КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра теоретичної кібернетики

Кваліфікаційна робота

на здобуття ступеня бакалавра
за спеціальністю 122 Комп'ютерні науки
на тему:

Визначення емоцій у текстах за допомогою машинного навчання.

Виконав: студент 4-го курсу
Вінь Хієн НГУЄН _____
(підпис)

Науковий керівник:
Кандидат технічних наук
Сергій КОНДРАТЮК _____
(підпис)

Засвідчую, що в цій роботі
немає запозичень з праць інших авторів
без відповідних посилань.
Студент _____
(підпис)

Роботу розглянуто й допущено до захисту
на засіданні кафедри теоретичної кібернетики
«____» _____ 2023 р.,
протокол № ___
Завідувач кафедри
доктор фіз.-мат. наук, професор
Юрій КРАК

_____
(підпис)

КИЇВ-2023

РЕФЕРАТ

Обсяг роботи 45 сторінок, 23 ілюстрацій, 9 джерел посилань, 1 додаток.

МАШИНЕ НАВЧАННЯ, КЛАСИФІКАЦІЯ ТЕКСТУ, ВИЗНАЧЕННЯ ЕМОЦІЙ, PYTHON, LSVM, NAIV BAYES, RANDOM FOREST, КОМБІНОВАНИЙ ПІДХІД, LOGISTIC REGRASSION, ОБРОБКА ДАНИХ.

Об’єм роботи є аналіз підходів та побудова ансамблевого модуля для визначення емоцій у тексті за допомогою машинного навчання.

Об'єктом даної роботи є використання машинного навчання з ансамблевим навчання для класифікації емоцій у тексті.

Метою цієї роботи є розробка штучного інтелекту, здатного автоматично класифікувати емоцій у тексті за допомогою ансамблевого навчання.

Додатковою метою є аналіз кількох різних моделей та побудова ансамблевого модуля для даної задачі.

У цій роботі використовуються наступні методи розроблення: машинне навчання, нормалізація слів у тексті та визначення емоцій у тексті. Для реалізації використовуються мова програмування Python та бібліотека для машинного навчання Scikit-learn.

Результатом роботи є аналіз чотирьох моделей машинного навчання та створення ансамблеву модель. Усі чотири моделі виявилися ефективними та точними у передбаченнях. Було проведено порівняльний аналіз цих моделей та їх результатів.

Ця програма може бути застосована у різних галузях, таких як маркетинг та реклама, соціальні медіа та веб-платформи, контент-аналітика, комп'ютерні ігри та віртуальна реальність, клінічна психологія та дослідження.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ **4**

ВСТУП **6**

1 ПРЕДМЕТНА ОБЛАСТЬ **9**

1.1 9

1.2 11

1.3 13

1.4 13

1.5 15

1.6 Комбінований підхід 16

1.7 Мова Python та інші бібліотеки для реалізації 17

Інші бібліотеки інструментів для реалізації 18

2. Аналіз підходів до розпізнавання емоцій у текстах **20**

2.1 Аналіз існуючих підходів 20

2.2 Аналіз результатів тренувань підходів 20

2.3 Порівняння аналізу результатів тренувань інших підходів з комбінованим підходом 26

2.4 28

3 ПРОГРАМНА РЕАЛІЗАЦІЯ **34**

3.1 34

3.2 37

3.3 38

3.4 39

3.5 39

3.6 39

3.7 40

3.8 Результати тестування моделі комбінованого класифікатора 40

ВИСНОВОК **42**

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ **43**

Додаток А **45**

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАЧЕННЯ

МН - машинне навчання

SVC - support vector machine, машина опорних векторів

NLTK - Natural Language Toolkit, інструментарій природної мови

IDE - Integrated development environment, Інтегроване середовище розробки

IDF - inverse document frequency, обернена частота документа

TF - term frequency, частота слова

ВСТУП

Оцінка сучасного об'єкта розробки в контексті виявлення емоцій у тексті за допомогою машинного навчання включає оцінку існуючих моделей, алгоритмів та методів, що використовуються для вирішення цієї задачі. Наразі існує велика кількість підходів, які забезпечують високу точність виявлення емоцій, але деякі питання, такі як розуміння контексту та семантики, все ще залишаються відкритими.

З точки зору оцінки, багато сучасних моделей для виявлення емоцій у тексті базуються на машинному навчанні та нейромережевих архітектурах. Наприклад, моделі на основі рекурентних нейронних мереж (RNN) або трансформаторів показують хороші результати в розпізнаванні емоційних станів. Крім того, використання попереднього навчання на великій кількості даних допомагає підвищити якість моделей і забезпечити кращі результати.

Актуальність роботи та підстави для виконання її написання полягає у зростаючій кількості текстових даних, що виникають у різних сферах життя. Люди активно висловлюють свої емоції та думки в текстовій формі, особливо в соціальних мережах, блогах, форумах, коментарях та відгуках. Аналіз цих текстів на предмет виявлення емоцій може мати значний вплив на різні сфери.

Причини для виявлення емоцій у тексті за допомогою машинного навчання наступні:

1. Збільшення обсягу текстових даних: Завдяки зростаючому використанню соціальних мереж та інших онлайн-платформ, обсяг доступних текстових даних значно збільшився. Це дає можливість використовувати машинне навчання для аналізу та розуміння емоцій, виражених у цих даних.
2. Потреба в автоматичному аналізі емоцій: У різних галузях, таких як маркетинг, психологія, клінічна медицина, соціологія, важливо мати

можливість автоматично аналізувати і розуміти емоційні стани людей на основі текстових даних. Це допомагає приймати обґрунтовані рішення та надає нові можливості для досліджень і застосування.

3. Підвищення точності та якості аналізу: Сучасні підходи машинного навчання можуть підвищити точність виявлення емоцій у тексті. Це дає можливість розробляти більш точні та надійні моделі, які можна використовувати в реальних ситуаціях.

Мета і завдання роботи:

Метою даного завдання є створення штучного інтелекту, який здатний класифікувати текст, де текстові дані перетворені у токени, і виявляти емоційні слова серед цих токенів. Крім того, потрібно порівняти різні моделі та розробити ансамблеву модель.

Для цього нам необхідно:

1. Дані для навчання: Необхідно мати доступ до великого обсягу текстових даних, які мають відповідні етикетки або мітки емоцій. Ці дані можуть бути зібрані вручну, де люди анотують текстові зразки з відповідними емоційними категоріями. Важливо, щоб ці дані були репрезентативними та різноманітними, щоб модель мала змогу навчитися розпізнавати різні емоційні вирази.
2. Попередня обробка даних: Перед початком навчання необхідно виконати попередню обробку даних, включаючи токенізацію (розбиття тексту на окремі слова або фрази), очищення від зайвих символів, лематизацію (перетворення слів на їх базові форми) та векторизацію (перетворення текстових даних на числові вектори). Цей процес допомагає підготувати дані для подальшого використання в моделі машинного навчання.
3. Вибір моделі: Наступним кроком є вибір відповідної моделі машинного навчання.

