання задачі розпізнавання емоцій за допомогою алгоритмів машинного навчання.

2.3 Формування корпусу текстових даних

Формування корпусу текстових даних є одним із ключових етапів у процесі розпізнавання емоційного стану. Якість даних, що використовуються для навчання та тестування моделей машинного навчання, суттєво впливає на точність та надійність отриманих результатів. Для формування корпусу тексту можуть застосовуватися різноманітні підходи залежно від завдання, доступності даних та особливостей текстового середовища.

Насамперед формування корпусу текстових даних може починатися з визначення джерела, з якого будуть отримані текстові дані. Для задач розпізнавання емоцій це можуть бути соціальні мережі, форуми, блоги, новинні сайти, відгуки користувачів, електронні листи, чат-боти тощо. Для аналізу емоційного забарвлення текстів, створених у соціальних мережах, можна використовувати API таких платформ, як Twitter, Facebook чи Reddit. Збір даних із цих джерел часто передбачає використання інструментів веб-скрапінгу або відповідних програмних інтерфейсів для автоматизованого отримання повідомлень, коментарів або інших текстових форматів.

Дуже важливим є процес визначення необхідного обсягу даних, які будуть використані у дослідженні. Набір даних повинен бути репрезентативним, тобто враховувати різноманіття мовних виразів, стилів написання, тематики текстів і типів емоцій, які потрібно розпізнавати. Для аналізу необхідних категорій емоцій (наприклад, позитивних, негативних та нейтральних) слід забезпечити рівномірний розподіл текстів для кожної з цих категорій, щоб уникнути дисбалансу в навчанні моделі.

Наступним аспектом є структура набору даних. У більшості задач корпус текстів представляється у вигляді таблиці, де одна з колонок містить текст, а інша

– відповідну мітку (емоцію). Такий підхід дозволяє ефективно організовувати дані та спрощує подальшу обробку.

Не менш важливою складовою є попередня обробка корпусу текстів. У початковому вигляді тексти можуть містити помилки, зайві символи, дублікати чи інші елементи, які не несуть смислового навантаження. Наприклад, набір даних, взятий із соціальних мереж, може включати хештеги, згадки, URL-адреси чи емодзі. У такому випадку виконуються наступні дії: очищення тексту від непотрібних елементів, приведення тексту до нижнього регістру, видалення стоп-слів, токенізація, стемінг або лематизація. Такі операції дозволяють стандартизувати дані та виділити ключову інформацію, необхідну для аналізу.

Особливої уваги вимагає анотація даних. У готових наборах даних мітки вже визначені. Однак, якщо дослідник самостійно формує корпус, йому доводиться забезпечувати точність анотації. Це може виконуватись вручну за участі експертів або автоматично, з використанням спеціалізованих інструментів.

Після створення та обробки корпусу даних необхідно здійснити його поділ на навчальний та тестовий набори. Це дозволяє навчати модель на одній частині даних та оцінювати її якість на іншій. Загальноприйнятим підходом є розділення даних у співвідношенні 80:20 або 70:30, що допомагає забезпечити достатній обсяг даних для навчання, не жертвуючи якістю тестування.

Для виконання даної роботи використаний готовий набір даних. Його перевагою є те, що він містить чітко визначені мітки емоцій, представлений у зручному форматі та є достатньо репрезентативним, оскільки містить в собі 20 000 записів. Однак, використання готових наборів даних не виключає необхідності попередньої перевірки їхньої якості.

У процесі аналізу цього набору даних виявлено достатньо вагомі дисбаланси між кількістю записів для різних емоцій. Проблема дисбалансу даних виникає тоді, коли в наборі даних суттєво нерівномірно представлені різні класи. Наприклад, у наборі емоційних текстів може бути набагато більше записів, позначених як «нейтральні», ніж «позитивні» або «негативні». Такий дисбаланс

може суттєво вплинути на якість роботи моделі машинного навчання, оскільки модель буде схильна «надавати перевагу» класам із більшою кількістю зразків, ігноруючи менш представлені класи, що призводить до зниження точності, зокрема, для рідкісних класів.

Для вирішення цієї проблеми використовуються методи балансування, які спрямовані на вирівнювання частот представленості класів у навчальному наборі даних. Серед таких методів можна виділити надсемплінг, підсемплінг та аугментацію текстів.

Кожен із цих підходів має свої переваги та обмеження. Надсемплінг (oversampling – процес збільшення кількості зразків для менш представлених класів) дозволяє зберегти всі доступні дані, але може збільшити час навчання моделі та ризик перенавчання. Це можна досягти шляхом дублювання існуючих прикладів або створенням нових синтетичних даних. Наприклад, метод SMOTE (Synthetic Minority Oversampling Technique) генерує синтетичні зразки, використовуючи векторні представлення тексту (рис. 2.1) [16].

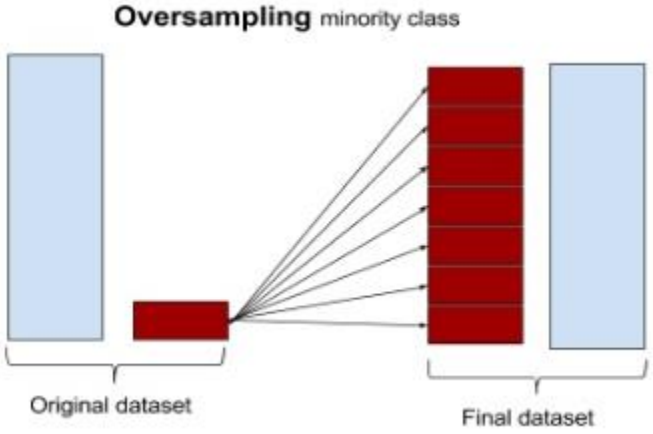

Рисунок 2.1 – Надсемплінг датасету

Надсемплінг допомагає збільшити вагу менш представлених класів у моделі, що робить її результати більш збалансованими. Проте дублювання даних може призвести до перенавчання, особливо якщо додаткові дані не додають нової інформації.

Підсемплінг (undersampling – зменшення кількості зразків у більш представлених класах) забезпечує простоту реалізації, але може призвести до втрати важливих даних. Суть методу полягає в досягненні балансу між класами шляхом випадкового видалення частини даних із переважаючих класів. Підхід дозволяє вирівняти розподіл класів, зменшуючи вплив домінуючого класу на модель (рис. 2.2) [17].

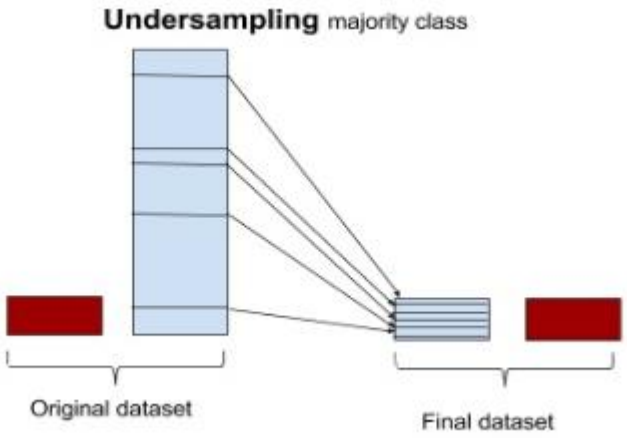

Рисунок 2.2 – Підсемплінг датасету

Однак, метод підсемплінгу може призвести до втрати важливої інформації, особливо якщо видалені приклади є унікальними або критично важливими для задачі.

Аугментація текстів (augmentation – штучне створення нових текстових зразків для менш представлених класів шляхом модифікації існуючих) є гнучким методом, який дозволяє створювати різноманітні дані, проте вимагає додаткових обчислювальних ресурсів і може бути складною в реалізації.

Наприклад, можна замінити слова у тексті їхніми синонімами, змінити порядок слів або додати переклади через проміжну мову (рис. 2.3). Генеративні моделі, такі як GPT або BERT, також можуть бути використані для створення нових текстів. Аугментація підвищує різноманітність даних і робить модель стійкішою до змін у формулюваннях тексту.

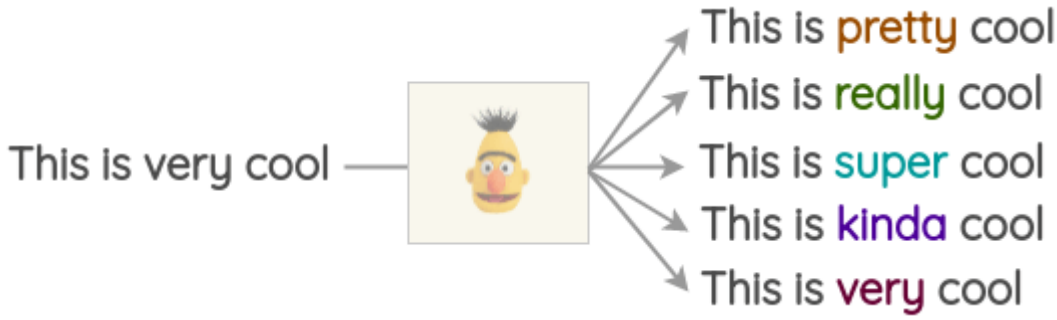

Рисунок 2.3 – Аугментація текстів

Однак, у разі невдалої трансформації текстів існує ризик втрати сенсу або створення некоректних даних, що додає суттєвих ризиків у навчанні моделі.

Формування корпусу текстових даних є багатоетапним процесом, що вимагає врахування багатьох факторів. Від якості корпусу залежить точність моделі машинного навчання, а також можливість її використання в реальних умовах. Правильний підхід до збору, обробки та структурування даних є основою успішного вирішення задачі розпізнавання емоцій у текстах.

2.4 Обґрунтування вибору моделей машинного навчання

Для успішного виконання задачі розпізнавання емоцій у текстових даних важливо забезпечити правильний вибір моделей машинного навчання, які найбільш підходять для цієї задачі. Вибір ґрунтується на аналізі характеристик алгоритмів, їхньої здатності працювати з текстовими даними та врахуванні особливостей наявного набору даних. Основна мета – досягнення високої точності класифікації, враховуючи специфіку задачі, включаючи дисбаланс класів та різнорідність текстових даних.

Для виконання задачі розпізнавання емоцій використовуються різні моделі машинного навчання, серед яких Logistic Regression, Support Vector Machine (SVM) з реалізацією через Stochastic Gradient Descent (SGD), Random Forest, Multinomial Naive Bayes та Decision Tree. Кожна з цих моделей має унікальні

властивості, які роблять її відповідною для певних аспектів задачі класифікації текстів.

Logistic Regression обрана як базова модель через її простоту та здатність забезпечити швидке навчання. Ця модель є одним із найпоширеніших методів для класифікації, що ґрунтується на знаходженні лінійної залежності між вхідними ознаками та мітками класів. Вона дозволяє оцінити значущість кожної ознаки, що сприяє кращому розумінню структури даних. Крім того, Logistic Regression добре підходить для роботи з текстовими даними, представленими у векторному вигляді за допомогою методів CountVectorizer та TfidfTransformer.

SVM з реалізацією через SGD був обраний через його здатність забезпечувати високу точність класифікації навіть у складних задачах. Метод стохастичного градієнтного спуску дозволяє ефективно працювати з великими наборами даних і налаштовувати модель відповідно до специфіки задачі. Завдяки своїй гнучкості та здатності знаходити оптимальну гіперплощину між класами, SVM став одним із провідних методів для задач класифікації текстів.

Деревоподібні моделі, такі як Decision Tree та Random Forest, були включені до аналізу через їхню здатність обробляти дані з різними структурами. Decision Tree будує послідовність правил, які дозволяють розподілити дані за класами. Однак, цей підхід може бути схильним до перенавчання, особливо на невеликих наборах даних. Random Forest, як ансамблевий метод, вирішує цю проблему шляхом створення множини дерев і об'єднання їхніх результатів, що забезпечує більш стійкі та точні прогнози.

Multinomial Naïve Bayes є класичним методом для роботи з текстовими даними, який ґрунтується на теорії ймовірностей. Такий підхід дозволяє швидко навчати модель навіть на великих наборах даних, що робить його популярним вибором для задач класифікації текстів.

Для підготовки текстових даних до навчання всіх моделей використовувалися методи векторизації, зокрема CountVectorizer і TfidfTransformer. Векторизація дозволяє перетворювати текст у числовий формат, який є зрозумілим для моделей машинного навчання.

CountVectorizer створює матрицю частот слів, що дозволяє врахувати частоту появи кожного слова у текстах.

TfidfTransformer додає вагові коефіцієнти до цих частот, що знижує значення часто вживаних, але менш інформативних слів, таких як «the» або «and», що забезпечує більш релевантну інформацію для класифікації.

Вибір моделей обумовлений необхідністю оцінити різні підходи до класифікації текстових даних, включаючи лінійні, деревоподібні та ансамблеві методи. Такий підхід дозволяє враховувати різні аспекти задачі, включаючи дисбаланс класів, розмірність даних і складність текстових структур. Аналіз результатів роботи кожної моделі дозволяє визначити їхні сильні сторони та обмеження, а також зробити висновки щодо їхньої придатності для задачі розпізнавання емоцій у текстах.

2.5 Проектні рішення для реалізації системи

Для створення системи розпізнавання емоцій у тексті було розроблено й реалізовано комплексне рішення, яке охоплює кілька ключових етапів: попередню обробку даних, векторизацію тексту, навчання моделей машинного навчання та оцінювання їхньої ефективності. Обрані підходи і засоби дозволяють забезпечити високу точність роботи системи та її масштабованість.

Попередня обробка текстових даних. На першому етапі виконувалася попередня обробка текстових даних. Для цього було розроблено кастомний клас TextPreprocessor, який об'єднує всі необхідні операції для очищення тексту, включаючи видалення пунктуації та чисел, токенізацію, видалення стоп-слів і лематизацію, що забезпечує стандартизацію текстів та підготовку їх до векторизації. Реалізація класу дозволяє легко масштабувати процес обробки на нові дані та інтегрувати його у виробниче середовище.

Векторизація текстових даних. Текстові дані перетворюються у числовий формат за допомогою методів CountVectorizer і TfidfTransformer. CountVectorizer дозволяє представити текст як матрицю частот слів, тоді як TfidfTransformer

знижує вагу часто вживаних, але малозначущих слів, забезпечуючи краще відображення семантики тексту. Ці підходи інтегровані в пайплайн, що автоматизує процес обробки тексту перед подачею на вхід моделі.

Навчання моделей машинного навчання. Для класифікації емоцій у текстах обрано кілька алгоритмів машинного навчання. Кожна з моделей була навчена на векторизованих даних із використанням навчальної вибірки, яка становила 70% від загального обсягу даних, інші 30% використовувались для тестування. Для забезпечення точності роботи моделей використовувалася стандартизована процедура оцінювання, яка включала розрахунок метрик точності (accuracy), повноти (recall), F1-міри та побудову матриць плутанини, що забезпечило комплексну перевірку ефективності кожної моделі.

Інтеграція моделей у систему. Для інтеграції обробки текстів та класифікації емоцій у єдину систему було використано бібліотеку Pipeline з scikit-learn. Це дозволяє об'єднати всі етапи обробки тексту, векторизації та класифікації у єдину послідовність дій, що значно спрощує використання системи. Створена модель була серіалізована за допомогою бібліотеки dill, що дозволяє її легко завантажувати та використовувати у виробничих середовищах або інших проєктах.

Візуалізація результатів. Для аналізу роботи моделей та інтерпретації результатів класифікації розроблено графічні представлення, зокрема, графіки розподілу довжини текстів, розподілу ключових слів для кожної емоції та порівняння точності моделей. Візуалізація дозволила оцінити ефективність кожної моделі та ідентифікувати можливі області для покращення.

Створення інтерактивного додатку. Для демонстрації роботи системи розроблено інтерактивний веб-додаток із використанням бібліотеки Streamlit. Додаток забезпечує зручний інтерфейс для введення текстів, автоматичної класифікації емоцій та візуалізації результатів у реальному часі. Користувач може ввести текст, після чого додаток визначає емоційну категорію та показує ймовірність класифікації. Для більш детального аналізу додаток також надає

можливість перегляду розподілу ймовірностей у вигляді стовпчикових і радарних діаграм.

Інтерфейс додатку дозволяє швидко тестувати модель і забезпечує зрозуміле представлення результатів навіть для користувачів без технічної підготовки. Завдяки своїй модульності додаток може бути легко адаптований до нових моделей або інших завдань, пов’язаних із класифікацією текстів.

Технологічні засоби. Реалізація системи здійснена мовою програмування Python із використанням таких бібліотек, як pandas, numpy, nltk, scikit-learn, matplotlib і seaborn. Обробка даних та навчання моделей виконувалися в середовищі Jupyter Notebook, а розробка веб-додатку за допомогою streamlit забезпечила інтерактивність та зручність у тестуванні моделі.

Переваги обраного рішення. Створене проєктне рішення поєднує в собі ефективність і простоту використання. Завдяки модульності система легко адаптується до нових задач або змін у вихідних даних. Обрані моделі машинного навчання забезпечують високу точність класифікації, а пайплайн дозволяє автоматизувати всі етапи обробки тексту та навчання моделей. Крім того, використання сучасних бібліотек і методів забезпечує швидкість роботи системи навіть на великих обсягах даних.

Таким чином, розроблене проектне рішення є універсальним інструментом для автоматизованого розпізнавання емоцій у текстових даних і може бути адаптоване для використання у різних сферах, таких як аналіз соціальних мереж, підтримка клієнтів або моніторинг громадської думки.

2.6 Висновки до Розділу 2

У другому розділі було викладено методологію створення системи розпізнавання емоцій у текстових даних. Обґрунтовано вибір підходів до обробки тексту, визначено інструменти для попередньої обробки та формування корпусу даних, а також вибір моделей машинного навчання.