4. Тренування моделі: Після вибору моделі необхідно провести процес тренування. Це включає передачу підготовлених даних моделі та налаштування параметрів моделі для досягнення найкращих результатів. Під час тренування модель адаптується до вхідних даних та навчається розпізнавати емоції на основі зразків
5. Оцінка та підвищення продуктивності моделі: Після тренування моделі необхідно оцінити її продуктивність на валідаційному наборі даних. Використовуються різні метрики, такі як точність, відновлення, F-мера тощо.
6. Тестування та розгортання моделі

Об'єктом дослідження є процес визначення емоцій у тексті за допомогою машинного навчання. У цьому контексті, об'єктом дослідження є сам алгоритм або модель машинного навчання, яка навчається розпізнавати та класифікувати емоції в текстових даних.

Предметом дослідження може включати в себе вибір та порівняння різних моделей, аналіз їх продуктивності та ефективності, а також пошук оптимальних методів та параметрів для досягнення найкращих результатів у визначенні емоцій у тексті.

Засоби реалізації можуть включати такі елементи: мова програмування Python та декілька бібліотек, зокрема Tensorflow, Scikit-learn, Pandas, matplotlib, numpy і seaborn.

Можливі сфери застосування:

- Маркетинг та реклама
- Соціальні медіа та веб-платформи
- Контент-аналітика
- Комп'ютерні ігри та віртуальна реальність

- Клінічна психологія та дослідження

1 ПРЕДМЕТНА ОБЛАСТЬ

1.1 Машинне навчання

Машинне навчання, також відоме як machine learning, є підгалуззю інформатики у сфері штучного інтелекту. Воно часто використовує статистичні методи для навчання комп'ютерів здатності "вчитися" з даних без явного програмування.

Термін "машинне навчання" був вперше використаний у 1959 році Артуром Семюелем. Ця галузь розвинулася з досліджень розпізнавання образів та обчислювального навчання в сфері штучного інтелекту. Машинне навчання досліджує вивчення та розробку алгоритмів, які можуть самостійно навчатися та робити передбачення на основі даних. Ці алгоритми працюють шляхом побудови моделі на основі вхідних даних і здатні здійснювати прогнози та приймати рішення, не дотримуючись жорстких програмних інструкцій.

Машинне навчання застосовується в багатьох обчислювальних задачах, де розробка ефективних алгоритмів шляхом традиційного програмування є складною або неможливою. Прикладами застосування є фільтрування електронної пошти, виявлення мережевих загроз або зловмисних інсайдерів, оптичне розпізнавання символів (OPC), навчання ранжуванню та комп'ютерне зіру.

Машинне навчання має глибокі зв'язки з обчислювальною статистикою, яка також фокусується на прогнозуванні за допомогою комп'ютерів. Воно також пов'язане з математичною оптимізацією, яка надає методи і теорію для цієї галузі. Іноді машинне навчання поєднується з обробкою даних, де друга підгалузь, відома як навчання без учителя, фокусується на розвідувальному

аналізі даних. Машинне навчання також може бути використано для навчання та моделювання основних характеристик різних суб'єктів, а також для виявлення виразних аномалій.

У сфері аналізу даних машинне навчання є методом, який використовується для розробки складних моделей та алгоритмів для прогнозування. В комерційному застосуванні це відомо як передбачувальна аналітика. Ці аналітичні моделі дозволяють дослідникам, науковцям з даних, інженерам та аналітикам зробити надійні та повторювані рішення, а також отримати глибше розуміння шляхом аналізу історичних зв'язків та тенденцій у даних.

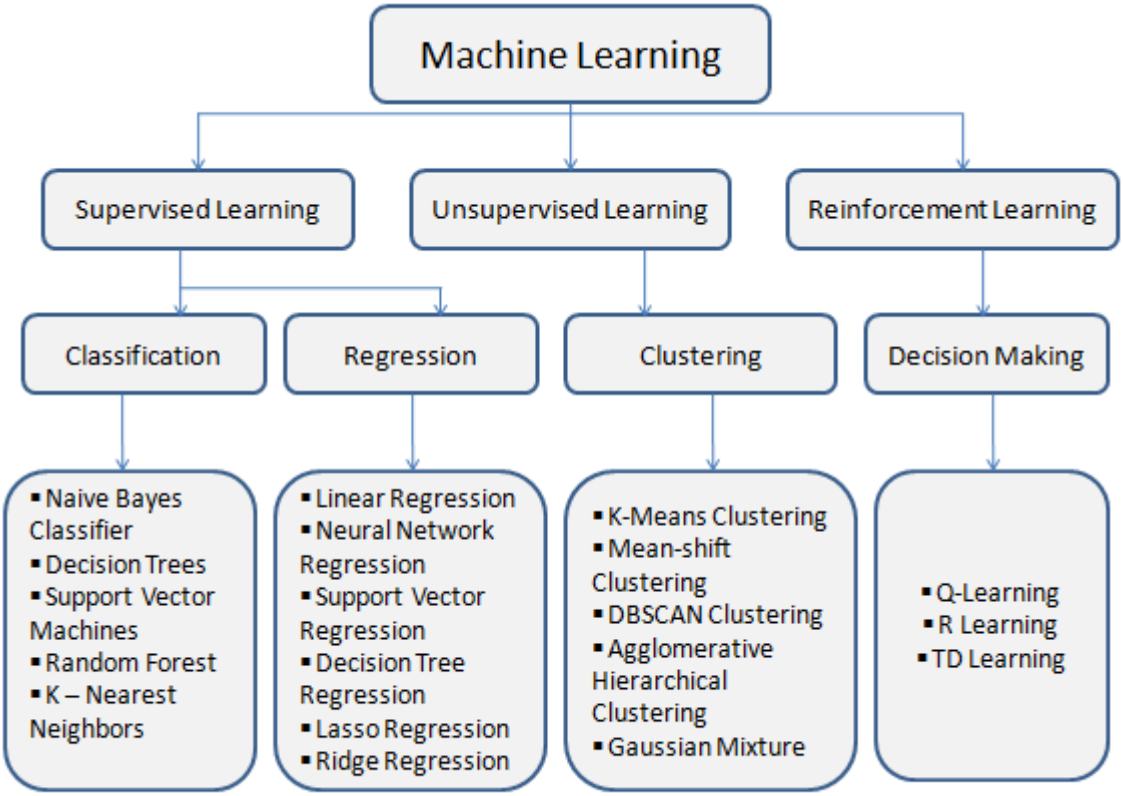

Рисунок 1.1 – Приклад типів машинного навчання

Задача класифікації є однією з основних задач штучного інтелекту, яка використовується для визначення приналежності об'єкта до однієї з певних категорій або класів на підставі його характеристик або ознак. Основна мета полягає в тому, щоб побудувати модель або алгоритм, який може автоматично визначати клас об'єкта на основі навчальних даних.

У задачі класифікації зазвичай є певний набір класів, до яких може належати об'єкт. Навчальні дані складаються з прикладів, де кожен приклад має вхідні ознаки або характеристики, які описують об'єкт, а також мітку або клас, до якого він належить. На основі цих даних модель машинного навчання навчається встановлювати зв'язок між вхідними ознаками та відповідним класом.

Після навчання модель може бути використана для класифікації нових, раніше невідомих об'єктів. Вона приймає вхідні ознаки цих об'єктів і передбачає їх клас на основі отриманих знань з навчання.

1.2 Класифікатор Наївний Байєс

Наївний Байєс є алгоритмом класифікації для різних типів задач, таких як бінарна та багатокласова класифікація. Його називають "наївним" або "ідіотським" Байєсом через його спрощені обчислення ймовірностей, що робить їх доступними для розрахунку. Замість обчислення ймовірностей кожного значення атрибуту у відношенні до гіпотези, використовуються припущення про умовну незалежність та обчислюються окремо для кожного атрибуту з урахуванням цільового значення.

Це спрощення припускає, що атрибути не взаємодіють між собою, що є неправдоподібним у реальних даних. Однак, наївний Баєсовий алгоритм працює добре навіть у випадках, коли це припущення не виконується. Його сильна

сторона полягає в його ефективності та швидкості обчислень, що робить його популярним для багатьох задач класифікації, зокрема аналізу тексту та фільтрації спаму.

У машинному навчанні ми часто зацікавлені в тому, щоб вибрати найкращу гіпотезу (h) на основі доступних нам даних (d).

У задачі класифікації, наприклад, гіпотеза (h) може представляти клас, який потрібно призначити новому набору даних (d).

Один з найпростіших способів вибору найбільш ймовірної гіпотези з використанням наявних даних полягає в теоремі Байєса, яка дозволяє обчислити ймовірність гіпотези, враховуючи наші попередні знання.

Теорема Байєса може бути сформульована наступним чином:

$$P(h|d) = (P(d|h) * P(h)) / P(d)$$

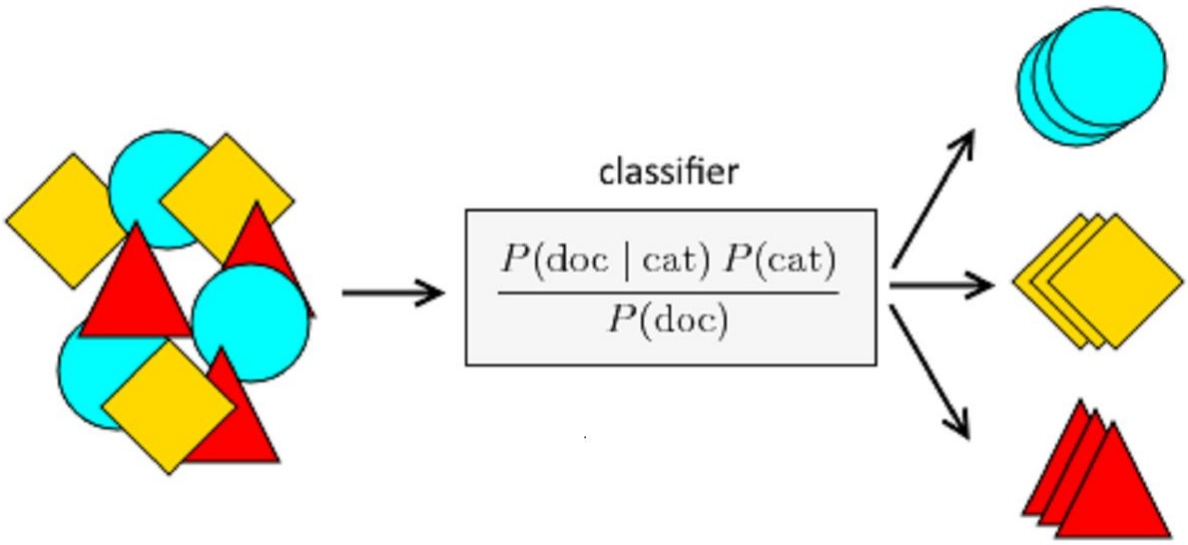

Рисунок 1.3 – Приклад алгоритму наївного Байєса

1.3 Класифікатор Випадкові Ліси

Випадкові ліси або ліси випадкових рішень - це метод ансамблевого навчання, який використовується для класифікації, регресії та інших завдань. Цей метод працює шляхом побудови багатьох дерев рішень під час навчання. Випадковий ліс повертає результат класифікації, який визначається більшістю голосів дерев. Для завдань регресії, результатом є середнє або середнє прогнозування окремих дерев.

Випадкові ліси вирішують проблему переобладнання, яка часто виникає з деревами рішень. Вони зазвичай показують кращі результати порівняно з окремими деревами рішень, хоча їх точність може бути нижчою, ніж у методів, таких як градієнтний бустинг. Продуктивність випадкових лісів також може залежати від особливостей даних.

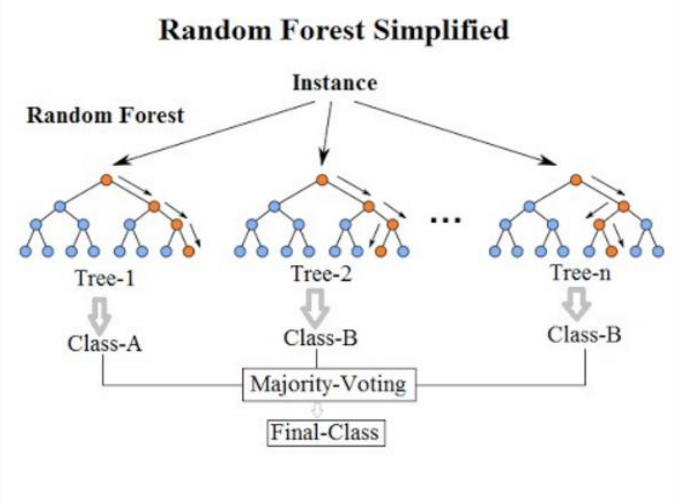

Рисунок 1.4 – Приклад діаграми лісу випадкових рішень.

1.4 Класифікатор логістичної регресії

Логістична регресія є одним з найпопулярніших алгоритмів машинного навчання, який використовується для прогнозування категоріальної залежної змінної на основі заданого набору незалежних змінних.