Особлива увага приділялася попередній обробці тексту, яка включає очищення, токенізацію, видалення стоп-слів та лематизацію, що забезпечує підготовку даних для подальшої векторизації. Використання методів CountVectorizer і TfidfTransformer дозволяє перетворити текстові дані у формат, зрозумілий для моделей машинного навчання. Обрані моделі навчаються на цьому векторизованому корпусі текстів. Їхня ефективність оцінюється за допомогою стандартних метрик, таких як точність, повнота, F1-міра та матриця плутанини.

Крім того, у розділі описано створення інтерактивного додатку для демонстрації роботи системи розпізнавання емоцій. Цей додаток реалізований на основі бібліотеки Streamlit і забезпечує зручний інтерфейс для введення текстів та отримання результатів класифікації. Після введення текстового фрагмента додаток відображає передбачену емоційну категорію, ймовірність передбачення, а також графічні візуалізації розподілу ймовірностей у вигляді стовпчикової та радарної діаграм.

Додаток є важливою складовою системи, оскільки дозволяє не лише перевіряти роботу моделей у реальному часі, але й демонструвати результати у зрозумілому для користувача форматі. Завдяки використанню інтерактивного інтерфейсу додаток може бути легко адаптований для подальшого вдосконалення або інтеграції у виробничі системи.

Таким чином, у розділі наведено обґрунтовані рішення, які забезпечують високу точність і масштабованість системи, а також описано реалізацію додатку, що сприяє зручному тестуванню моделей та отриманню аналітичних даних. Це створює надійну основу для подальших експериментальних досліджень і практичного застосування розробленої системи.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

3.1 Опис експериментального середовища

Експериментальне середовище для виконання задач розпізнавання емоцій у текстових даних було організоване таким чином, щоб забезпечити ефективність і надійність усіх етапів обробки, навчання моделей та оцінювання їхньої продуктивності. Основу апаратного забезпечення склав комп'ютер із процесором AMD Ryzen 5 2600. Навчання моделей виконувалося на центральному процесорі, без використання графічного прискорювача, що є достатнім для розв'язання задач із невеликими наборами даних.

Програмне забезпечення реалізовано за допомогою мови Python, яка забезпечує гнучкість і зручність у роботі з текстовими даними. Для роботи з таблицями та числовими обчисленнями використовувалися бібліотеки pandas і numpy. Попередня обробка текстів виконувалася за допомогою бібліотеки nltk, яка забезпечує функції токенізації, видалення стоп-слів і лематизації. Для побудови моделей машинного навчання та їхнього навчання використовувалася бібліотека scikit-learn. Створення графічних представлень реалізовано за допомогою matplotlib і seaborn, що дозволило візуалізувати результати роботи моделей і розподіл даних.

Всі етапи реалізації виконувалися в інтерактивному середовищі Jupyter Lab. Це середовище забезпечує зручність розробки, тестування та документування процесів у реальному часі. Використання Jupyter Lab дозволило інтегрувати код, текст і результати в єдиному інтерфейсі, що значно спростило аналіз та внесення змін до алгоритмів.

Результати експериментів серіалізувалися у форматі .pkl за допомогою бібліотеки dill. Це забезпечило можливість повторного використання моделей без необхідності їхнього повторного навчання, а також спростило інтеграцію системи в інші проекти.

Для інтерактивного тестування роботи системи було створено веб-додаток за допомогою бібліотеки Streamlit. Додаток забезпечує користувачам можливість введення текстів і автоматичного визначення емоційної категорії, а також ймовірності передбачення.

Виконання експериментів передбачало послідовне навчання кожної з моделей на підготовлених даних і оцінювання їхньої ефективності за допомогою стандартних метрик, таких як точність, повнота, F1-міра та матриця плутанини. Зокрема, оцінка результатів дозволила зробити висновки щодо переваг і недоліків кожного алгоритму в контексті розпізнавання емоцій у текстах.

Експериментальне середовище було організоване таким чином, щоб забезпечити надійність, автоматизацію та легкість повторного використання, що дозволило досягти високої ефективності в реалізації, тестуванні та аналізі моделей машинного навчання.

3.2 Реалізація алгоритмів розпізнавання емоцій

3.2.1 Підготовка набору даних. Реалізація алгоритмів розпізнавання емоцій охоплює інтеграцію всіх етапів обробки тексту, векторизації та навчання моделей машинного навчання для класифікації текстових даних.

Етап завантаження та попереднього ознайомлення з даними є ключовим для розуміння їхньої структури та особливостей. Набір даних зберігається в форматі CSV файлу та містить текстові повідомлення, опубліковані в соціальній мережі Twitter, разом із відповідними мітками емоцій. Кожен запис у цьому наборі складається з двох основних стовпців: «Text» та «Emotion». Стовпець «Text» містить текстові повідомлення, які варіюються за довжиною та змістом, тоді як стовпець «Emotion» представляє категорії емоцій, до яких належить повідомлення.

Емоційні категорії включають позитивні, негативні та нейтральні емоції, такі як «радість», «сум», «гнів», «здивування», «любов» та інші. Дисбаланс у розподілі категорій є однією з основних характеристик цього набору, оскільки

деякі емоції представлені значно частіше за інші, що створює додаткові виклики під час навчання моделей машинного навчання та вимагає застосування методів балансування даних.

Загалом набір даних включає 20 000 записів, що забезпечує достатній обсяг інформації для навчання та тестування моделей. Середня довжина текстових повідомлень варіюється від кількох слів до кількох речень, що дозволяє моделювати різноманітні текстові структури. Очищення та підготовка цих даних є важливими етапами, які забезпечують високу якість вхідної інформації для машинного навчання.

Для попереднього аналізу даних виконується перегляд кількох перших та останніх рядків таблиці, що дозволяє ознайомитися з вмістом і переконатися у правильності завантаження. Також здійснюється перевірка наявності пустих значень таблиці, що показує кількість відсутніх значень в датасеті.

	Text	Emotion
0	i didnt feel humiliated	sadness
1	i can go from feeling so hopeless to so damned...	sadness
2	im grabbing a minute to post i feel greedy wrong	anger
3	i am ever feeling nostalgic about the fireplac...	love
4	i am feeling grouchy	anger
...	...	...
19995	im having ssa examination tomorrow in the morn...	sadness
19996	i constantly worry about their fight against n...	joy
19997	i feel its important to share this info for th...	joy
19998	i truly feel that if you are passionate enough...	joy
19999	i feel like i just wanna buy any cute make up ...	joy

```
Dataset size: (20000, 2)
Columns are: Index(['Text', 'Emotion'], dtype='object')
Text       0
Emotion    0
dtype: int64
```

Рисунок х.х – Перегляд завантажених даних

Для розуміння розподілу емоцій виконується попередній аналіз, що дозволяє оцінити кількість записів для кожної категорії емоцій, виявити дисбаланс у даних та врахувати це під час навчання моделей машинного навчання (рис. 3.1).

Таблиця 3.1 – Кількість записів для кожної емоції

№	Emotion	Count
1	joy	6761
2	sadness	5797
3	anger	2709
4	fear	2373
5	love	1641
6	surprise	719

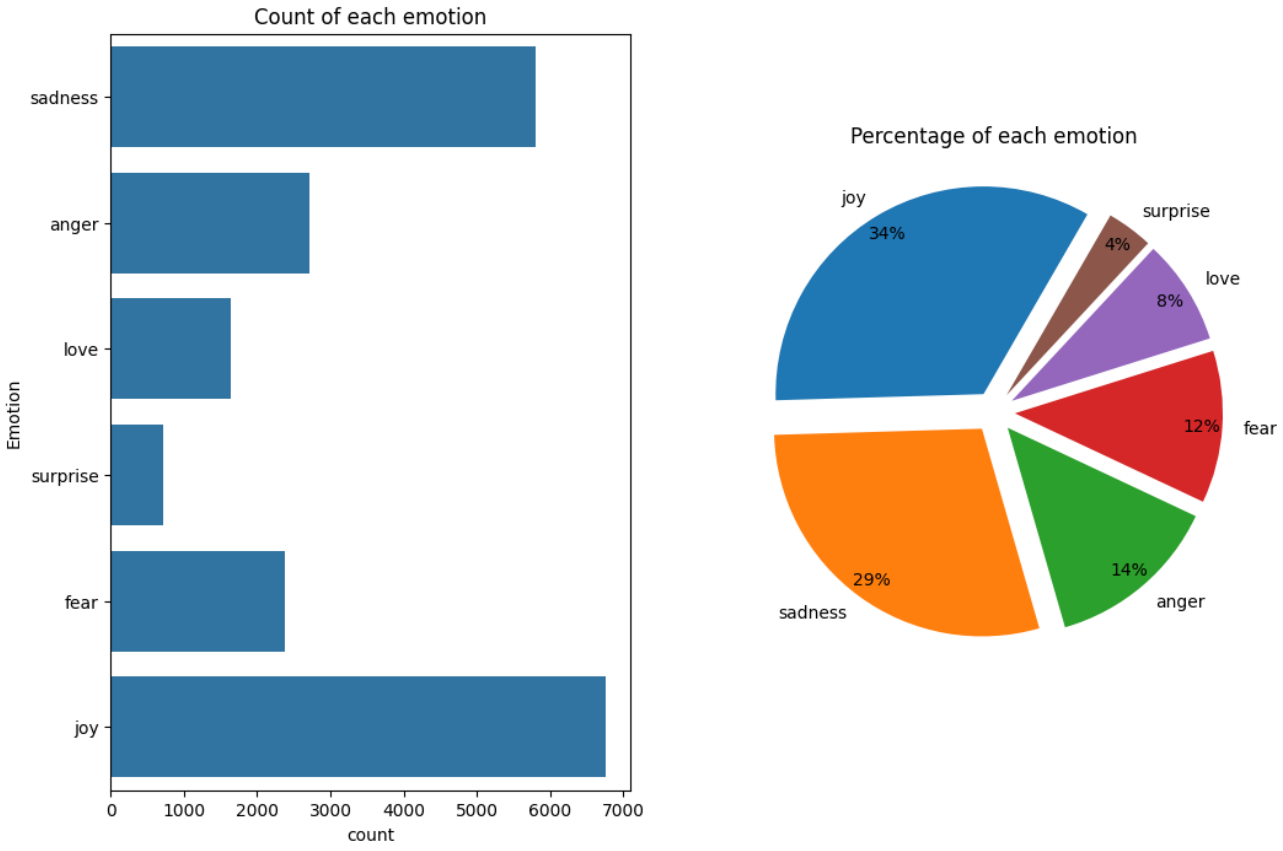

Рисунок 3.1 – Співвідношення категорій емоцій в датасеті

Результати аналізу розподілу емоцій у наборі даних показують наступну картину. Найбільшою категорією є «joy» із 6761 записами, категорія «sadness» займає приблизно таку ж кількість записів – 5797 що вказує на домінування у наборі позитивних та повідомлень з сумним змістом. Емоція «anger» представлена 2709 записами, а «fear» – 2373, що вказує на баланс між негативними емоціями та емоціями гніву. Найменш поширена категорія «surprise» із 719 записами також відіграє важливу роль у наборі.

Такий розподіл вказує на дисбаланс у даних, що може вплинути на ефективність моделей машинного навчання. Для врахування цього аспекту можливе застосовувати методи балансування даних або аугментації для менш представлених категорій.

Хоча балансування даних є важливою стратегією в задачах машинного навчання, особливо коли дисбаланс між класами може негативно вплинути на результати, однак, у деяких випадках, таких як у задачі розпізнавання емоцій у текстах, балансування даних не завжди є доцільним і може навіть погіршити результати. Це обґрунтовується наступними причинами:

Перш за все, розподіл емоцій у наборі даних часто відображає реальний світ, де деякі емоції є більш поширеними, ніж інші. Наприклад, позитивні або нейтральні емоції, такі як «joy» або «neutral», зустрічаються частіше, ніж емоції здивування чи злості. Збереження цього природного розподілу дозволяє моделі краще узагальнювати і адаптуватися до реальних сценаріїв, де дані мають подібний дисбаланс. Штучне вирівнювання класів через надсемплінг чи підсемплінг може порушити цю природну пропорцію, що призведе до менш реалістичних результатів.

По-друге, збільшення кількості прикладів менш представлених класів через надсемплінг (наприклад, дублювання даних або створення синтетичних зразків) може спричинити проблему перенавчання. Модель буде занадто добре адаптуватися до доданих прикладів, але її узагальнююча здатність на нових даних знизиться. Особливо це стосується невеликих категорій, таких як «surprise», де додані зразки можуть бути не повністю репрезентативними.

Підсемплінг, навпаки, зменшує кількість даних домінуючих класів, таких як «joy» або «sadness», що може призвести до втрати важливої інформації. Менша кількість зразків обмежує здатність моделі навчатися на різноманітних ситуаціях, що негативно позначиться на її продуктивності.

Третім аспектом є метрики оцінки моделі. Для дисбалансованих даних використовуються такі метрики, як повнота (recall) і F1-міра, які дозволяють оцінити якість класифікації для кожного класу окремо. Навіть якщо категорія є менш представленою, модель все одно здатна забезпечити високу точність (precision) і повноту для цих класів без необхідності балансування даних.

Нарешті, у задачах класифікації емоцій важливо не лише правильно класифікувати кожен клас, але й враховувати контекст, в якому ці дані будуть використовуватися. У реальних сценаріях (наприклад, аналіз соціальних мереж) часто важливіша точність класифікації більш поширених емоцій, оскільки вони мають більший вплив на загальні висновки. Балансування даних може штучно збільшити вагу рідкісних емоцій, що призведе до менш реалістичних результатів у таких випадках.

Таким чином, у задачі розпізнавання емоцій у текстах балансування даних є необов’язковим через збереження природного розподілу, ризики перенавчання та втрати даних, а також завдяки використанню відповідних метрик, які дозволяють адекватно оцінювати якість роботи моделі на кожному класі.

3.2.2 Аналіз текстових характеристик. Після первинної підготовки даних проводиться аналіз текстових характеристик, таких як довжина текстових повідомлень у символах та кількість слів у кожному повідомленні. До набору додається інформація про ці параметри, що дозволяє оцінити загальну структуру текстів і виявити можливі особливості, які можуть вплинути на подальшу обробку.

Довжина текстів у символах обчислюється шляхом підрахунку загальної кількості символів у кожному текстовому повідомленні, що дає змогу виявити середні та екстремальні значення довжин текстів, а також оцінити їхній розподіл.

Крім того підраховується кількість слів у текстах задля розуміння, наскільки різноманітними є тексти за обсягом інформації, яку вони містять.

Для представлення цих даних використовуються графіки розподілу, які показують частоту різних довжин текстів у символах і словах. Такі візуалізації дозволяють ідентифікувати типові довжини повідомлень, оцінити ступінь їхньої варіативності та виявити аномалії. Наприклад, дуже короткі повідомлення можуть не містити достатньої інформації для класифікації емоцій і потребувати виключення, тоді як надмірно довгі тексти можуть бути скорочені для забезпечення узгодженості довжин текстів у вибірці (рис. 3.2).

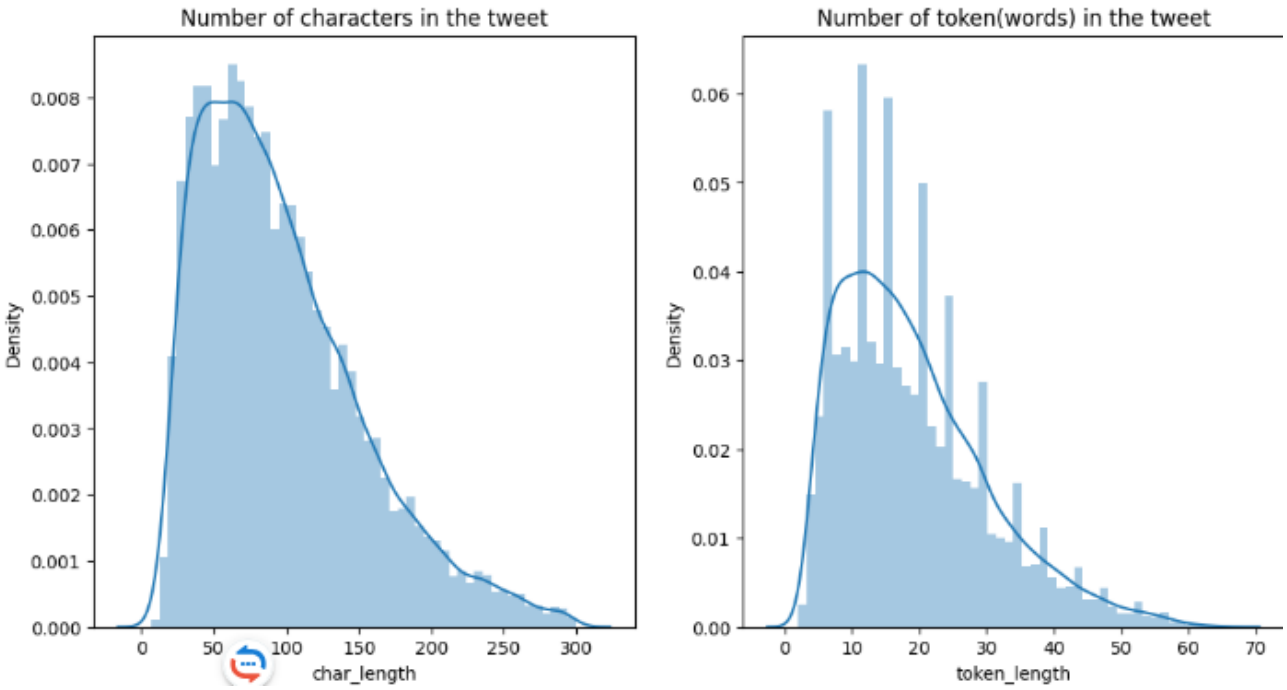

Рисунок 3.2 – Графіки розподілу довжини тексту у символах та словах

Додатково здійснюється аналіз розподілу довжин текстів залежно від емоційної категорії. Для цього створюються графіки розподілу довжин текстів у символах і словах окремо для кожної емоції, що дозволяє виявити характерні особливості текстів, пов’язаних із різними емоціями. Наприклад, повідомлення, що виражають радість або любов, можуть бути коротшими за тексти з емоціями тривоги чи смутку. Ці графіки також дозволяють оцінити ступінь накладення між

категоріями за їхніми текстовими характеристиками, що є важливим для подальшого навчання моделей (рис. 3.3).

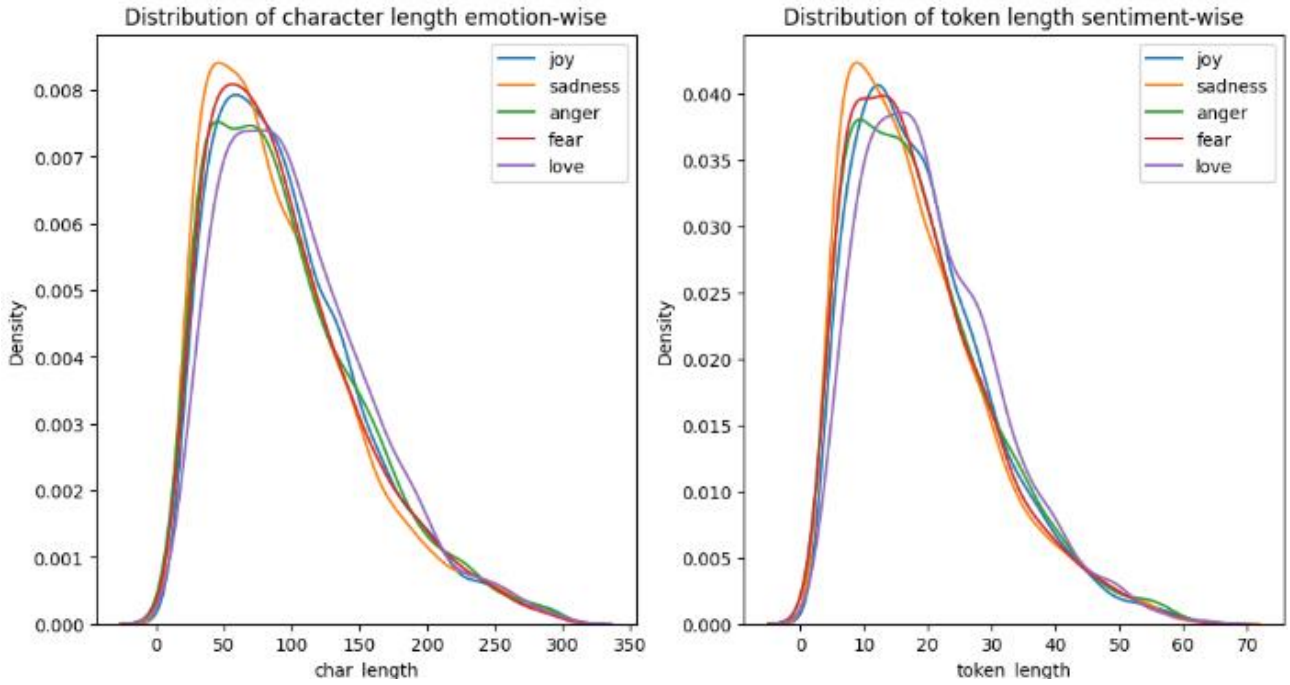

Рисунок 3.3 – Графіки розподілу довжини тексту у символах та словах залежно від емоційної категорії

Результати аналізу текстових характеристик допомагають краще зрозуміти структуру даних та прийняти обґрунтовані рішення щодо їхньої подальшої обробки, наприклад, виключення нерелевантних текстів або їхнє редагування, що може сприяти підвищенню якості підготовки до навчання моделей машинного навчання.

3.2.3 Попередня обробка текстових даних. На даному етапі роботи з даними проводиться ретельна підготовка текстів з набору даних. Цей процес має на меті очистити текстові дані від зайвої інформації, стандартизувати їх та підготувати до подальшого машинного навчання. Оскільки текстові дані з соціальних мереж часто містять помилки, спеціальні символи та інші елементи, які не несуть смислового навантаження, підготовка даних відіграє критично важливу роль у забезпеченні якості моделювання.

Для забезпечення високої якості вхідних текстів виконується повна обробка, що включає очищення текстів, видалення зайвих символів і підготовку до подальшої роботи з моделями машинного навчання. Використовується спеціально розроблений клас TextPreprocessor, який дозволяє автоматизувати всі ключові етапи обробки текстових даних.

На першому етапі виконується видалення пунктуації та числових символів. Символи пунктуації, такі як коми, крапки, дужки, лапки, не несуть значної смислової інформації для задач розпізнавання емоцій, їх видалення дозволяє очистити текст від зайвих елементів, які не несуть інформаційного навантаження для аналізу емоційного забарвлення. Далі текст переводиться у нижній регістр, що сприяє стандартизації текстів і дозволяє уникнути дублювання слів через різницю у регістрі (рис. 3.4).

	Text	Emotion	Tweet_punct
0	i didnt feel humiliated	sadness	i didnt feel humiliated
1	i can go from feeling so hopeless to so damned...	sadness	i can go from feeling so hopeless to so damned...
2	im grabbing a minute to post i feel greedy wrong	anger	im grabbing a minute to post i feel greedy wrong
3	i am ever feeling nostalgic about the fireplac...	love	i am ever feeling nostalgic about the fireplac...
4	i am feeling grouchy	anger	i am feeling grouchy
5	ive been feeling a little burdened lately wasn...	sadness	ive been feeling a little burdened lately wasn...
6	ive been taking or milligrams or times recomme...	surprise	ive been taking or milligrams or times recomme...
7	i feel as confused about life as a teenager or...	fear	i feel as confused about life as a teenager or...
8	i have been with petronas for years i feel tha...	joy	i have been with petronas for years i feel tha...
9	i feel romantic too	love	i feel romantic too

Рисунок 3.4 – Результат очищення тексту від пунктуації та інших символів

Токенізація забезпечує поділ тексту на окремі слова (токени), що дає змогу працювати з текстами на рівні окремих елементів. Наприклад, речення «I love music» після токенізації перетворюється на список слів [«I», «love», «music»].

Токенізація дозволяє працювати з текстом на рівні окремих слів, що є необхідним для подальшої векторизації.

Наступним етапом є видалення стоп-слів. Стоп-слова – часто вживані слова, такі як «the», «is», «and», які не несуть смислового навантаження, але впливають на результати моделювання через їхню велику частоту (рис. 3.5).

	Text	Emotion	Tweet_tokenized	Tweet_nonstop
0	i didnt feel humiliated	sadness	[i, didnt, feel, humiliated]	[humiliated]
1	i can go from feeling so hopeless to so damned...	sadness	[i, can, go, from, feeling, so, hopeless, to, ...	[go, feeling, hopeless, damned, hopeful, aroun...
2	im grabbing a minute to post i feel greedy wrong	anger	[im, grabbing, a, minute, to, post, i, feel, g...	[grabbing, minute, post, greedy, wrong]
3	i am ever feeling nostalgic about the fireplac...	love	[i, am, ever, feeling, nostalgic, about, the, ...	[ever, feeling, nostalgic, fireplace, know, st...
4	i am feeling grouchy	anger	[i, am, feeling, grouchy]	[feeling, grouchy]
5	ive been feeling a little burdened lately wasn...	sadness	[ive, been, feeling, a, little, burdened, late...	[feeling, little, burdened, lately, wasnt, sure]
6	ive been taking or milligrams or times recomme...	surprise	[ive, been, taking, or, milligrams, or, times,...	[taking, milligrams, times, recommended, amoun...
7	i feel as confused about life as a teenager or...	fear	[i, feel, as, confused, about, life, as, a, te...	[confused, life, teenager, jaded]
8	i have been with petronas for years i feel tha...	joy	[i, have, been, with, petronas, for, years, i,...	[petronas, years, petronas, performed, well, m...
9	i feel romantic too	love	[i, feel, romantic, too]	[romantic]

Рисунок 3.5 – Результат токенізації та видалення стоп-слів

Завершальним етапом обробки є лематизація, яка зводить слова до їхньої базової форми. Лематизація полягає у приведенні кожного слова до його базової форми (леми). Наприклад, слова «running», «ran» і «runs» після лематизації перетворюються на «run». Лематизація сприяє скороченню розмірності даних і покращує їхню узгодженість (рис. 3.6).

	Text	Emotion	Tweet_nonstop	Tweet_lemmatized
0	i didnt feel humiliated	sadness	[humiliated]	[humiliated]
1	i can go from feeling so hopeless to so damned...	sadness	[go, feeling, hopeless, damned, hopeful, aroun...	[go, feeling, hopeless, damned, hopeful, aroun...
2	im grabbing a minute to post i feel greedy wrong	anger	[grabbing, minute, post, greedy, wrong]	[grabbing, minute, post, greedy, wrong]
3	i am ever feeling nostalgic about the fireplac...	love	[ever, feeling, nostalgic, fireplace, know, st...	[ever, feeling, nostalgic, fireplace, know, st...
4	i am feeling grouchy	anger	[feeling, grouchy]	[feeling, grouchy]
5	ive been feeling a little burdened lately wasn...	sadness	[feeling, little, burdened, lately, wasnt, sure]	[feeling, little, burdened, lately, wasnt, sure]
6	ive been taking or milligrams or times recomme...	surprise	[taking, milligrams, times, recommended, amoun...	[taking, milligram, time, recommended, amount,...
7	i feel as confused about life as a teenager or...	fear	[confused, life, teenager, jaded]	[confused, life, teenager, jaded]
8	i have been with petronas for years i feel tha...	joy	[petronas, years, petronas, performed, well, m...	[petronas, year, petronas, performed, well, ma...
9	i feel romantic too	love	[romantic]	[romantic]

Рисунок 3.6 – Результат лематизації

Для кожної емоційної категорії створюється окремий список ключових слів. Ключовими словами вважаються найпоширеніші терміни, які найчастіше зустрічаються у текстах, що належать до певної емоційної категорії. Спочатку всі тексти однієї емоції об'єднуються в єдиний документ, після чого з отриманого документа виділяються найчастіше вживані слова з використанням методу підрахунку частотності.

Для наочного представлення результатів аналізу створюються горизонтальні стовпчасті графіки, які показують 25 ключових слів для кожної емоційної категорії (рис. 3.7).

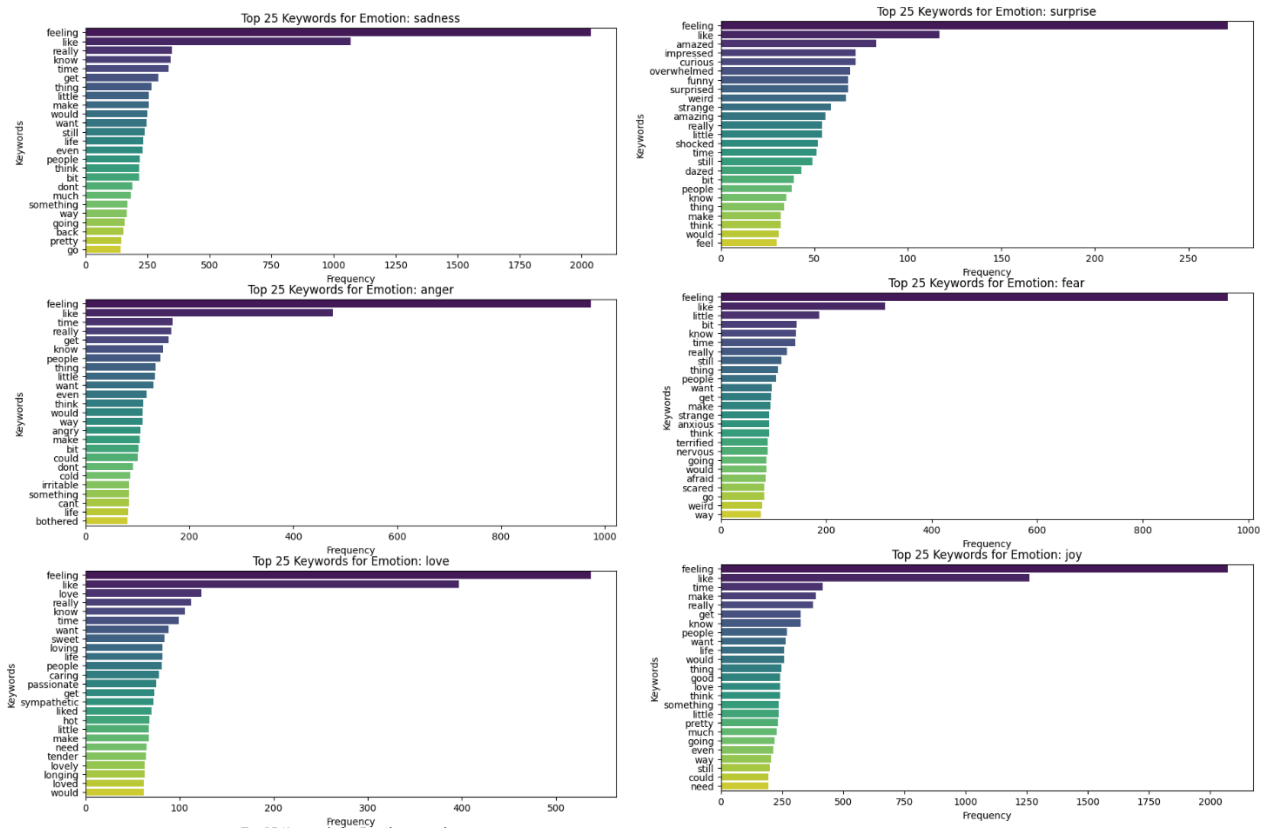

Рисунок 3.7 – Ключові слова для категорій емоцій

Візуалізація підтверджує, що ключові слова суттєво відрізняються між емоціями. Наприклад, для категорії «positive» часто зустрічаються слова, пов'язані з радістю та задоволенням («happy», «love», «good»), тоді як для

категорії «negative» характерні слова зі смисловим забарвленням, що вказує на тривогу або смуток («miss», «sorry», «bad», «hate»).

Завдяки використанню даних методів вдається значно зменшити розмірність даних, уникнувши зайвих термінів, і покращити якість підготовки до подальшої векторизації. Очищені та структуровані дані, отримані після цього етапу, готові до перетворення у числові представлення. Таким чином, підготовлений корпус текстів стає базою для навчання моделей машинного навчання, що забезпечує високу якість класифікації емоцій.

3.2.4 Векторизація текстів. Процес векторизації тексту полягає у перетворенні текстових даних у числові вектори, які можна використовувати для подальшого аналізу або навчання моделей машинного навчання. Цей етап є критично важливим для роботи з текстом у машинному навчанні, оскільки алгоритми працюють з числовими даними.

CountVectorizer дозволяє перетворити текст у вектор, де кожне унікальне слово з корпусу текстів представляється окремим стовпцем матриці. Рядки цієї матриці відповідають текстовим документам, а значення у клітинках – частоті появи слова у відповідному документі.

Розглянемо приклад із двох документів:

1. It was the dawn of hope.
2. It was the dusk of despair.
3. It is the age of uncertainty.

Словник або набір унікальних слів із цих текстів виглядає так:

['age', 'dawn', 'despair', 'dusk', 'hope', 'is', 'it', 'of', 'the', 'uncertainty', 'was']

Для кожного документа формується рядок матриці, де значення відповідають кількості появ слова у тексті, дана матриця використовується як базове представлення тексту для моделей машинного навчання (рис. 3.8).