У логістичній регресії результат передбачається в категоричному або дискретному вигляді, наприклад, "Так" або "Ні", 0 або 1, "Істинно" або "Хибно". Замість точних значень 0 і 1, логістична регресія надає ймовірнісні значення, які знаходяться між 0 і 1.

Цей алгоритм схожий на лінійну регресію, але використовується для розв'язання задач класифікації. Замість підгонки лінії, в логістичній регресії використовується логістична функція у формі "S", яка передбачає два можливих значення (0 або 1).

Крива логістичної функції відображає ймовірність відповіді на певне явище, наприклад, ймовірність наявності ракових клітин, вірогідність миші страждати ожирінням, в залежності від їх ваги та інших факторів.

Логістична регресія є важливим алгоритмом машинного навчання, оскільки вона дозволяє надавати ймовірності та класифікувати нові дані на основі неперервних та дискретних наборів даних. Вона може бути застосована для класифікації спостережень за допомогою різних типів даних та визначення найбільш ефективних змінних для класифікації.

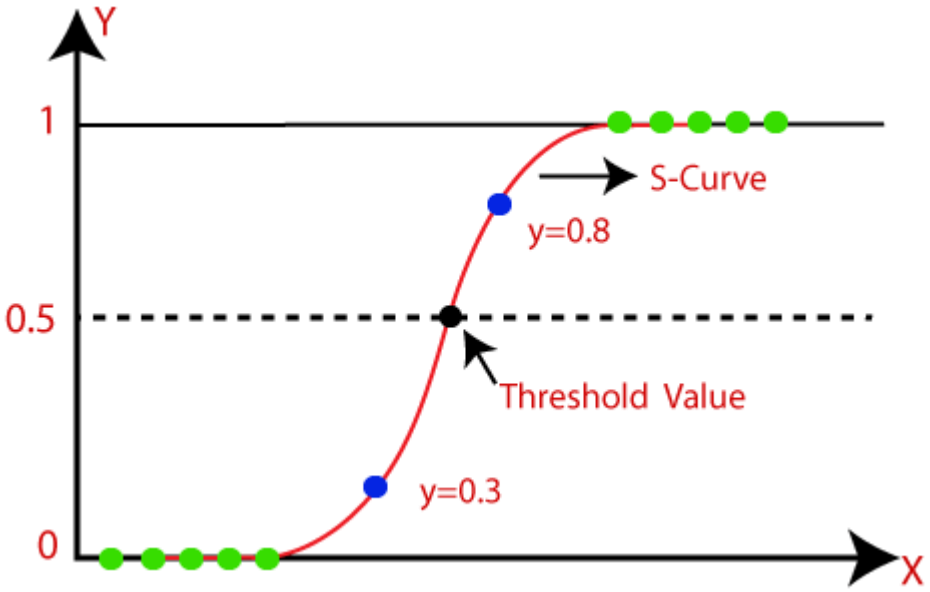

Рисунок 1.5 - Приклад функції логістики

1.5 Класифікатор SVM та Лінійний SVM

SVM, або машина опорних векторів, є одним з найпопулярніших алгоритмів навчання з учителем, який використовується для класифікації та регресії. Хоча його основне застосування полягає в класифікації в машинному навчанні.

Основною метою алгоритму SVM є створення оптимальної лінії або границі рішення, яка здатна розділити n-вимірний простір на класи, щоб ми могли легко класифікувати нові дані у відповідні категорії в майбутньому. Ця границя, яка надає найкращий результат, називається гіперплощиною.

SVM вибирає крайові точки або вектори, які сприяють створенню гіперплощини. Ці випадки, що знаходяться на межі, називаються опорними векторами, і сам алгоритм отримав назву "машина опорних векторів".

Лінійний SVM використовується для розв'язання задач класифікації, де дані можна лінійно розділити. Це означає, що якщо набір даних може бути точно розділений на два класи за допомогою прямої лінії, то такі дані

вважаються лінійно роздільними, і для їх класифікації використовується лінійний класифікатор SVM.

Нелінійний SVM використовується для розв'язання задач класифікації, де дані не можуть бути лінійно розділені. Якщо набір даних не може бути правильно класифікований за допомогою прямої лінії, то такі дані вважаються нелінійними, і для їх класифікації використовується нелінійний класифікатор SVM.

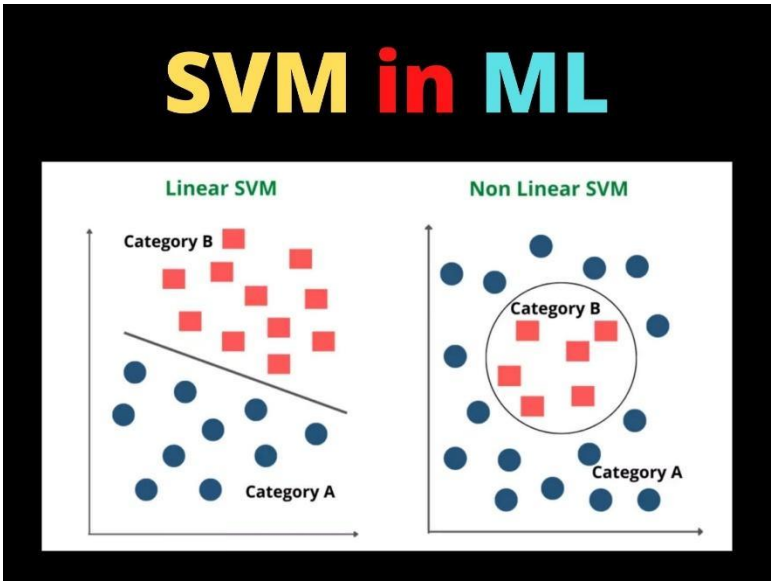

Рисунок 1.6 – Графік різниці між Лінійний SVM та Нелінійний SVM

1.6 Комбінований підхід

Комбінований підхід є одним з ефективних методів в машинному навчанні, який поєднує декілька моделей для досягнення кращої передбачувальної точності. Ідея полягає в тому, що різні моделі можуть мати різні сильні сторони та слабкі сторони, тому комбінування їх прогнозів може привести до покращення результатів.

Один з підходів комбінованого навчання - голосування меншості найкращого (англ. minority voting). У цьому підході кожна модель здійснює передбачення для вхідного зразка даних, і остаточне передбачення визначається шляхом голосування меншості найкращого. Замість того, щоб враховувати більшість голосів, ми приймаємо рішення, виходячи з меншості передбачень.

Один з популярних алгоритмів комбінованого підходу - VotingClassifier. Він дозволяє об'єднати кілька моделей, використовуючи різні алгоритми класифікації або регресії. VotingClassifier може використовувати різні стратегії голосування, такі як голосування більшості або голосування за вагою. У випадку голосування меншості найкращого, враховуються лише передбачення меншості моделей, а не більшості.

Цей комбінований підхід може бути особливо корисним в ситуаціях, коли окремі моделі мають різні сильні сторони або коли набір даних має нерівномірний розподіл класів. Використання голосування меншості найкращого дозволяє збалансувати передбачувальну силу різних моделей та забезпечити кращі результати на менш представлених класах.

У "soft" голосуванні кожна модель видає ймовірності належності до різних класів, а фінальне рішення формується шляхом підрахунку середніх або зважених значень цих ймовірностей. Наприклад, якщо у ансамблі є 5 базових моделей, і вони дають ймовірності для класу "A" як [0.8, 0.7, 0.9, 0.6, 0.75], а для класу "B" як [0.2, 0.3, 0.1, 0.4, 0.25], то при "soft" голосуванні можна взяти середнє значення кожної ймовірності і порівняти їх. У цьому випадку середня ймовірність для класу "A" буде 0.76, а для класу "B" - 0.26. Залежно від порогового значення, можна встановити, що обраною класифікацією є клас "A".

1.7 Мова Python та інші бібліотеки для реалізації

Python - це високорівнева мова програмування загального призначення, яка здобула популярність серед програмістів завдяки своїй простоті і ефективності. Вона була розроблена Гвідо ван Россумом і вперше випущена у 1991 році.

Основними особливостями Python є читабельний синтаксис і простота в освоєнні. Вона використовує відступи для позначення блоків коду, що робить код більш структурованим і зрозумілим. Python підтримує об'єктно-орієнтоване, функціональне і структурне програмування.

Python має широкий спектр бібліотек і модулів, які розширюють його можливості. Наприклад, бібліотека NumPy дозволяє працювати з числовими даними і виконувати наукові обчислення, бібліотека Pandas дозволяє ефективно аналізувати та обробляти дані, а бібліотека TensorFlow підтримує розвиток інтелектуальних систем із застосуванням нейронних мереж.

Python використовується для розробки різноманітних програм і проектів, включаючи веб-розробку, наукові дослідження, обробку даних, штучний інтелект, машинне навчання та інше. Великою перевагою Python є його велика спільнота розробників, яка надає підтримку, документацію та безліч корисних ресурсів для початківців і досвідчених програмістів.

Загалом, Python - це потужна і зручна мова програмування, яка підходить для багатьох завдань і має широкі можливості, роблячи її однією з найпопулярніших мов серед програмістів.

Інші бібліотеки інструментів для реалізації

1. Pandas - це потужна бібліотека для обробки та аналізу даних, побудована на мові програмування Python. Вона надає зручні інструменти для роботи з табличними даними, такими як датафрейми (DataFrame), що дозволяють ефективно маніпулювати, фільтрувати та аналізувати дані.

2. NumPy (Numerical Python) - це бібліотека для обчислювальної математики у мові програмування Python. Вона надає потужні масиви та функції для роботи з числовими даними.
3. NLTK (Natural Language Toolkit) - це бібліотека для обробки природної мови (Natural Language Processing, NLP) у мові програмування Python. Вона надає набір інструментів та ресурсів для роботи з текстовими даними.
4. Matplotlib - це бібліотека для візуалізації даних у мові програмування Python. Вона надає широкі можливості для створення різноманітних графіків, діаграм, даних та інших типів візуалізацій.
5. Scikit-learn (також відомий як sklearn) є однією з найпопулярніших бібліотек машинного навчання для мови програмування Python. Вона надає набір інструментів та алгоритмів для виконання завдань класифікації, регресії, кластеризації, зменшення розмірності, оцінювання моделей та багато іншого.

2. Аналіз підходів до розпізнавання емоцій у текстах

2.1 Аналіз існуючих підходів

У даному аналізу я буду описувати мій власний підхід до розпізнавання емоцій та порівнювати його з популярними методами, такими як Naive Bayes, Random Forest, Logistic Regression та Linear SVM.

Починаючи з методу Naive Bayes, я буду досліджувати його ефективність у розпізнаванні емоцій, використовуючи підготовлені набори даних. Потім перейду до Random Forest, Logistic Regression та Linear SVM, досліджуючи їх здатність до точного класифікації емоційних відгуків. В кожному випадку буду проводити оцінку результатів за допомогою показників якості, таких як точність, матриці плутанини та F-мера.

Однак, після порівняння цих методів, я детально розгляну ансамблевий підхід з м'яким голосуванням як потенційну альтернативу. Цей підхід використовує комбінацію різних класифікаторів для досягнення кращих результатів. За допомогою м'якого голосування, я зможу об'єднати внутрішні прогнози різних методів та прийняти рішення на основі їх голосів.

2.2 Аналіз результатів тренувань підходів

Для аналізу підходів нам необхідно дослідити результати тренувань на підготовлених даних, використовуючи метрику точності, матрицю плутанини та F-меру.

Давайте почнемо з методу Naive Bayes і проаналізуємо отримані результати тренування:

Accuracy: 67.02%

F1 Score: 67.02

Confusion Matrix:

[[469 32 44 28 120]

[73 420 55 16 115]

[56 18 475 68 90]

[61 20 76 385 96]

[68 20 48 15 525]]

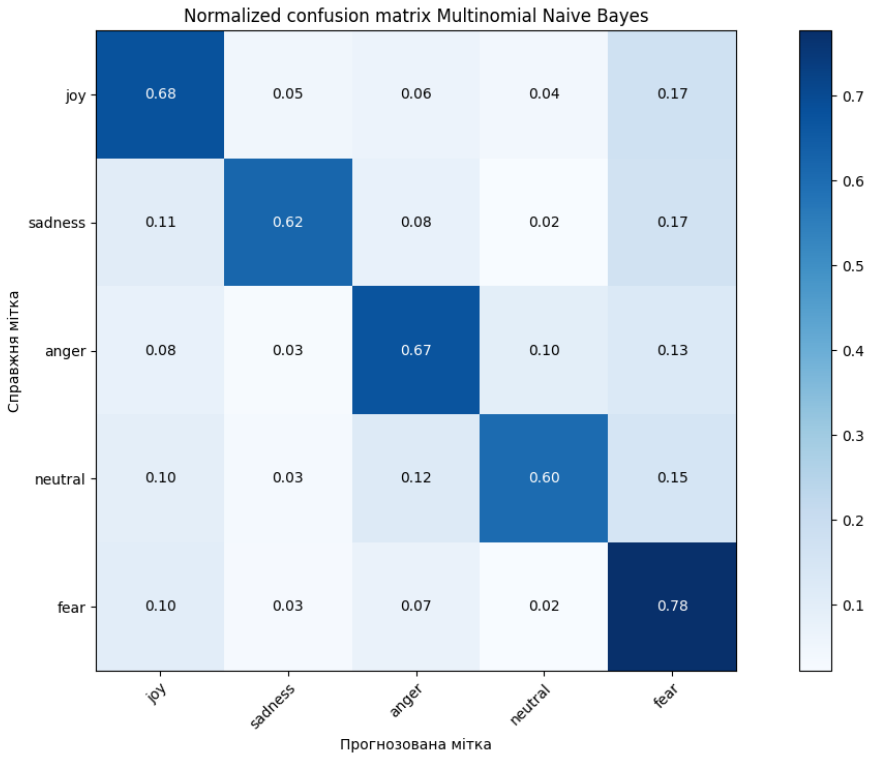

Рисунок 2.2.1 – Нормалізована матриця плутанини методу Naive Bayes

З результатів тренування методу Naive Bayes видно, що він досягає точності 67.02% та F-мери 67.02. Матриця плутанини показує, що по головній діагоналі зазначений метод правильно класифікував наступні кількості