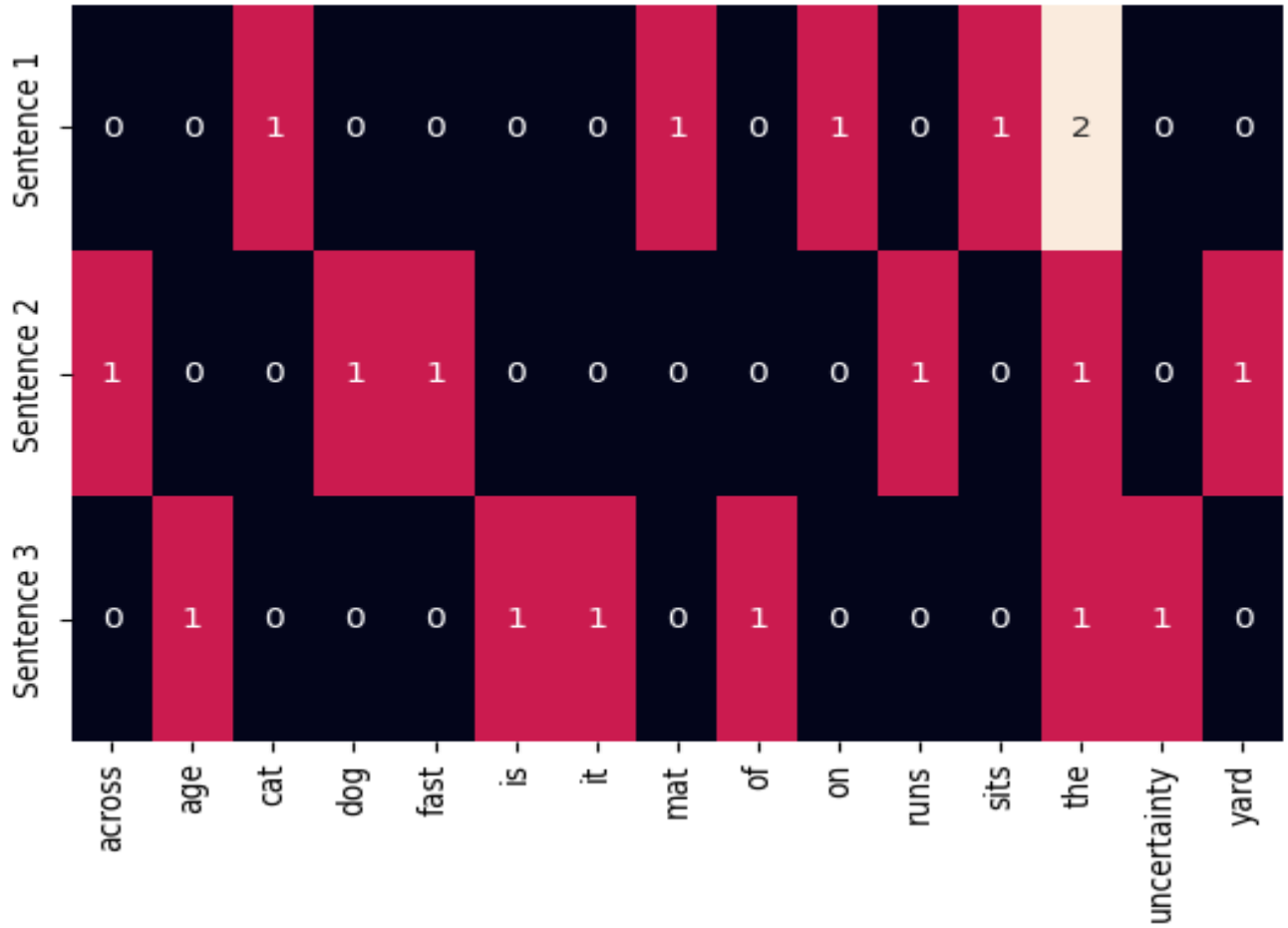

Рисунок 3.8 – Результат векторизації документу

TF-IDF (Term Frequency-Inverse Document Frequency) – метод, який зважує значення слів, враховуючи, наскільки часто вони з'являються у конкретному документі порівняно з усім корпусом текстів (формула 3.1).

Розглянемо приклад для тих самих документів:

1. It was the dawn of hope.
2. It was the dusk of despair.
3. It is the age of uncertainty.

$$TF_{t,d} = \frac{n_{t.d}}{\text{Загальна кількість слів у документі } d} \tag{3.1}$$

$n_{t.d}$ – кількість разів слово t з'являється в документі d.

Таким чином,кожен документ і слово матиме власне значення TF.

Розрахуємо значення TF для документу 2: It was the dusk of despair.

Значення TF для слова ‘it’ = (кількість разів появи слова ‘it’ в документі 2) / (кількість слів в документі 2) = 1/6.

Отже, таким чином для документу 2 маємо:

- TF(‘age’) = 0;
- TF(‘dawn’) = 0;
- TF(‘despair’) = 1/6;
- TF(‘dusk’) = 1/6;
- TF(‘hope’) = 0;
- TF(‘is’) = 0;
- TF(‘it’) = 1/6;
- TF(‘of’') = 1/6;
- TF(‘the’) = 1/6;
- TF(‘uncertainty’) = 0;
- TF(‘was’) = 1/6.

Таким чином розрахуємо всі TF для всіх слів усіх документів.

Таблиця 3.2 – Розрахункові TF значення документів

Слово	Док. 1	Док. 2	Док. 3	TF Док. 1	TF Док. 2	TF Док. 3
age	0	0	1	0	0	1/6
dawn	1	0	0	1/6	0	0
despair	0	1	0	0	1/6	0
dusk	0	1	0	0	1/6	0
hope	1	0	0	1/6	0	0
is	0	0	1	0	0	1/6
it	1	1	1	1/6	1/6	1/6
of	1	1	1	1/6	1/6	1/6
the	1	1	1	1/6	1/6	1/6
uncertainty	0	0	1	0	0	1/6
was	1	1	0	1/6	1/6	0

IDF є показником важливості слова, дане значення необхідне, оскільки обчислення лише TF недостатньо, щоб зрозуміти важливість слів (формула 3.2).

$$IDF_t = log \frac{\text{Загальна кількість документів}}{\text{Кількість документів зі словом } t} \quad (3.2)$$

Проблема з підрахуванням частоти слів полягає в тому, що дуже часто зустрічаються такі слова, як «is», «the», «a», які починають домінувати в документі, але не містять стільки «корисної інформації» для моделі у порівнянні з більш рідкісними, але специфічними словами.

Один із підходів полягає в тому, щоб змінити частоту слів на те, як часто вони з'являються в усіх документах, щоб оцінки для таких слів, як «the», які часто зустрічаються в усіх документах, були зменшені.

Цей підхід до оцінки називається частотою термінів – інверсною частотою документа або TF-IDF, де:

- частота термінів (TF): оцінка частоти слова в поточному документі;
- інверсна частота документа (IDF): наскільки рідкісним є слово.

Таким чином, IDF рідкісного слова є високим, тоді як IDF частого слова буде низьким.

Обрахуємо IDF значення для документу 2: It was the dusk of despair.

IDF('it') = log (загальна кількість документів) / (кількість документів, в яких присутнє слово 'it') = log(3/3) = log(1) = 0.

IDF('was') = log (загальна кількість документів) / (кількість документів, в яких присутнє слово 'was') = log(3/2) = 0.176.

Таким чином розрахуємо IDF для всіх слів у документах.

Таблиця 3.3 – Розрахункове IDF значення

Слово	Документ 1	Документ 2	Документ 3	IDF
age	0	0	1	0.477
dawn	1	0	0	0.477
despair	0	1	0	0.477
dusk	0	1	0	0.477
hope	1	0	0	0.477
is	0	0	1	0.477
it	1	1	1	0.0
of	1	1	1	0.0
the	1	1	1	0.0
uncertainty	0	0	1	0.477
was	1	1	0	0.176

Тепер можна обчислити оцінку TF-IDF для кожного слова в корпусі. Слова з вищим балом важливіші, а з нижчим – менш важливі (формула 3.3).

$$(TF - IDF)_{t,d} = TF_{t,d} * IDF_t \tag{3.3}$$

Обрахуємо TF-IDF значення для документу 2: It was the dusk of despair.

TF-IDF(‘it’, документ 2) = TF(‘it’, документ 2) * IDF(‘it’) = 1/6 * 0 = 0

Таким чином:

- TF-IDF (‘it’) = 1/6 * 0 = 0;
- TF-IDF (‘was’) = 1/6 * 0.176 = 0.029;
- TF-IDF (‘the’) = 1/6 * 0 = 0;
- TF-IDF (‘dusk’) = 1/6 * 0.477 = 0.795;
- TF-IDF (‘of’) = 1/6 * 0 = 0;
- TF-IDF (‘despair’) = 1/6 * 0.477 = 0.795.

Якщо слово є дуже поширеним і зустрічається у всіх документах, його IDF стає рівним нулю та в результаті TF-IDF оцінка для такого слова також буде дорівнювати нулю. Це означає, що занадто поширені терміни повністю «штрафуються» і не впливають на подальший аналіз.

Якщо термін зустрічається дуже рідко, можливо, лише в одному документі, його IDF набуває максимального значення, що підкреслює його важливість у контексті цього документа. Таким чином, TF-IDF допомагає виділити унікальні терміни, які мають значну семантичну вагу.

3.2.5 Навчання моделей машинного навчання. Для класифікації емоцій у текстах використовується кілька алгоритмів машинного навчання, зокрема Logistic Regression, SVM, Random Forest, Multinomial Naïve Bayes та Decision Tree. Кожна модель навчається на раніше підготовлених векторизованих даних. Вибір моделей базувався на їх здатності працювати з текстовими даними, забезпечувати високу точність класифікації та ефективно адаптуватися до специфіки задачі.

Процес навчання моделей включає поділ даних на навчальну (70%) та тестову (30%) вибірки для оцінювання ефективності. Моделі налаштовуються з використанням базових гіперпараметрів, а їх продуктивність оцінюється метриками точності (accuracy), повноти (recall), F1-міри та матриці плутанини. Для покращення точності моделі використовуються сучасні методи оптимізації, такі як стохастичний градієнтний спуск (SGD) у випадку SVM.

Для забезпечення зручності використання всі алгоритми інтегруються у пайплайн із використанням бібліотеки Pipeline з пакету scikit-learn. Це дозволяє об'єднати всі етапи обробки тексту, векторизації та класифікації у єдиний процес. Застосування пайплайна забезпечує модульність і масштабованість системи, а також спрощує її використання на нових даних або при зміні моделей.

Моделі серіалізуються з використанням бібліотеки dill для забезпечення їх повторного використання. Це дозволяє зберігати не лише параметри моделі, але й усю конфігурацію пайплайна, включаючи процеси обробки текстів і векторизації. Збережені моделі можуть бути легко завантажені для використання у виробничих середовищах або для додаткового навчання.

Для аналізу роботи моделей і перевірки якості їхньої класифікації створюються графічні представлення. Зокрема, матриці плутанини, графіки

розподілу точності, повноти та F1-міри для кожної моделі. Ці візуалізації надають можливість детально оцінити, як моделі справляються з класифікацією кожної емоційної категорії, та ідентифікувати області для подальшого покращення (рис. 3.9).

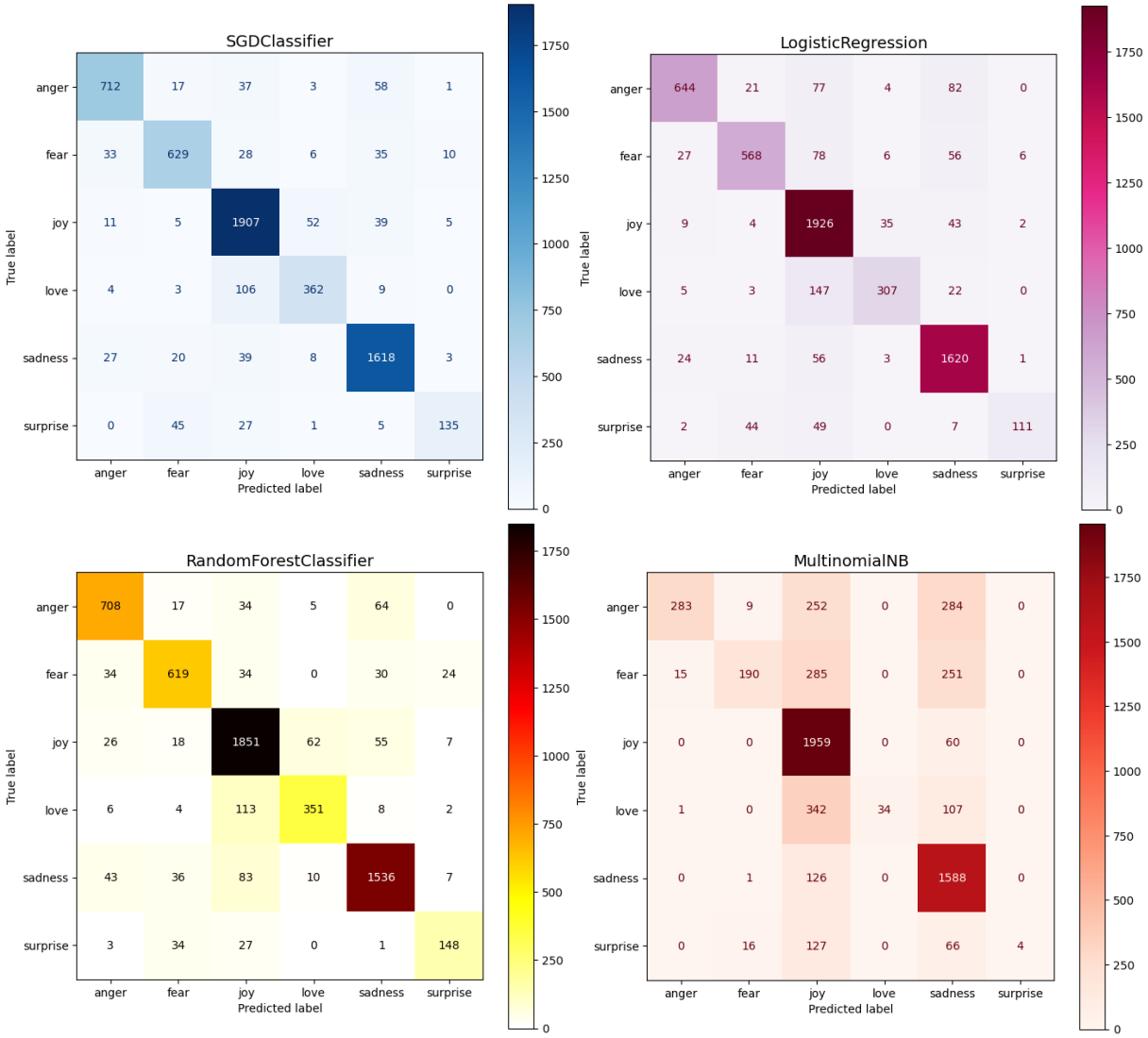

Рисунок 3.9 – Результат навчання моделей (матриці плутанини)

Аналіз матриць плутанини для різних моделей дозволяє виявити певні закономірності та особливості їхньої роботи з даними. Усі моделі демонструють найкращу продуктивність для класифікації класу joy, що може свідчити про його домінування в навчальній вибірці або про наявність чітко виражених характеристик цього класу. Ця тенденція показує, що моделі добре розпізнають

емоційний контекст, який найчастіше зустрічається або найкраще представлений у даних.

Водночас, для інших емоцій, таких як anger, fear, love, sadness і особливо surprise, результати значно гірші. Це може бути зумовлено декількома факторами. По-перше, дисбаланс у кількості прикладів для кожної категорії у вибірці негативно впливає на моделі, адже менш представлені класи отримують меншу увагу під час навчання. По-друге, можлива схожість між деякими емоціями на рівні тексту створює додаткові труднощі. Наприклад, емоції anger та sadness часто плутаються, що може бути результатом спільних лексичних чи синтаксичних патернів у текстах.

Реалізовані алгоритми розпізнавання емоцій забезпечують достатньо високу точність. Завдяки модульній структурі розробки система легко адаптується до змін у даних або вимогах задачі, що дозволяє її інтегрувати у різні прикладні додатки.

3.3 Реалізація додатку для розпізнавання емоцій

На основі створеної та навченої моделі реалізовано додаток для розпізнавання емоцій у тексті. Додаток створено за допомогою бібліотеки Streamlit, яка дозволяє розробляти інтерактивні веб-додатки для аналізу даних і машинного навчання. Основне завдання додатку – надавати користувачам інтерфейс для введення тексту, аналізувати його емоційне забарвлення та наочно відображати результати.

Головна мета цього додатку – забезпечити інтуїтивно зрозумілий спосіб використання моделі розпізнавання емоцій у реальному часі. Він може бути корисним для широкого спектра застосувань, включаючи аналіз текстів у соціальних мережах, оцінку настроїв клієнтів у відгуках або коментарях та інші задачі аналізу текстових даних.

Додаток дозволяє:

- ввести текст для аналізу емоційного забарвлення;
- отримати передбачену моделлю емоцію для введеного тексту;
- побачити ймовірності належності тексту до кожної емоції;
- наочно проаналізувати результати у вигляді графіка.

Головний інтерфейс додатку виглядає наступним чином (рис. 3.10):

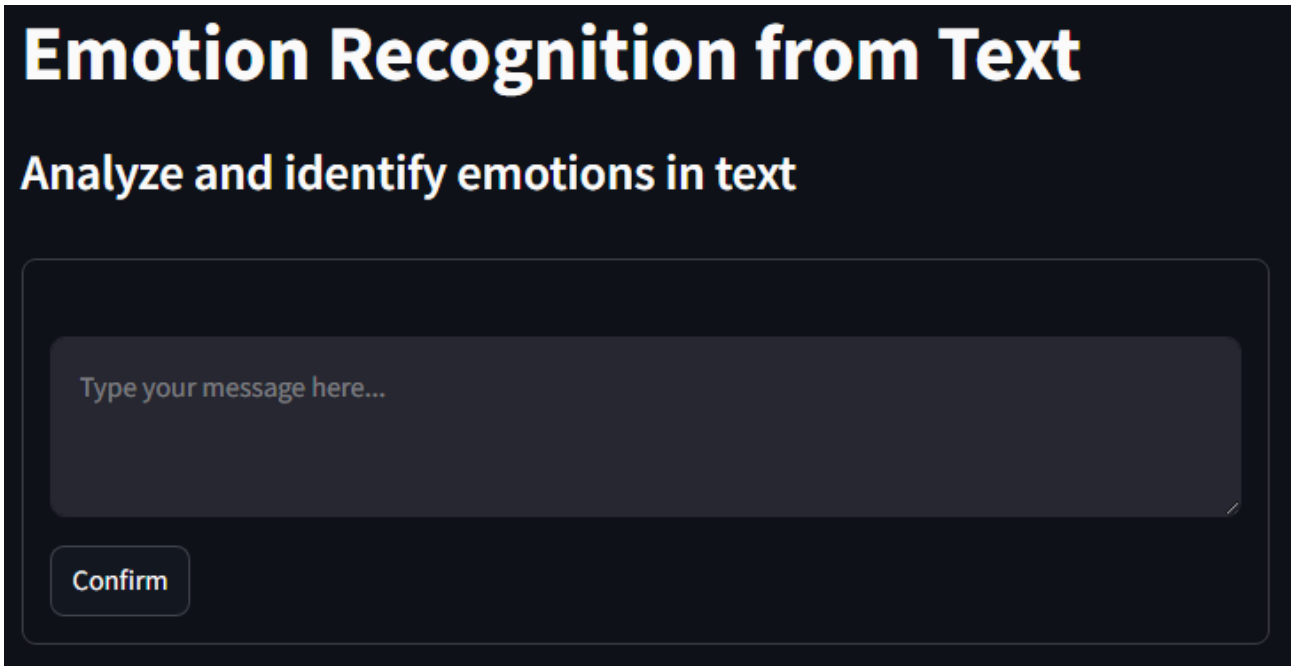

Рисунок 3.10 – Головний інтерфейс додатку

Після введення тексту та натискання кнопки підтвердження додаток виконує обробку введеного тексту за допомогою попередньо навченої моделі. На екрані з'являється результат аналізу, що включає передбачену емоцію та відповідну ймовірність.