прикладів для кожної мітки: [469, 420, 475, 385, 525]. Нормалізована матриця плутанини вказує на наступні значення по головній діагоналі: [0.68, 0.62, 0.67, 0.60, 0.78]. Це означає, що метод добре прогнозує мітку "fear",але погано прогнозує 2 і 4 міток.

Далі розглянем метод Random Forest і проаналізуємо отримані результати тренування:

Accuracy: 66.25%

F1 Score: 66.25

Confusion Matrix:

[[415 87 41 81 69]

[75 451 39 74 40]

[61 53 445 102 46]

[36 21 36 514 31]

[79 57 59 58 423]]

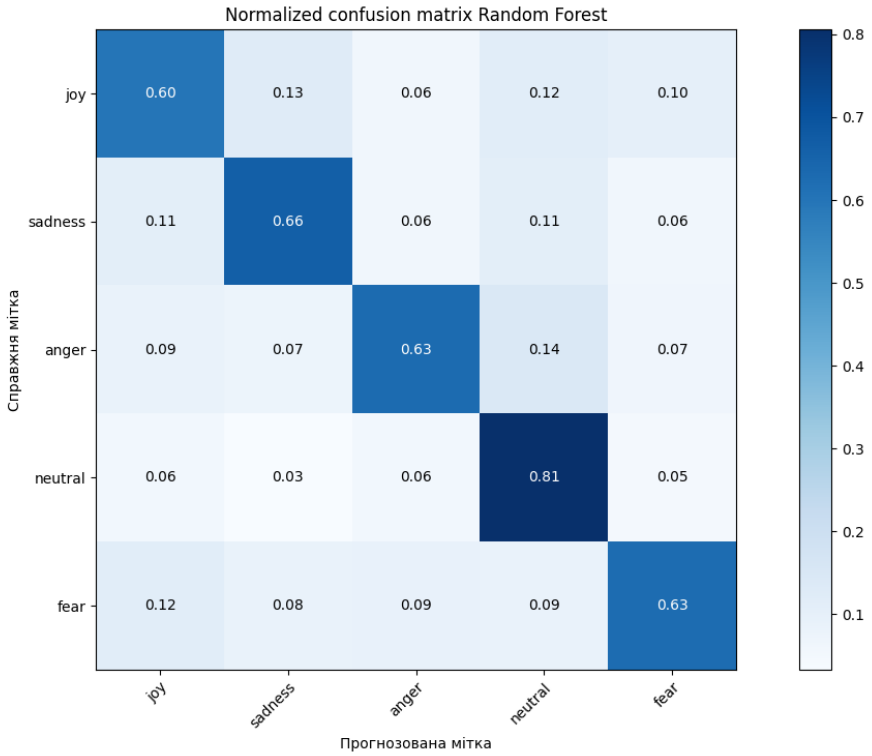

Рисунок 2.2.2 – Нормалізована матриця плутанини методу Random Forest

З аналізу результатів тренування методу Random Forest видно, що він досягає точності 66.25% та F-мери 66.25. Матриця плутанини показує, що на головній діагоналі зазначений метод правильно класифікував такі кількості прикладів для кожної мітки: [415, 451, 445, 514, 423]. За нормалізованою матрицею плутанини можна побачити, що значення на головній діагоналі складають: [0.60, 0.66, 0.63, 0.81, 0.63]. Це свідчить про те, що метод добре прогнозує мітку "neutral", але погано прогнозує 1, 3 і 5 міток.

Проаналізуємо наступний отримані результати тренування методу Logistic Regression:

Accuracy: 69.35%

F1 Score: 69.35

Confusion Matrix:

[[456 67 44 68 58]

[65 483 42 50 39]

[56 34 476 101 40]

[41 23 42 498 34]

[82 60 51 43 440]]

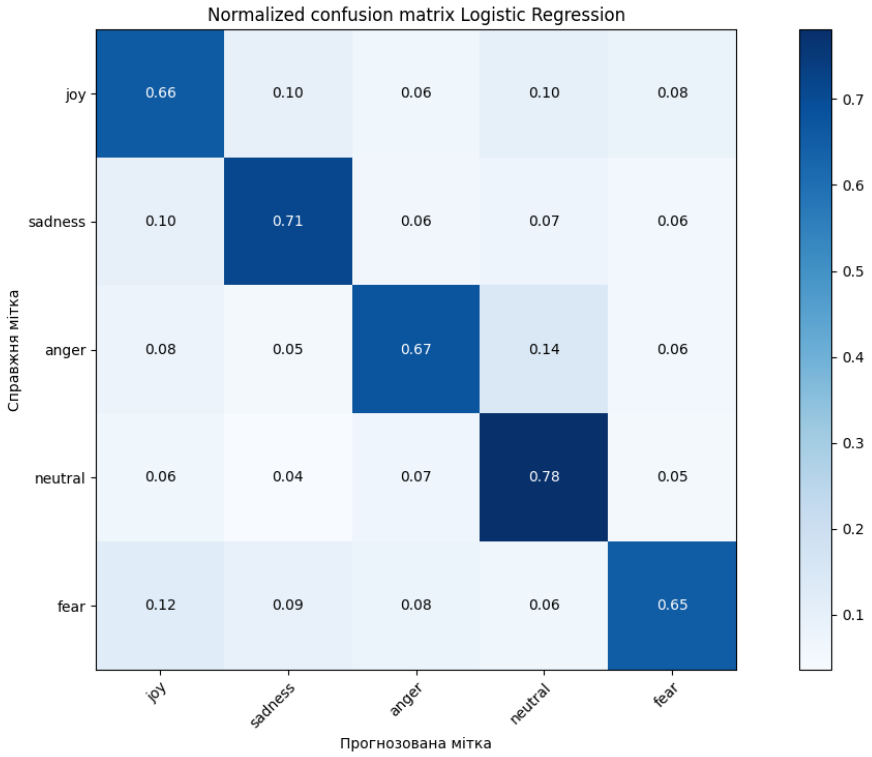

Рисунок 2.2.3 – Нормалізована матриця плутанини методу Logistic Regression

Після аналізу результатів тренування методу Logistic Regression видно, що він досягає точності 69.35% та F-мери 69.35. Матриця плутанини показує, що на головній діагоналі зазначений метод правильно класифікував такі кількості прикладів для кожної мітки: [456, 483, 476, 498, 440]. За нормалізованою матрицею плутанини видно, що значення на головній діагоналі складають: [0.66, 0.71, 0.67, 0.78, 0.65]. Це свідчить про те, що метод добре прогнозує міток "sadness" та "neutral", але з порівняно з Random Forest він прогнозує мітку "neutral" трішки гірше.