Крім того, виводиться розподіл ймовірностей для кожної з можливих емоцій у вигляді гістограми, що дозволяє користувачеві не лише дізнатися основний емоційний стан тексту, але й побачити ступінь впевненості моделі в своєму передбаченні та оцінити альтернативні варіанти класифікації. Такий підхід забезпечує зручність для розуміння та прозорість результатів.

На прикладі тексту «I was able to help chai lifeline with your support and encouragement is a great feeling and i am so glad you were able to help me» додаток передбачив емоцію «joy» з ймовірністю 0.6167 (рис. 3.11).

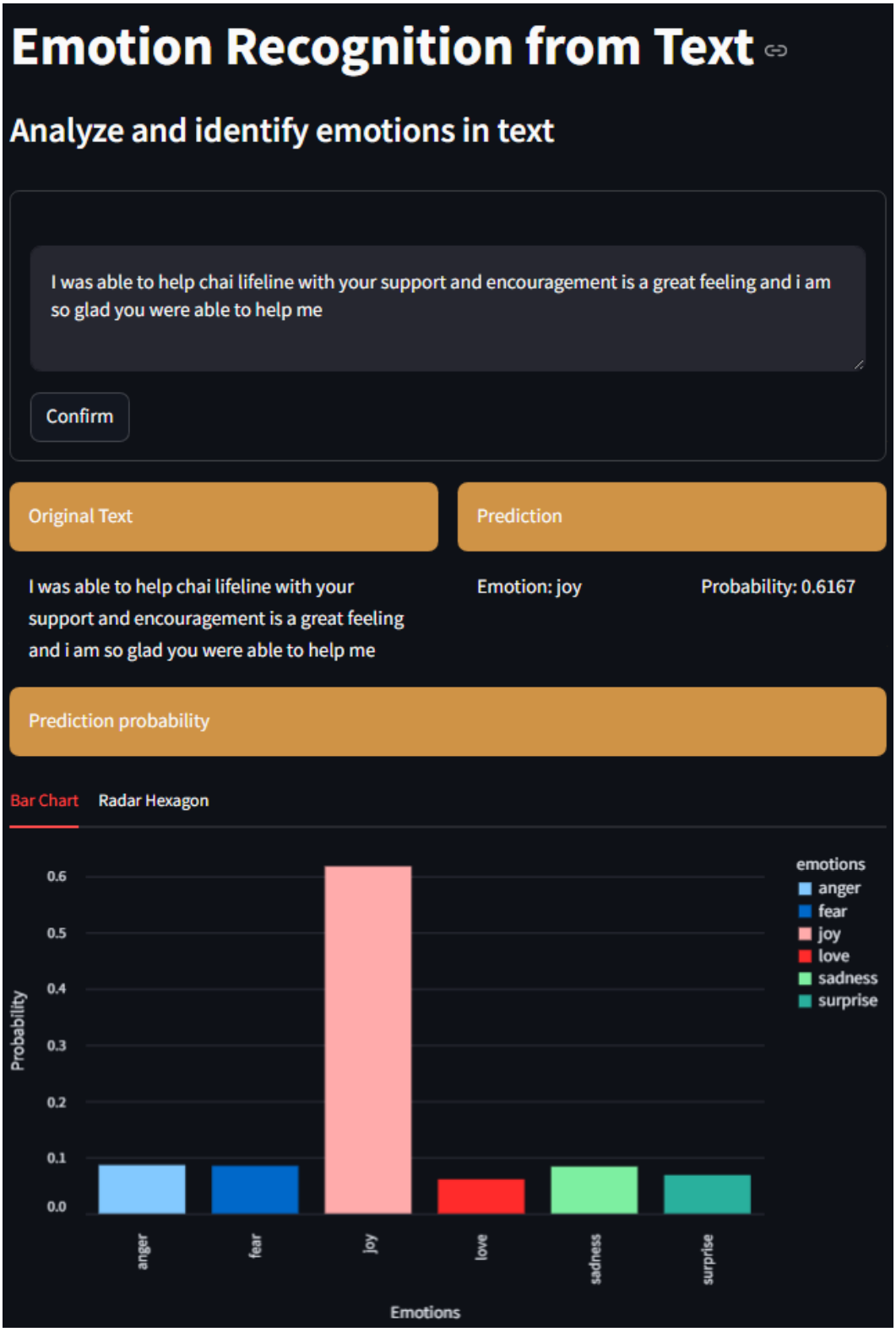

Рисунок 3.11 – Результат визначення емоційного забарвлення тексту додатком

Також, додаток підтримує відображення іншого типу діаграми – радарної (шестикутної) діаграми. Такий підхід надає більше можливостей для візуального аналізу даних, адже кожен тип графічного представлення має свої переваги. Стовпчикова діаграма дозволяє чітко порівнювати ймовірності для кожної емоції, надаючи зрозуміле відображення величин, тоді як радарна діаграма показує розподіл ймовірностей у більш компактній формі, акцентуючи на відмінностях між категоріями (рисю. 3.12).

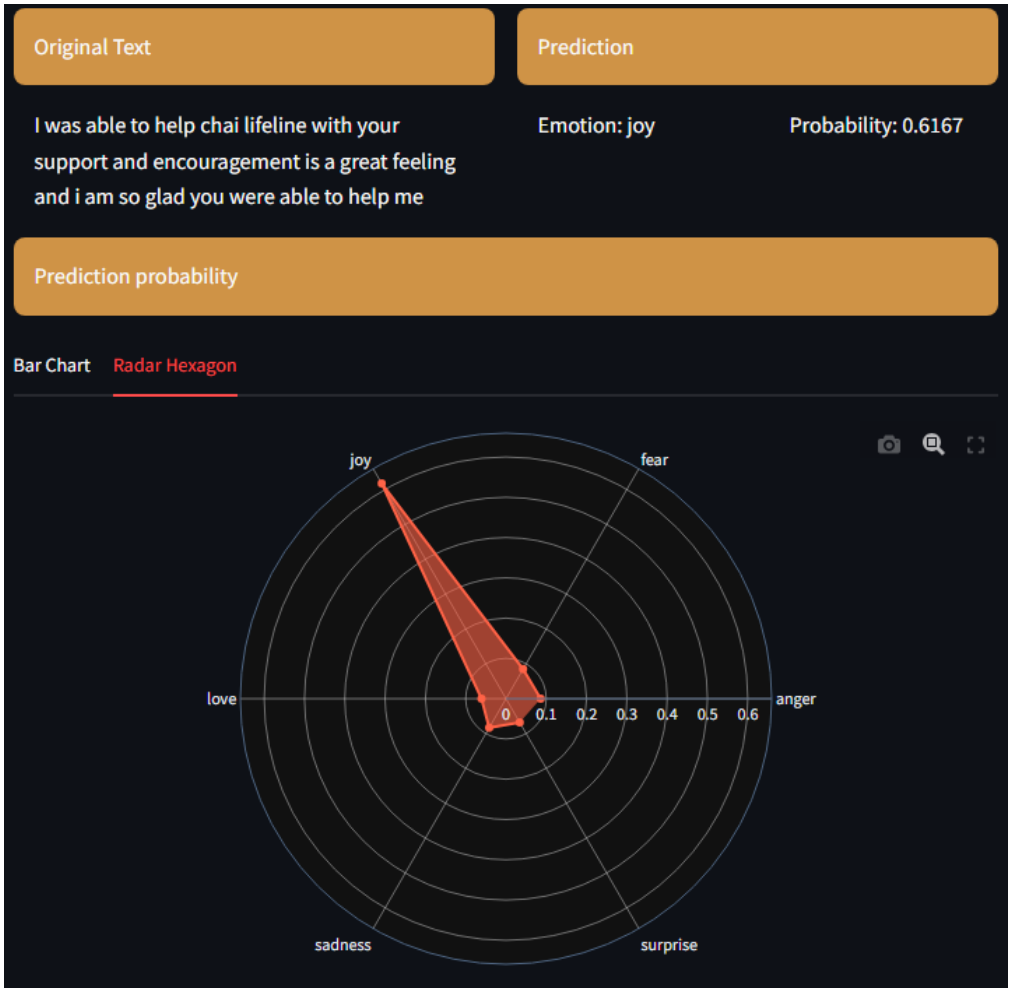

Рисунок 3.12 – Радарна (шестикутна) діаграма

Додаток є ефективним інструментом для взаємодії з моделлю розпізнавання емоцій. Завдяки використанню бібліотеки Streamlit вдалося створити інтерактивний, зручний у використанні додаток із високою швидкістю

роботи. Він легко інтегрується в різні бізнес-процеси або дослідницькі проекти, що робить його корисним для аналізу текстових даних у реальному часі.

У майбутньому можливе розширення функціональності додатку за рахунок:

- додавання підтримки багатомовного аналізу тексту;
- інтеграції з платформами соціальних мереж для аналізу текстів;
- покращення інтерфейсу для аналізу великих обсягів даних.

3.4 Аналіз отриманих результатів розпізнавання емоцій

Аналіз отриманих результатів показує відмінності у роботі різних моделей машинного навчання для розпізнавання емоційного стану людини. Зокрема, такі показники, як точність (Precision) і повнота (Recall) демонструють, як моделі справляються з передбаченням різних емоцій, таких як anger, fear, joy, love, sadness та surprise, а загальна точність (Accuracy) демонструє якість моделі в цілому.

Для загальної оцінки якості моделі використовується показник загальної точності (accuracy), який показує, яку частку всіх передбачень було зроблено правильно (рис. 3.13). Він розраховується як відношення кількості правильно передбачених об'єктів до загальної кількості об'єктів у тестовій вибірці (формула 3.4).

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}, \tag{3.4}$$

- де TP (True Positive) – правильно класифіковані позитивні випадки;
- TN (True Negative) – правильно класифіковані негативних випадки;
- FP (False Positive) – кількість помилкових позитивних передбачень;
- FN (False Negative) – кількість помилкових негативних передбачень.

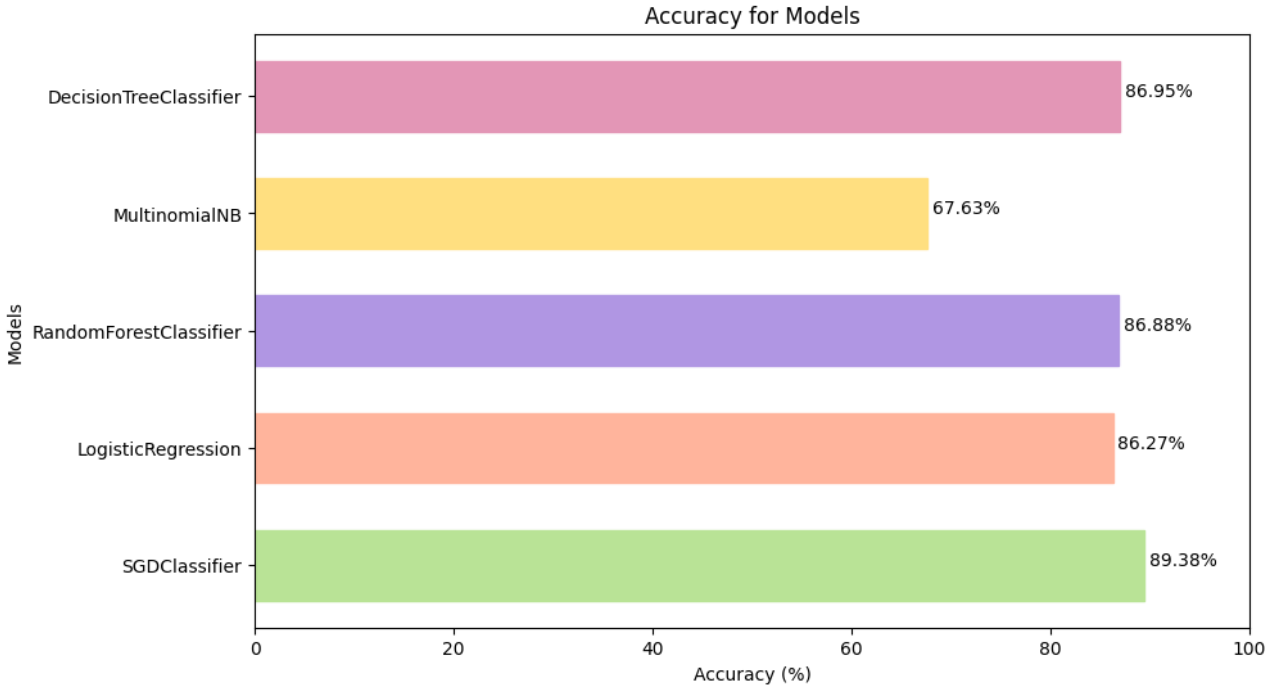

Рисунок 3.13 – Загальна точність моделей

Даний показник дає уявлення про загальну продуктивність моделі, однак у випадках із несбалансованими даними, використання лише accuracy може бути недостатнім. У таких ситуаціях важливо також аналізувати показники Precision і Recall для кожної категорії окремо.

Точність (Precision) показує, яка частка об'єктів, які алгоритм передбачив як належні до певного класу, дійсно належить до цього класу (формула 3.5). Вона характеризує здатність алгоритму уникати хибнопозитивних передбачень (рис. 3.14).

$$Precision = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Positives\ (FP)} \quad (3.5)$$

	SGDClassifier	LogisticRegression	RandomForestClassifier	MultinomialNB	DecisionTreeClassifier
anger	90.482	90.577	86.341	94.649	89.158
fear	87.065	87.250	85.027	87.963	82.111
joy	88.899	82.555	86.415	63.378	91.262
love	83.371	86.479	82.009	100.000	75.908
sadness	92.036	88.525	90.673	67.402	88.259
surprise	88.158	92.500	78.723	100.000	73.973

Рисунок 3.14 – Результат точності класифікації моделей

Повнота відображає, яка частка всіх об'єктів певного класу була правильно визначена алгоритмом (формула 3.6). Вона характеризує здатність алгоритму знаходити всі об'єкти, що належать до цього класу, і уникати хибнонегативних передбачень (рис. 3.15).

$$Recall = \frac{True\ Positives\ (TP)}{True\ Positives\ (TP) + False\ Negatives\ (FN)} \tag{3.6}$$

	SGDClassifier	LogisticRegression	RandomForestClassifier	MultinomialNB	DecisionTreeClassifier
anger	86.111	77.778	85.507	34.179	84.420
fear	84.480	76.653	83.536	25.641	86.100
joy	94.403	95.394	91.679	97.028	87.420
love	75.620	63.430	72.521	7.025	82.025
sadness	94.344	94.461	89.563	92.595	90.729
surprise	62.911	52.113	69.484	1.878	76.056

Рисунок 3.15 – Результат повноти класифікації моделей

Модель SGDClassifier виявилася однією з найбільш збалансованих, вона демонструє високу точність і повноту для більшості емоцій. Наприклад, точність для sadness досягла 92.036%, а для joy – 88.899%. Повнота також показала сильні результати: 94.344% для sadness і 94.403% для joy. Проте спостерігаються певні зниження для менш поширених емоцій, таких як surprise (62.911% повноти).

LogisticRegression продемонструвала стабільні результати для емоцій з високою частотою, таких як joy і sadness. Наприклад, для joy повнота становить 95.394%, а для sadness – 94.461%. Однак ця модель має труднощі із передбаченням емоцій з меншою кількістю даних, таких як love і surprise. Її точність для love становить 86.479%, тоді як повнота для цієї ж емоції – лише 63.430%.

RandomForestClassifier демонструє хорошу продуктивність для основних емоцій, наприклад, joy (86.415% точності) і sadness (90.673% точності). Однак результати для love і surprise залишаються відносно слабшими, що вказує на обмежену здатність моделі працювати з менш поширеними емоціями.

Модель MultinomialNB виділяється своєю високою точністю для емоцій love (100.000%) та surprise (100.000%), але значно поступається в повноті. Наприклад, повнота для love становить лише 7.025%, а для surprise – 1.878%. Це свідчить про те, що модель має тенденцію до високої специфічності для певних класів, але втрачає здатність ефективно ідентифікувати всі приклади цих емоцій.

DecisionTreeClassifier демонструє різну продуктивність залежно від емоцій. Наприклад, вона показує високу точність для joy (91.262%), але знижує свої показники для love (75.908%) і surprise (73.973%). Повнота також залишається помірною, найкраще модель справляється з sadness (90.729%) та fear (86.100%).

Отже, результати аналізу свідчать, що кожен алгоритм має свої переваги та обмеження. SGDClassifier є найкращим вибором для збалансованої роботи з усіма емоціями, особливо якщо важливі точність і повнота для більшості категорій. LogisticRegression є ефективною для більш розповсюджених емоцій, але має труднощі з менш поширеними. RandomForestClassifier забезпечує стабільну продуктивність для основних категорій, проте потребує оптимізації для специфічних емоцій. MultinomialNB підходить для задач, де важлива максимальна точність для окремих емоцій, але потребує доопрацювання через низьку повноту. DecisionTreeClassifier є більш варіативною в своїй

продуктивності, і її доцільно використовувати у випадках, де потрібна швидка інтерпретація результатів або мала вибірка.