Кінцевий аналіз результатів тренування Linear SVM:

Accuracy: 71.26%

F1 Score: 71.26

Confusion Matrix:

[[474 56 44 63 56]

[64 502 33 39 41]

[50 36 487 95 39]

[40 21 42 498 37]

[81 47 44 47 457]]

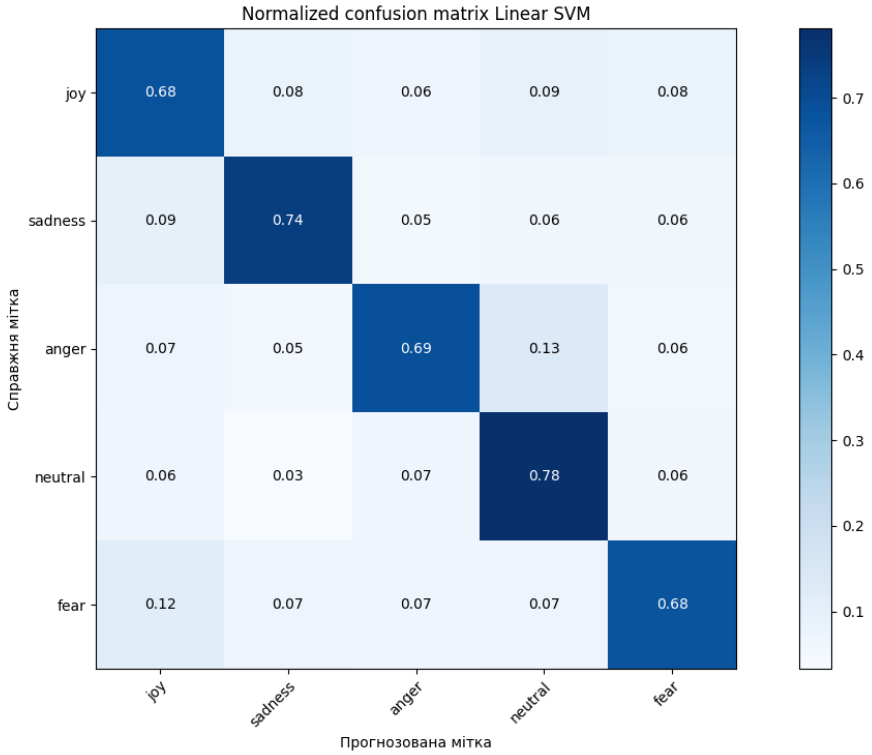

Рисунок 2.2.3 – Нормалізована матриця плутанини методу Linear SVM

З результатів аналізу тренування методу Linear SVM видно, що він досягає точності 71.26% та F-мери 71.26. Матриця плутанини показує, що на головній діагоналі зазначений метод правильно класифікував наступні кількості прикладів для кожної мітки: [474, 502, 487, 498, 457]. За нормалізованою матрицею плутанини можна побачити, що значення на головній діагоналі

становлять: [0.68, 0.74, 0.69, 0.78, 0.68]. Це свідчить про те, що метод добре прогнозує усі мітки.

2.3 Порівняння аналізу результатів тренувань інших підходів з комбінованим підходом

Тепер тренуємо комбінований підхід та порівняємо з іншими підходами. Результати тренування ансамбля:

Accuracy: 71.91%

F1 Score: 71.91

Confusion Matrix:

[[467 59 40 61 66]

[59 495 38 39 48]

[45 32 497 92 41]

[37 19 45 502 35]

[71 47 45 34 479]]

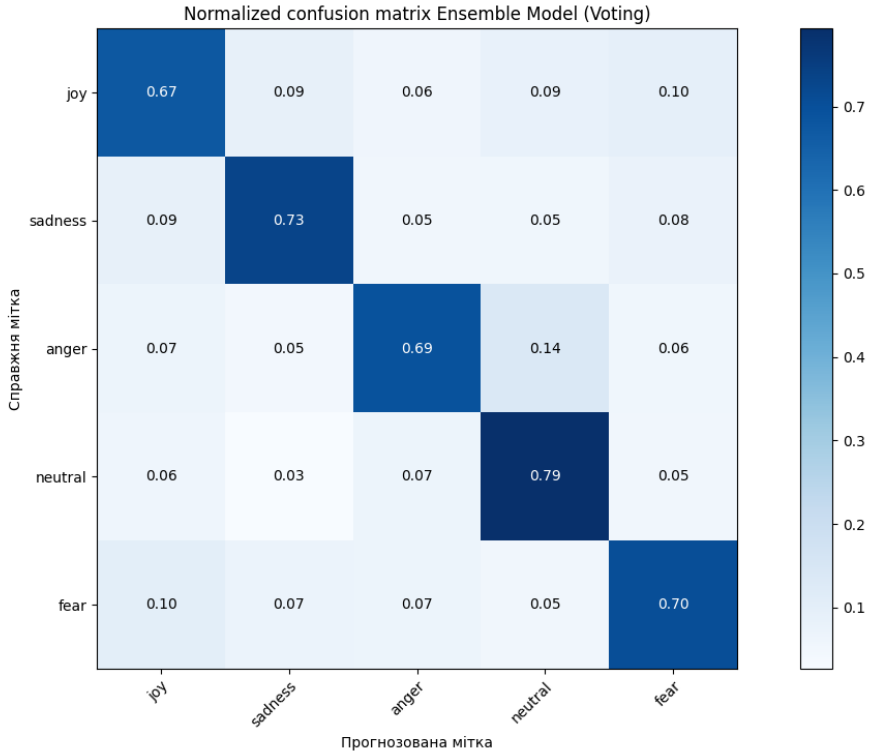

Рисунок 2.2.3 – Нормалізована матриця плутанини комбінованого підходу

Модель комбінованого підходу показує найкращі результати серед усіх моделей, з точністю 71.91% та F1-показником 71.91. Матриця помилок показує розподіл помилок для кожного класу, де числа на головній діагоналі відповідають правильно класифікованим прикладам.