Загалом, вибір алгоритму залежить від конкретних вимог задачі. Якщо потрібен баланс між точністю та повнотою, доцільно використовувати SGDClassifier. У випадках, де критично важливе передбачення специфічних емоцій, можуть бути використані MultinomialNB або RandomForestClassifier.

У даній роботі для перевірки функціонування додатку було обрано модель SGDClassifier, оскільки вона продемонструвала високу точність та збалансовані показники точності і повноти у більшості категорій емоцій. Отже, дана модель інтегрується у додаток, оскільки вона забезпечує надійний компроміс між ефективністю роботи та продуктивністю, а її використання дозволяє перевірити коректність реалізації інтерфейсу, візуалізації даних і точності класифікації емоцій, що є критично важливим для тестування та демонстрації роботи додатку.

Для перевірки роботи додатку вводяться тексти, що відображають різні емоційні стани, аби оцінити точність передбачень моделі. Наприклад, випадку з текстом «i know the pain parents feel when an enraged child becomes violent» модель передбачила емоцію «гнів», що відповідає очікуваному результату (рис. 3.16).

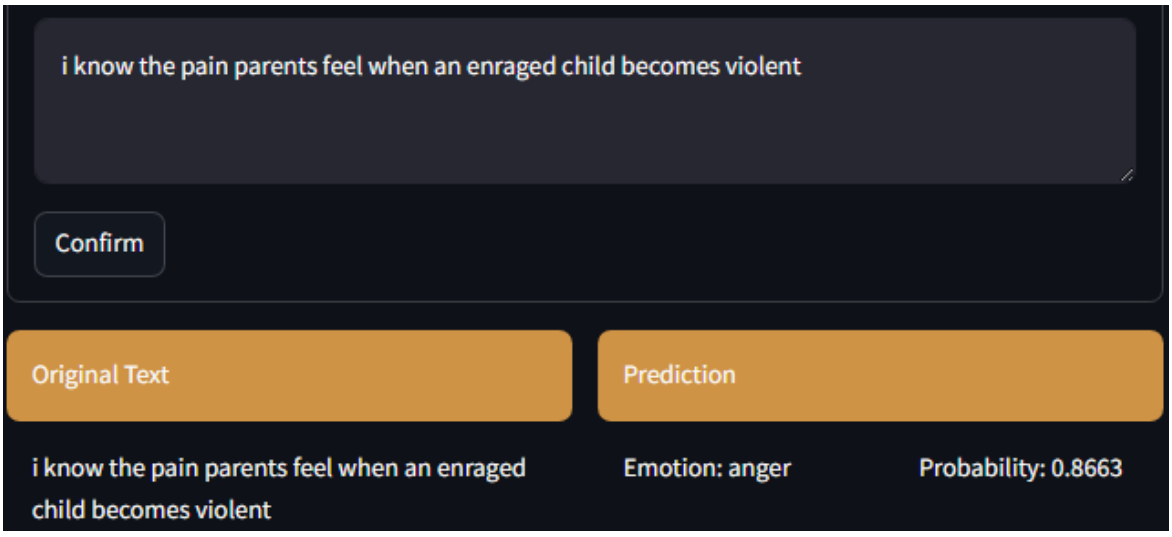

Рисунок 3.16 – Результат перевірки емоції гніву

Для категорії страху використаний текст «i will be able to lay on my bed in the dark and not feel terrified at least for a while». Модель продемонструвала правильну ідентифікацію цієї емоції. Висока ймовірність прогнозу свідчить про ефективність алгоритму в обробці емоцій тривоги та невизначеності (рис. 3.17).

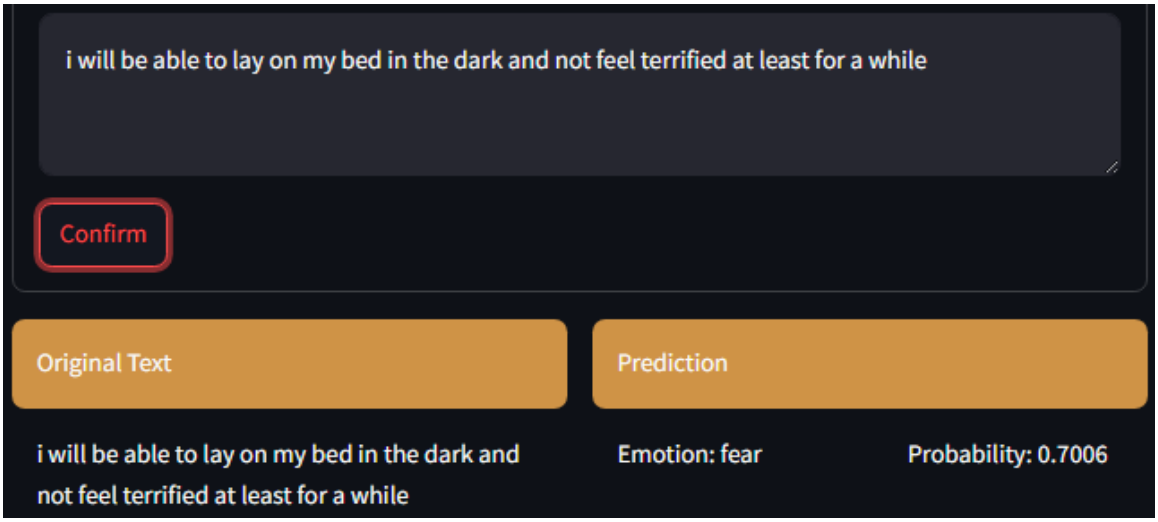

Рисунок 3.17 – Результат перевірки емоції страху

Тестування категорії «love» проводиться за допомогою тексту «i was ready to meet mom in the airport and feel her ever supportive arms around me». Модель успішно визначила емоцію любові, продемонструвавши здатність фіксувати глибокі, позитивні почуття прихильності (рис. 3.18).

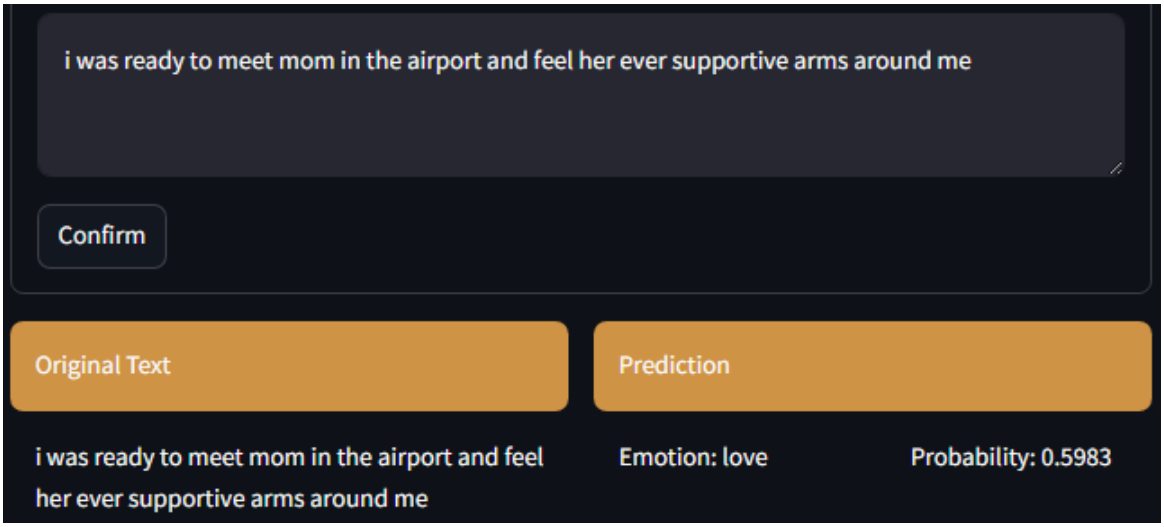

Рисунок 3.18 – Результат перевірки емоції любові

У тексті «i still feel sleep deprived she is almost sleeping through the night giving us» модель точно визначила емоцію «sadness». Високий рівень відповідності між введеним текстом і прогнозом підтвердив надійність класифікації для негативних емоцій (рис. 3.19).

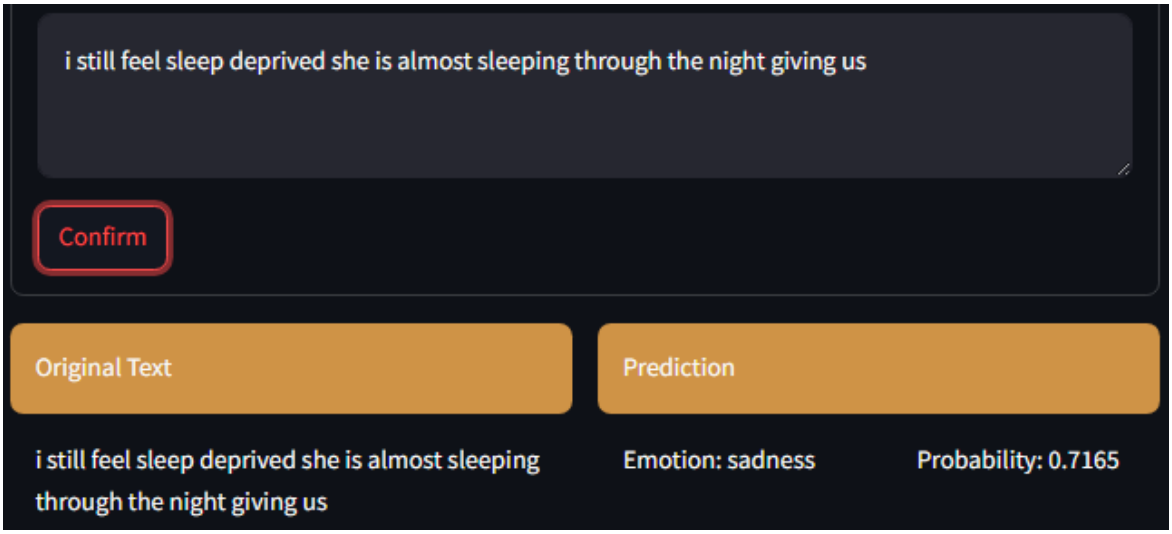

Рисунок 3.19 – Результат перевірки емоції смутку

Для категорії «surprise» використовується текст «i still feel a little dazed and have that sort of disbelieving feeling of oh my god». Модель успішно розпізнала емоцію здивування, що демонструє її здатність враховувати контекст і раптовий характер подій (рис. 3.20). Більше прикладів можна переглянути в додатку Б.

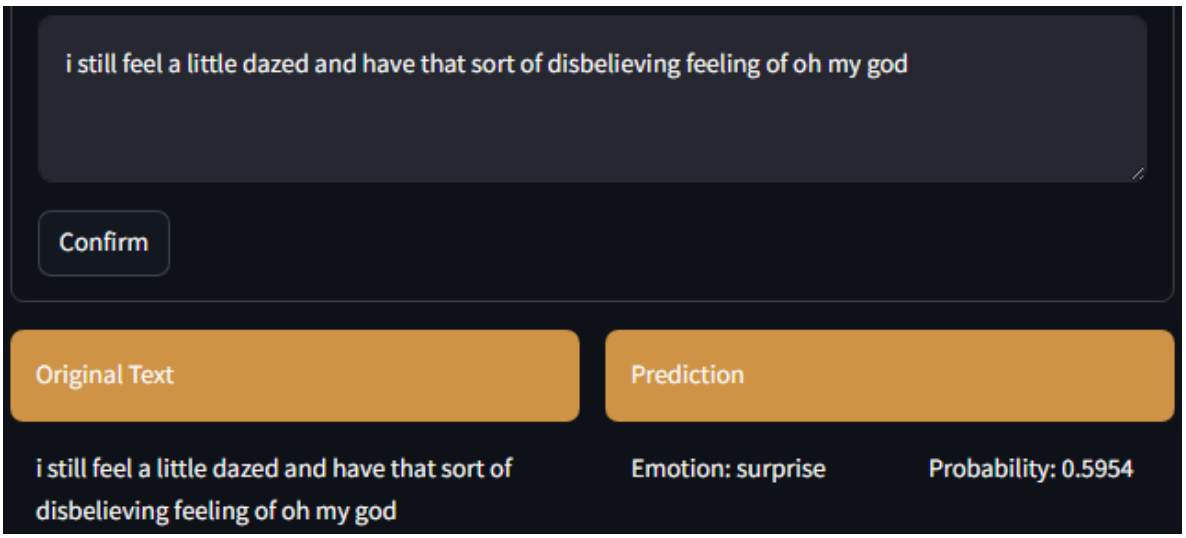

Рисунок 3.20 – Результат перевірки емоції здивування

Наведені приклади роботи програми демонструють її здатність ефективно класифікувати тексти за емоційними категоріями, підтверджуючи точність і надійність створеної системи. Завдяки використанню сучасних алгоритмів машинного навчання та інтерактивного інтерфейсу, програма забезпечує не лише високу якість класифікації, але й зручність для кінцевого користувача. Це свідчить про готовність системи до впровадження у прикладні задачі, такі як аналіз соціальних мереж, підтримка клієнтів або автоматизація обробки текстових даних в інших галузях.

3.5 Висновки до Розділу 3

У третьому розділі було реалізовано та проаналізовано результати роботи системи для автоматизованого розпізнавання емоцій у текстових даних. Основний акцент зроблено на інтеграції підготовлених моделей машинного навчання у практичну систему, перевірці їхньої ефективності та створенні зручного інтерфейсу для користувачів.

Результати класифікації показали, що обрані моделі забезпечують високу точність розпізнавання емоцій у текстах. Серед моделей найкращі результати продемонстрував SGDClassifier, який було обрано як основну модель для інтеграції у додаток. Аналіз матриць плутанини, точності та повноти дозволив оцінити сильні та слабкі сторони кожної моделі, що сприяло обґрунтованому вибору найбільш ефективного рішення.

Для демонстрації роботи системи було створено інтерактивний додаток за допомогою бібліотеки Streamlit. Він забезпечує користувачам можливість вводити текстові дані, отримувати передбачення емоційної категорії та аналізувати розподіл ймовірностей для кожної емоції. Додатково реалізована можливість вибору між різними типами діаграм для візуалізації результатів, що підвищує зручність аналізу та інтерпретації даних.

Експериментальні результати свідчать про те, що створена система є надійною та ефективною. Вона дозволяє аналізувати тексти з високою точністю,

зберігаючи при цьому масштабованість та адаптивність до різних типів текстових даних. Впровадження інтерактивного інтерфейсу забезпечує простоту використання, що робить систему доступною для широкого кола користувачів, включаючи фахівців із соціальних досліджень, маркетологів та розробників чат-ботів.

Таким чином, у розділі було досягнуто мети інтеграції розробленої моделі в прикладне середовище та забезпечено її ефективність як аналітичного інструменту для автоматизованого розпізнавання емоцій у текстових даних.

ВИСНОВКИ

У роботі виконано повний цикл створення системи для автоматизованого розпізнавання емоцій у текстових даних, починаючи з теоретичного обґрунтування і закінчуючи практичною реалізацією інтерактивного додатку.

У першому розділі розглянуто теоретичні основи розпізнавання емоційного стану людини. Проведено аналіз емоційних категорій та їх відображення у текстових даних, описано ключові алгоритми машинного навчання, що використовуються для розв’язання задач класифікації текстів за емоційними категоріями. Особливу увагу приділено специфіці текстових даних із соціальних мереж, зокрема їх лаконічності, використанню сленгу, емодзі та інших невербальних елементів, які ускладнюють автоматичний аналіз.

Другий розділ присвячено методології створення системи. У ньому детально описано процес підготовки текстових даних, включаючи їх очищення, токенізацію, видалення стоп-слів та лематизацію. Обґрунтовано вибір методів векторизації тексту, що забезпечують адекватне представлення текстів у числовій формі. Розглянуто підхід до вибору моделей машинного навчання, ефективність яких оцінювалася на основі метрик точності, повноти, F1-міри та аналізу матриць плутанини.

У третьому розділі реалізовано інтеграцію підготовлених моделей у практичну систему, що включає створення веб-додатку для аналізу текстів. Додаток надає можливість введення текстів, автоматичного визначення емоційної категорії та візуалізації результатів у вигляді діаграм. На основі проведених експериментів було встановлено, що модель SGDClassifier забезпечує найкращий баланс між точністю та швидкодією, що стало ключовим фактором для її використання в додатку.

Результати роботи свідчать, що розроблена система є ефективним інструментом для аналізу текстових даних із соціальних мереж. Вона дозволяє швидко та точно визначати емоційне забарвлення тексту. Інтерактивний додаток

забезпечує зручність використання, дозволяючи в реальному часі отримувати результати класифікації та аналізувати їх.

Створена система демонструє високу гнучкість і масштабованість, що дозволяє адаптувати її для вирішення нових задач. Вона також може бути інтегрована в різні виробничі середовища, забезпечуючи цінну аналітичну підтримку для бізнесу та наукових досліджень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бєгунов А.Б. Алгоритм автоматизованого визначення емоційного забарвлення текстових даних / А.Б. Бєгунов. // Інженерія програмного забезпечення. – 2011. – №2(6). – С. 93–97.

2. Бабіч О.М., Замаруєва І.В., Шелег М.Г. Метод оцінювання властивостей тексту в ході проведення аналітичного дослідження з обробки даних / О.М. Бабіч, І.В. Замаруєва, М.Г. Шелег. // Актуальні проблеми автоматизації та інформаційних технологій. – 2021. – Том 25. – С. 13–22.

3. Supervised and Unsupervised learning. [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – 2023. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/supervised-unsupervised-learning/.

4. Що таке токенізація в НЛП? [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://www.oksim.ua/2023/07/25/shho-take-tokenizacziya-v-nlp/.

5. Що таке стемінг і лемматизація? [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://www.guru99.com/uk/stemming-lemmatization-python-nltk.html.

6. Vectorization Techniques in NLP. [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/vectorization-techniques-in-nlp/.

7. Demystify TF-IDF in Indexing and Ranking. [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://ted-mei.medium.com/demystify-tf-idf-in-indexing-and-ranking-5c3ae88c3fa0.

8. What is a word embedding? [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://serokell.io/blog/word2vec.

9. Introduction to GloVe Embeddings. [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://sainivedh.medium.com/introduction-to-glove-embeddings-9f57d48d0ce4.

10. BERT language model. [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://www.techtarget.com/searchenterpriseai/definition/BERT-language-model.

11. Введення в наївний алгоритм Байєса. [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://codelabsacademy.com/uk/blog/naive-bayes.

12. Logistic Regression Overview. [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://medium.com/@Coursesteach/natural-language-processing-part-9-13690a56d5bb.

13. Що таке дерево рішень? [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://www.unite.ai/uk/what-is-a-decision-tree/.

14. Decision Trees, Random Forests, and Gradient Boosting: What's the Difference? [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://towardsdatascience.com/decision-trees-random-forests-and-gradient-boosting-whats-the-difference-ae435cbb67ad.

15. Що таке RNN і LSTM у Deep Learning? [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://www.unite.ai/uk/what-are-rnns-and-lstms-in-deep-learning/.

16. Oversampling—Handling Imbalanced Data. [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://medium.com/@abdallahashraf90x/oversampling-for-better-machine-learning-with-imbalanced-data-68f9b5ac2696.

17. What Is Undersampling? [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://www.mastersindatascience.org/learning/statistics-data-science/undersampling/.

18. Text augmentation techniques in NLP. [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу до ресурсу: https://www.geeksforgeeks.org/text-augmentation-techniques-in-nlp/.

ДОДАТОК А
ВИХІДНИЙ КОД ОСНОВНИХ КОМПОНЕНТІВ ПРОГАМИ

1. Навчання моделей

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

import nltk
import string
import re

from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.model_selection import train_test_split
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
import xgboost as xgb
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import MultinomialNB
from sklearn import tree

from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import f1_score
from sklearn.metrics import classification_report
from sklearn.metrics import ConfusionMatrixDisplay

df = pd.read_csv('text_emotions.csv')
df.columns = ['Text', 'Emotion']

print('Dataset size:',df.shape)
print('Columns are:',df.columns)
df.isnull().sum()
df['Emotion'].value_counts()

def plot_emotion_distribution(df, col='Emotion'):
    value_counts = df[col].dropna().value_counts().nlargest(10)
    sizes = value_counts.values
    labels = value_counts.index

    fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, figsize=(12, 8))
```

```
    sns.countplot(y=col, data=df, ax=ax1)

    ax2.pie(sizes, explode=[0.1] * len(sizes), startangle=60, labels=labels,
autopct='%1.0f%%', pctdistance=0.9)

    ax1.set_title("Count of each emotion")
    ax2.set_title("Percentage of each emotion")
    plt.show()

plot_emotion_distribution(df)

df_stats = df.copy()
df_stats['char_length'] = df['Text'].apply(lambda x : len(x))
df_stats['token_length'] = df['Text'].apply(lambda x : len(x.split(" ")))

fig, (ax1, ax2)  = plt.subplots(nrows=1, ncols=2, figsize=(12,6))
sns.distplot(df_stats['char_length'], ax=ax1)
sns.distplot(df_stats['token_length'], ax=ax2)
ax1.set_title('Number of characters in the tweet')
ax2.set_title('Number of token(words) in the tweet')
plt.show()


fig2, (ax1, ax2)  = plt.subplots(nrows=1, ncols=2, figsize=(12,6))
for emotion in df_stats['Emotion'].value_counts().nlargest(5).index.tolist():
    sns.kdeplot(df_stats[df_stats['Emotion']==emotion]['char_length'],ax=ax1,
label=emotion)
    sns.kdeplot(df_stats[df_stats['Emotion']==emotion]['token_length'],ax=ax2,
label=emotion)
ax1.legend()
ax2.legend()
ax1.set_title("Distribution of character length emotion-wise")
ax2.set_title("Distribution of token length sentiment-wise")
plt.show()

from sklearn.base import BaseEstimator, TransformerMixin

class TextPreprocessor(BaseEstimator, TransformerMixin):
    def __init__(self):
        pass

    def fit(self, X, y=None):
        return self

    def transform(self, X):
        # Обробка текстів у списку
        return [" ".join(self.clean_text(doc)) for doc in X]

    def clean_text(self, text):
        text = self.remove_punct(text)
```

```
        text = self.tokenization(text)
        text = self.remove_stopwords(text)
        text = self.lemmatizer(text)
        return text  # Повертаємо список токенів

    def remove_punct(self, text):
        text = "".join([char for char in text if char not in string.punctuation])
        text = re.sub('[0-9]+', '', text)
        return text

    def tokenization(self, text):
        text = text.lower()
        text = re.split('\W+', text)
        return text

    def remove_stopwords(self, text):
        stopword = nltk.corpus.stopwords.words('english')
        stopword.extend(['im','yr', 'year', 'woman', 'man', 'girl','boy','one',
'two', 'sixteen', 'yearold', 'fu', 'weeks', 'week',
              'treatment', 'associated', 'patients', 'may','day',
'case','old','u','n','didnt','ive','ate','feel','keep'
                ,'brother','dad','basic'])
        text = [word for word in text if word not in stopword]
        return text

    def lemmatizer(self, text):
        wn = nltk.WordNetLemmatizer()
        text = [wn.lemmatize(word) for word in text]
        return text


preprocessor = TextPreprocessor()
wn = nltk.WordNetLemmatizer()

df['Tweet_punct'] = df['Text'].apply(lambda x: preprocessor.remove_punct(x))
df['Tweet_tokenized'] = df['Tweet_punct'].apply(lambda x:
preprocessor.tokenization(x.lower()))
df['Tweet_nonstop'] = df['Tweet_tokenized'].apply(lambda x:
preprocessor.remove_stopwords(x))
df['Tweet_lemmatized'] = df['Tweet_nonstop'].apply(lambda x:
preprocessor.lemmatizer(x))


from collections import Counter
def extract_keywords(text,num=30):
    tokens = [token for token in text.split()]
    most_common_tokens = Counter(tokens).most_common(num)
    return dict(most_common_tokens)

def extract_keywords_for_emotions(df, num=30):
```

```
    keywords_by_emotion = {}

    for emotion in df['Emotion'].unique():
        emotion_list = df[df['Emotion'] == emotion]['Tweet_lemmatized']
        emotion_docx = ' '.join(emotion_list.apply(lambda x: ' '.join(x) if
isinstance(x, list) else str(x)).tolist())
        keywords_by_emotion[emotion] = extract_keywords(emotion_docx, num)
    return keywords_by_emotion


keywords_by_emotion = extract_keywords_for_emotions(df, num=30)


def plot_top_keywords_for_emotions(keywords_by_emotion, top_n=25):
    fig, axes = plt.subplots(len(keywords_by_emotion), 1, figsize=(10, 5 *
len(keywords_by_emotion)))

    for ax, (emotion, keywords) in zip(axes, keywords_by_emotion.items()):
        top_keywords = dict(sorted(keywords.items(), key=lambda item: item[1],
reverse=True)[:top_n])

        words = list(top_keywords.keys())
        counts = list(top_keywords.values())

        sns.barplot(x=counts, y=words, ax=ax, palette='viridis')
        ax.set_title(f'Top {top_n} Keywords for Emotion: {emotion}')
        ax.set_xlabel('Frequency')
        ax.set_ylabel('Keywords')

    plt.show()


plot_top_keywords_for_emotions(keywords_by_emotion)

X_train, X_test, y_train, y_test = train_test_split(df['Text'],
df['Emotion'],test_size=0.3, random_state = 0)

countVectorizer1 = CountVectorizer(analyzer=preprocessor.clean_text)
countVector1 = countVectorizer1.fit_transform(X_train)

countVector2 = countVectorizer1.transform(X_test)

tfidf_transformer_xtrain = TfidfTransformer()
x_train = tfidf_transformer_xtrain.fit_transform(countVector1)

tfidf_transformer_xtest = TfidfTransformer()
x_test = tfidf_transformer_xtest.fit_transform(countVector2)


def get_prec_recall_per_emotion(cm):
```

```
    n = len(cm)
    per_emotion_precision = []
    per_emotion_recall = []

    for i in range(n):
        # Обчислення Precision
        col_sum = sum(cm[j][i] for j in range(n))  # Сума по стовпцю
        true_positive = cm[i][i]
        precision = round((true_positive / col_sum) * 100, 3) if col_sum != 0 else
0.0
        per_emotion_precision.append(precision)

        # Обчислення Recall
        row_sum = sum(cm[i][j] for j in range(n))  # Сума по рядку
        recall = round((true_positive / row_sum) * 100, 3) if row_sum != 0 else 0.0
        per_emotion_recall.append(recall)

    return per_emotion_precision, per_emotion_recall


from sklearn.pipeline import Pipeline
import dill

models = [
    (SGDClassifier(), 'Blues'),
    (LogisticRegression(), 'PuRd'),
    (RandomForestClassifier(n_estimators=10, random_state=0), 'hot_r'),
    (MultinomialNB(), 'Reds'),
    (tree.DecisionTreeClassifier(), 'Greens')
]

# Підготовка списків для збереження метрик моделей
precision_dict = {}
recall_dict = {}
accuracy_dict = {}

# Цикл для оцінки моделей та їх збереження
for model, cmap in models:
    print(f'Evaluating {model.__class__.__name__}')

    # Навчання та оцінка моделі
    model.fit(x_train, y_train)
    y_pred = model.predict(x_test)
    cm = confusion_matrix(y_test, y_pred)

    # Виведення метрик
    acc = accuracy_score(y_test, y_pred)
    prec = precision_score(y_test, y_pred, average='macro')
    recal = recall_score(y_test, y_pred, average='macro')
    f1 = f1_score(y_test, y_pred, average='macro')
```

```
    print(f'Accuracy: {acc:.3f}')
    print(f'Precision: {prec:.3f}')
    print(f'Recall: {recal:.3f}')
    print(f'F1-score: {f1:.3f}')
    print(classification_report(y_test, y_pred))

    # Візуалізація матриці плутанини
    cm_display = ConfusionMatrixDisplay(cm, display_labels=model.classes_)
    fig, ax = plt.subplots(figsize=(8, 8))
    cm_display.plot(ax=ax, cmap=cmap)

    ax.set_title(f'{model.__class__.__name__}', fontsize=14, color='black')
    plt.show()

    # Обчислення метрик по кожній емоції
    per_emotion_precision, per_emotion_recall = get_prec_recall_per_emotion(cm)
    precision_dict[model.__class__.__name__] = per_emotion_precision
    recall_dict[model.__class__.__name__] = per_emotion_recall
    accuracy_dict[model.__class__.__name__] = acc * 100

    pipe = Pipeline([
        ('preprocessor', TextPreprocessor()),  # Передобробка тексту
        ('vectorizer', CountVectorizer()),     # Векторизація тексту
        ('tfidf', TfidfTransformer()),         # TF-IDF трансформер
        ('classifier', model)                  # Навчена модель
    ])

    filename = f"{model.__class__.__name__.lower()}_pipeline.pkl"
    with open(filename, "wb") as f:
        dill.dump(pipe, f)

    print(f"{model.__class__.__name__} has been saved successfully as {filename}")
    print('-' * 50)


# Створення таблиць точності та повноти
precision_df = pd.DataFrame(precision_dict, index=model.classes_)
recall_df = pd.DataFrame(recall_dict, index=model.classes_)

# Відображення результатів
print("Precision per emotion:")
display(precision_df)
print("Recall per emotion:")
display(recall_df)


metrics_df = pd.DataFrame({
    'Model': list(accuracy_dict.keys()),
    'Accuracy (%)': [round(acc, 2) for acc in accuracy_dict.values()]
```

```
})

fig, ax = plt.subplots(figsize=(10, 6))
bars = ax.barh(metrics_df['Model'], metrics_df['Accuracy (%)'], height=0.6,
color='black')
colors = ['#B9E396', '#FFB49C', '#B096E3', '#FFDF80', '#E396B6', '#7CAADE']

for i, bar in enumerate(bars):
    bar.set_color(colors[i % len(colors)])
for i, row in metrics_df.iterrows():
    ax.text(row['Accuracy (%)'] + 0.5, i, f"{row['Accuracy (%)']}%", color='black',
fontsize=10)

ax.set_xlim(0, 100)
ax.set_xlabel('Accuracy (%)', color='black')
ax.set_ylabel('Models', color='black')
ax.set_title('Accuracy for Models', color='black')
plt.show()
```

2. Побудова додатку

```
import streamlit as st
import pandas as pd
import numpy as np
import altair as alt
import nltk
import string
import re
import dill
import plotly.graph_objects as go

with open("text_emotion_5.pkl", "rb") as f:
    pipe = dill.load(f)

def predict_emotions(docx):
    results = pipe.predict([docx])
    return results[0]

def get_prediction_proba(docx):
    decision_scores = pipe.decision_function([docx])

    # Стабільний softmax
    exp_scores = np.exp(decision_scores - np.max(decision_scores, axis=1,
keepdims=True))
    probabilities = exp_scores / np.sum(exp_scores, axis=1, keepdims=True)
    return probabilities

# Створення радарної діаграми
def create_radar_chart(probabilities, emotions):
```

```
    fig = go.Figure()
    fig.add_trace(go.Scatterpolar(
        r=probabilities,
        theta=emotions,
        fill='toself',
        fillcolor='rgba(255, 99, 71, 0.6)',
        line=dict(color='rgba(255, 99, 71, 1)', width=2),
    ))
        fig.update_layout(
        polar=dict(
            radialaxis=dict(
                visible=True,
                 range=[0, 1],
                gridcolor='rgba(211, 211, 211, 0.5)',
                gridwidth=0.5
            ),
            angularaxis=dict(
                gridcolor='rgba(211, 211, 211, 0.5)',
                gridwidth=0.5
            ),
        ),
        showlegend=False,
        template='plotly_dark',
        margin=dict(l=60, r=60, t=10, b=60)
    )
    return fig

# Створення стовпчикової діаграми
def create_bar_chart(probabilities, emotions):
    df = pd.DataFrame({'emotions': emotions, 'probability': probabilities})
    fig = alt.Chart(df).mark_bar().encode(
        x=alt.X('emotions', sort=None, axis=alt.Axis(title="Emotions")),
        y=alt.Y('probability', axis=alt.Axis(title="Probability")),
        color='emotions'
    ).properties(
        width=500,
        height=400
    )
    return fig

def main():
    st.title("Emotion Recognition from Text")
    st.subheader("Analyze and identify emotions in text")

    st.markdown(
        """
        <style>
        .custom-container {
            background-color: #cf9346;
            border-radius: 8px;
```

```
            padding: 15px 0 15px 15px;
        }
        .custom-padding {
            padding: 0 15px 0 15px;
        }
        </style>
        """, unsafe_allow_html=True
    )

    with st.form(key='my_form'):
        raw_text = st.text_area("", placeholder="Type your message here...")
        submit_text = st.form_submit_button(label='Confirm')

    if submit_text:
        col1, col2 = st.columns(2)

        prediction = predict_emotions(raw_text)
        probability = get_prediction_proba(raw_text)
        probabilities = get_prediction_proba(raw_text)[0]
        emotions = pipe.classes_

        with col1:
            st.markdown('<p class="custom-container">Original Text</p>', True)
            st.markdown(f'<p class="custom-padding">{raw_text}</p>', True)

        with col2:
            st.markdown('<p class="custom-container">Prediction</p>', True)
            sub_col1, sub_col2 = st.columns(2)

            with sub_col1:
                st.markdown(f'<p class="custom-padding">Emotion: {prediction}</p>',
True)
            with sub_col2:
                st.markdown(f'<p class="custom-padding">Probability:
{np.max(probability):.4f}</p>', True)

        st.markdown('<p class="custom-container">Prediction probability</p>', True)
         # Створення діаграм
        radar_chart = create_radar_chart(probabilities, emotions)
        bar_chart = create_bar_chart(probabilities, emotions)

        tab1, tab2 = st.tabs(["Bar Chart", "Radar Hexagon"])
        with tab1:
            st.altair_chart(bar_chart, use_container_width=True)
        with tab2:
            st.plotly_chart(radar_chart, use_container_width=True)


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

ДОДАТОК Б
РЕЗУЛЬТАТИ РОБОТИ ІНТЕРАКТИВНОГО ДОДАТКУ

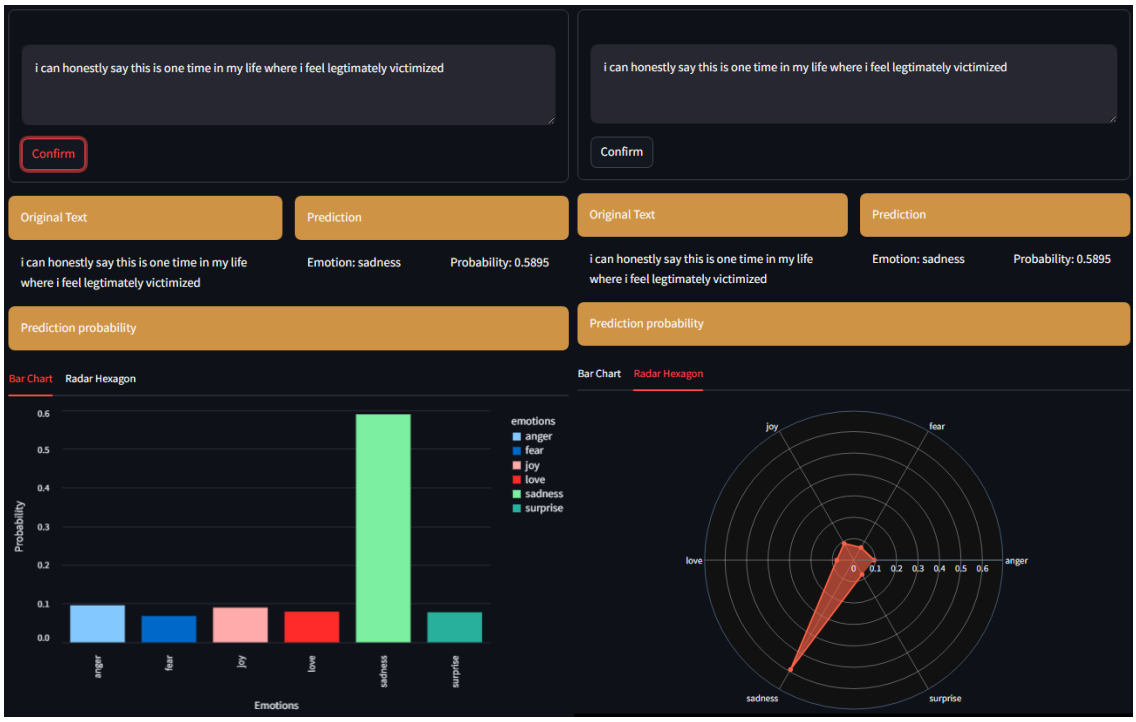

Рисунок Б.1 – Результати визначення емоційного забарвлення (смуток)

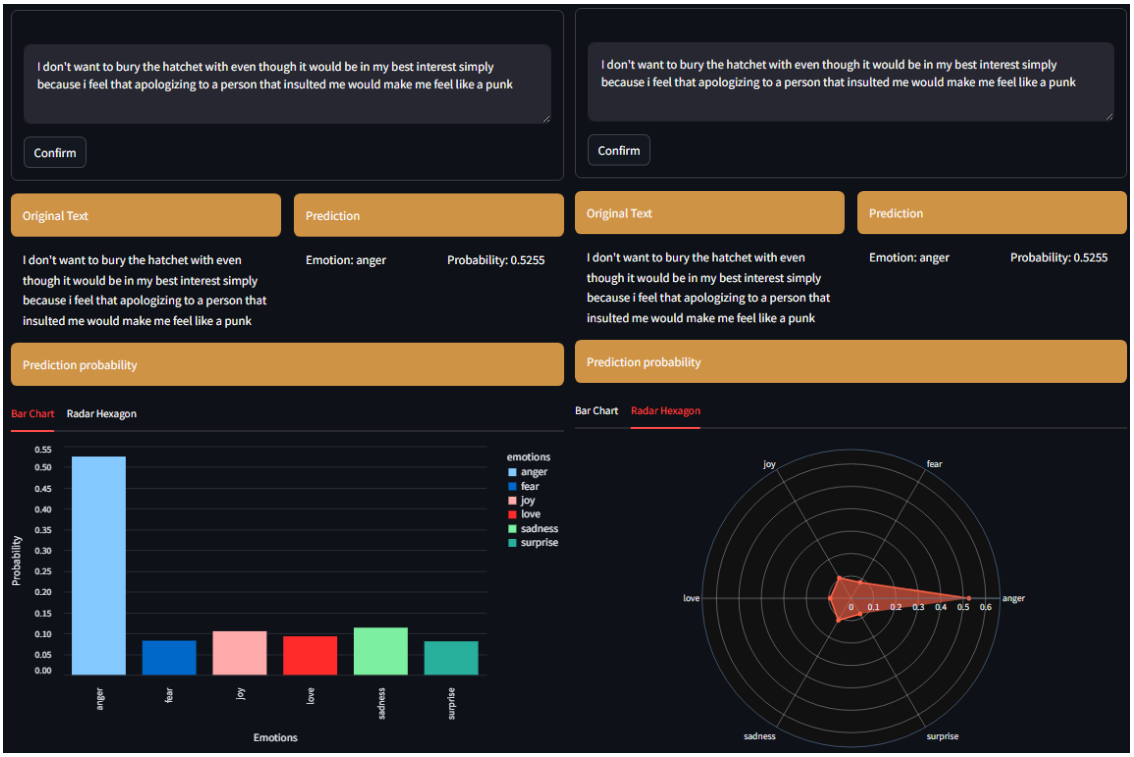

Рисунок Б.2 – Результати визначення емоційного забарвлення (гнів)

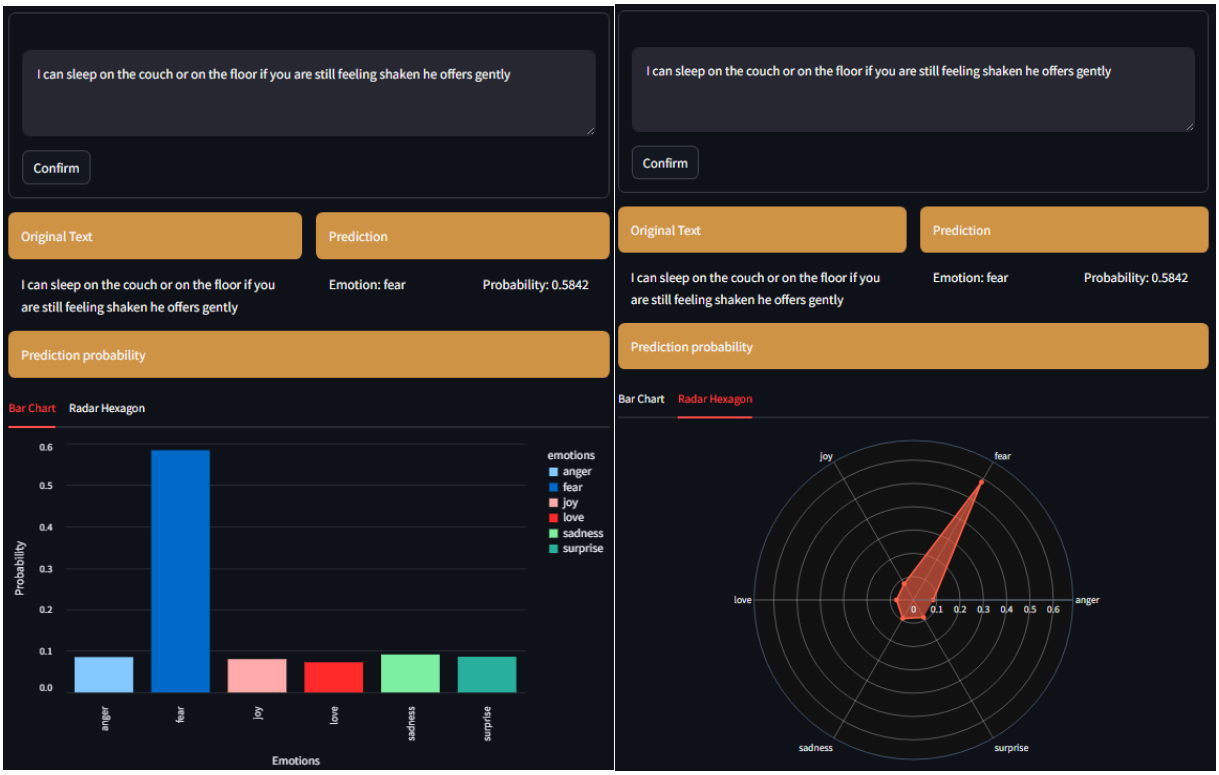

Рисунок Б.3 – Результати визначення емоційного забарвлення (страх)

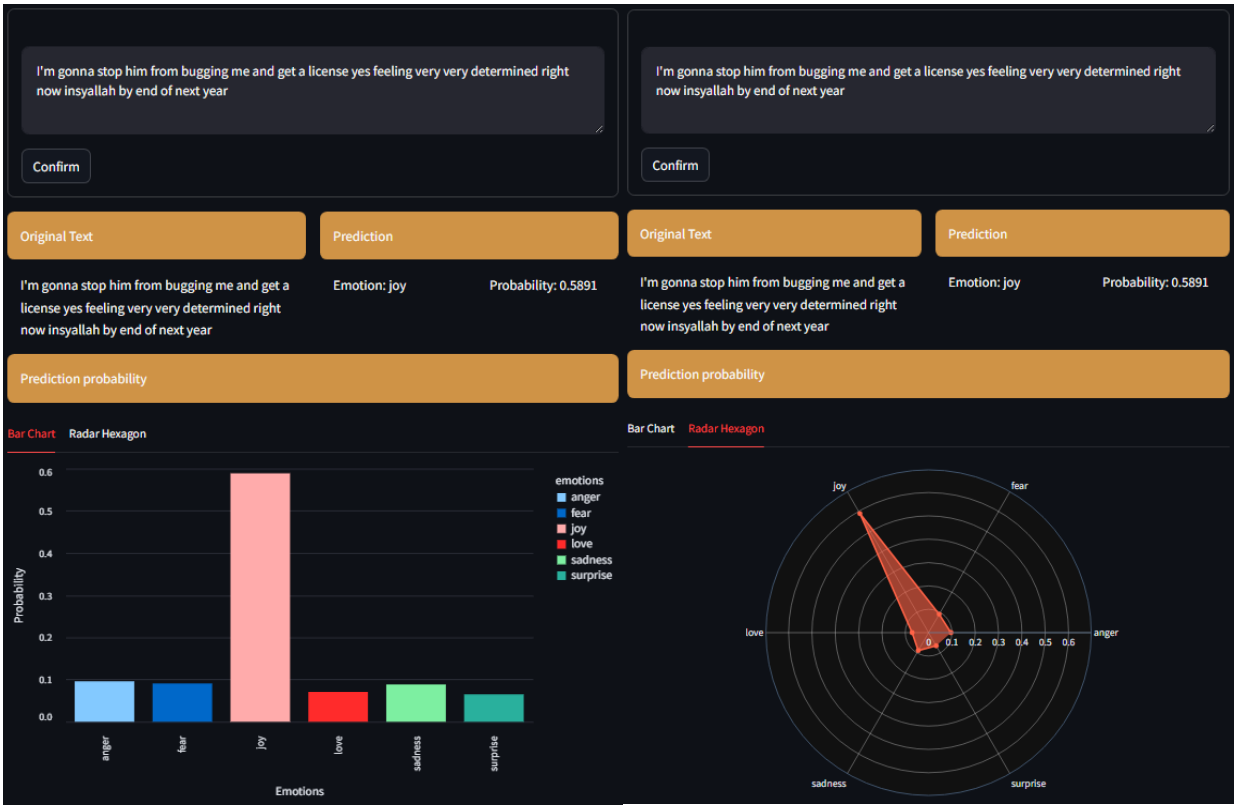

Рисунок Б.4 – Результати визначення емоційного забарвлення (радість)

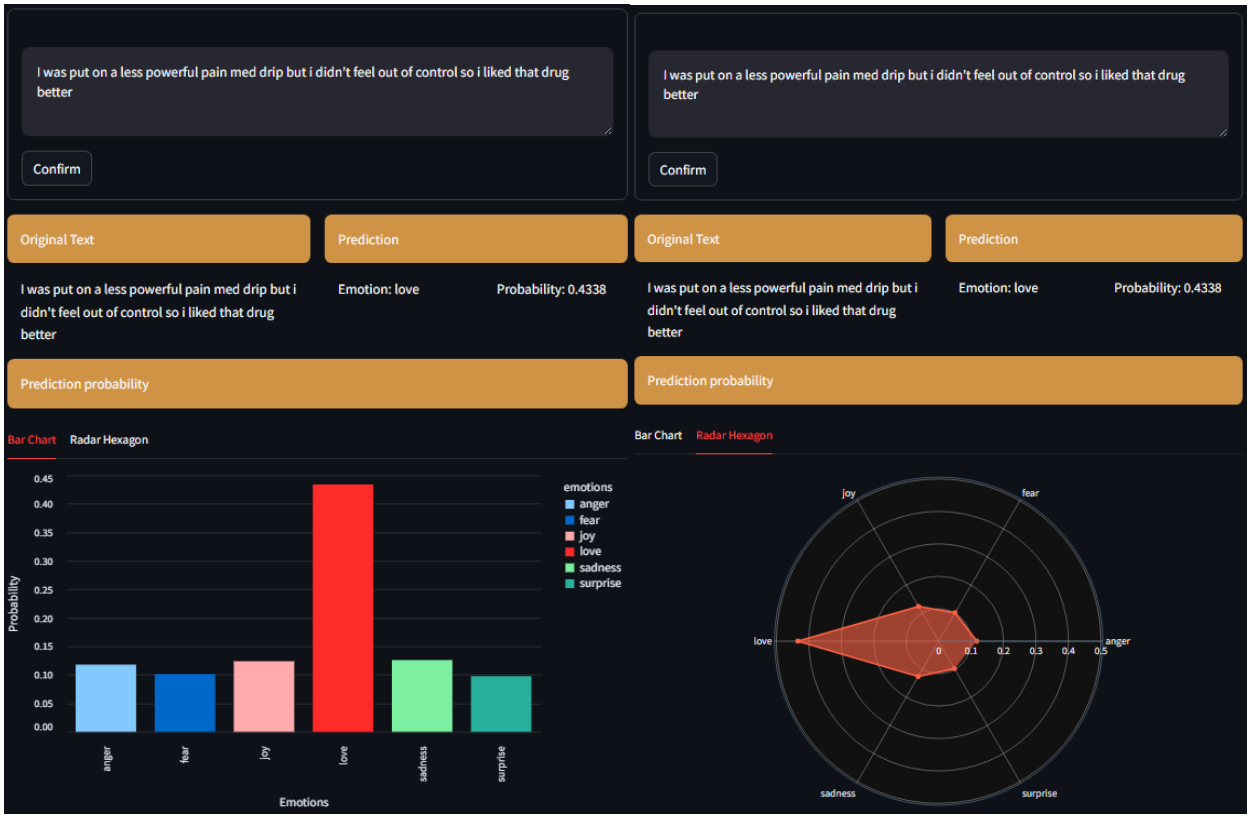

Рисунок Б.5 – Результати визначення емоційного забарвлення (любов)

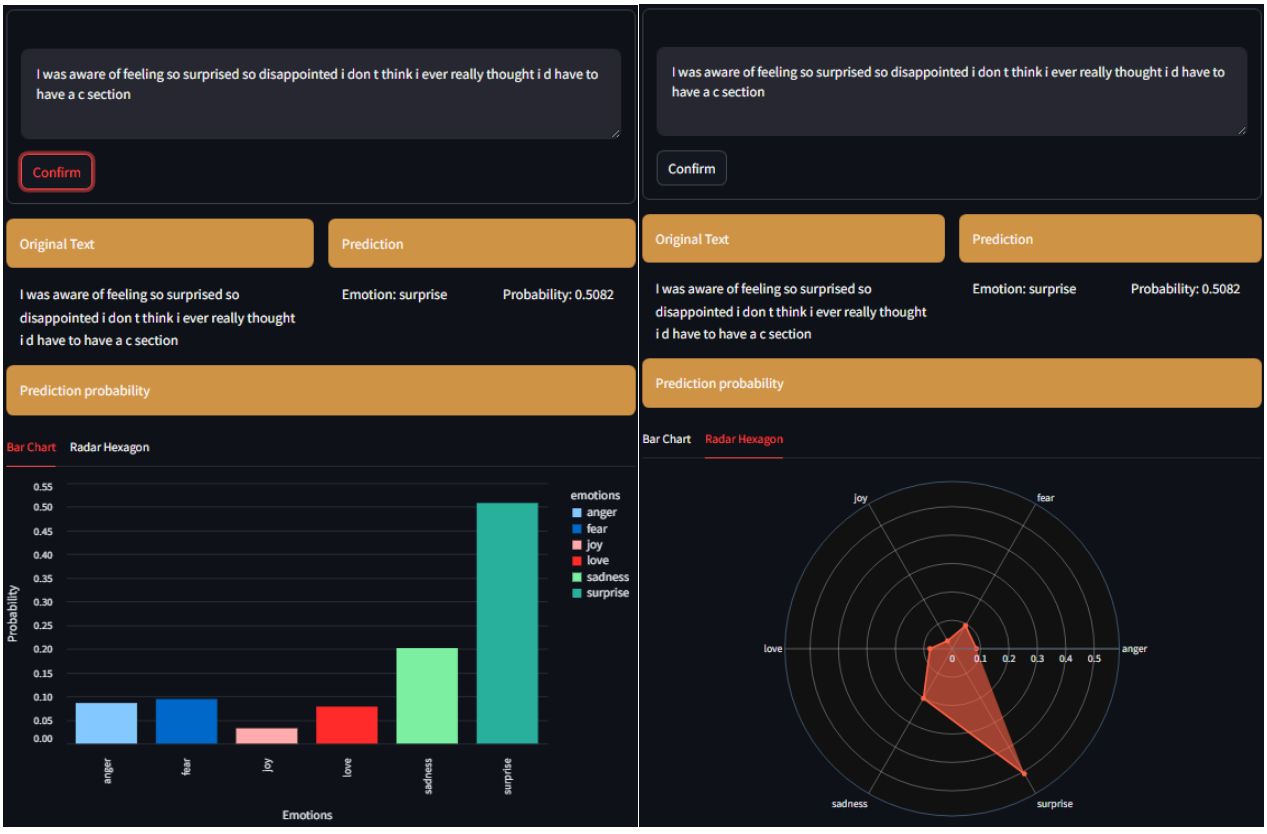

Рисунок Б.6 – Результати визначення емоційного забарвлення (здивування)