Порівнюючи з іншими моделями, модель комбінованого підходу має кращі результати, ніж Multinomial Naive Bayes, Random Forest, Logistic Regression та Linear SVM. За точністю та F1-показником вона перевищує всі інші моделі.

Оцінка точності (accuracy) є однією з основних метрик для порівняння моделей. Вона вказує на відсоток правильно класифікованих прикладів з усіх прикладів. В даному випадку, модель комбінованого підходу має найвищу точність (71.91%), що свідчить про її ефективність.

Також можна порівняти матриці помилок для кожної моделі, щоб побачити, які класи найбільше помилково класифіковані. Наприклад, в моделі комбінованого підходу, класи 1 і 5 мають більше помилок порівняно з іншими класами.

2.4 Порівняння нашої моделі MH з іншими сервісами

Для порівняння з іншими моделями, я використовую найпопулярніші сервіси у запиті в Google.

1. Перша модель для порівняння від Bytesview

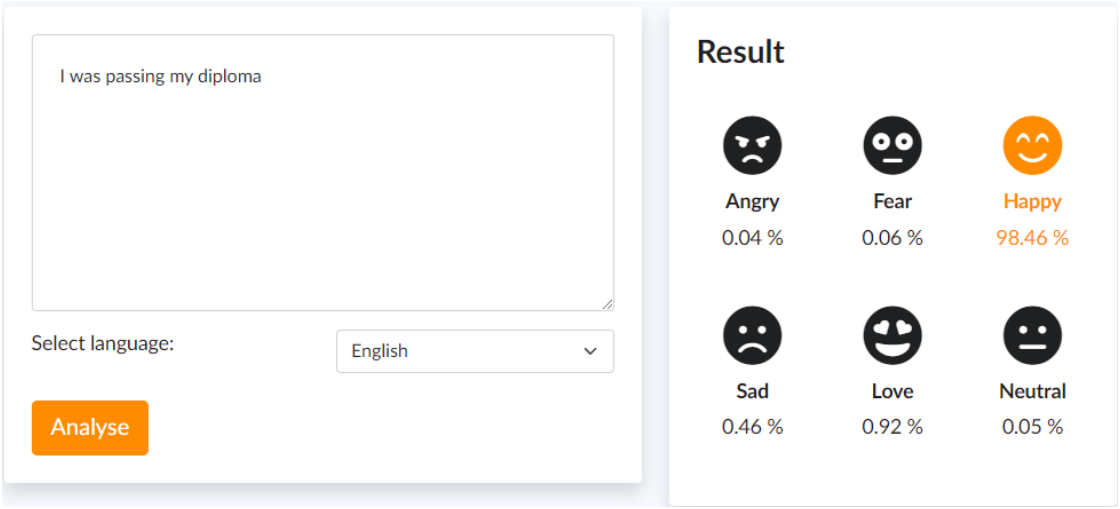

Рисунок 2.4.1 – Порівняння моделі Bytesview

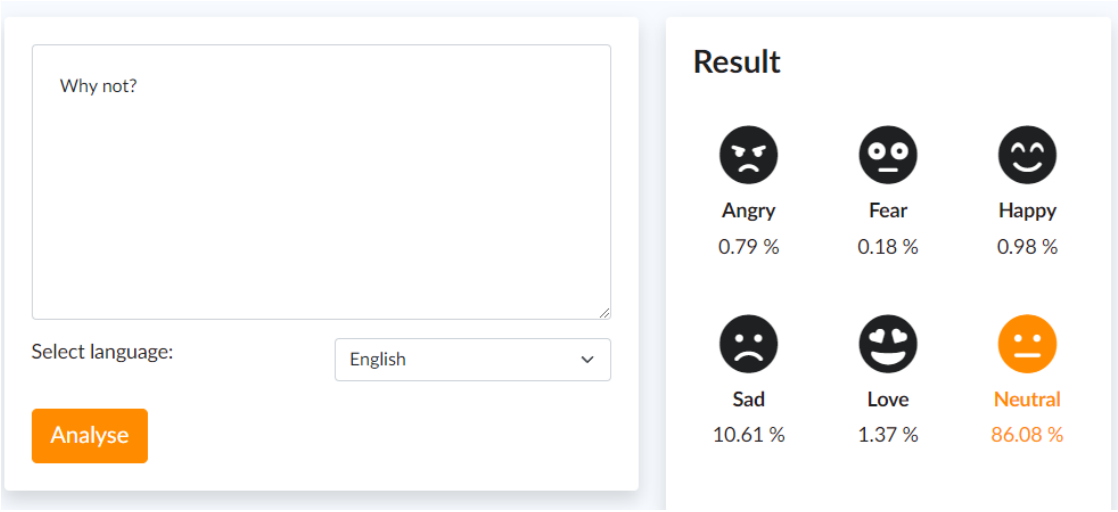

Рисунок 2.4.2 – Порівняння моделі Bytesview

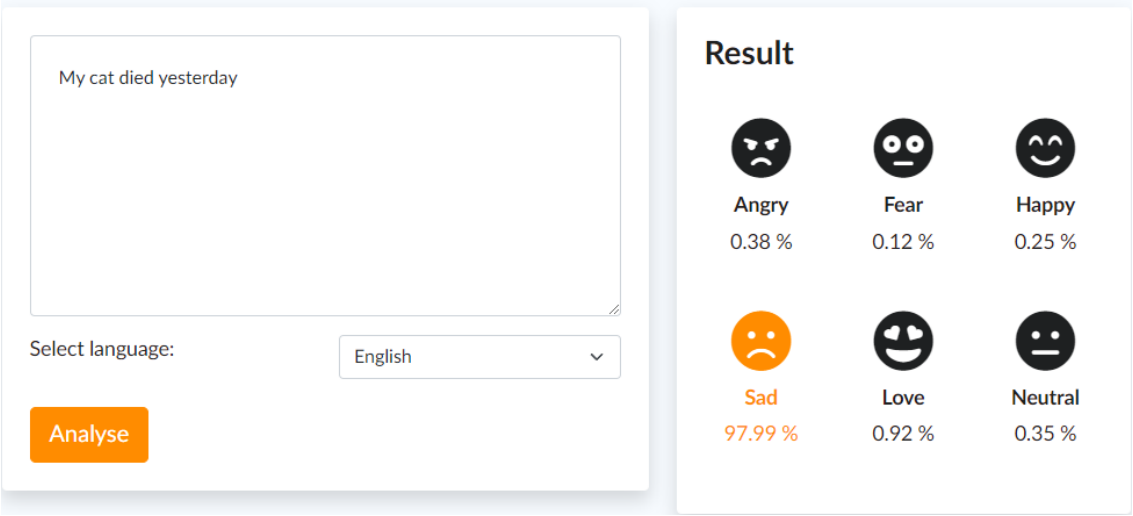

Рисунок 2.4.3 – Порівняння моделі Bytesview

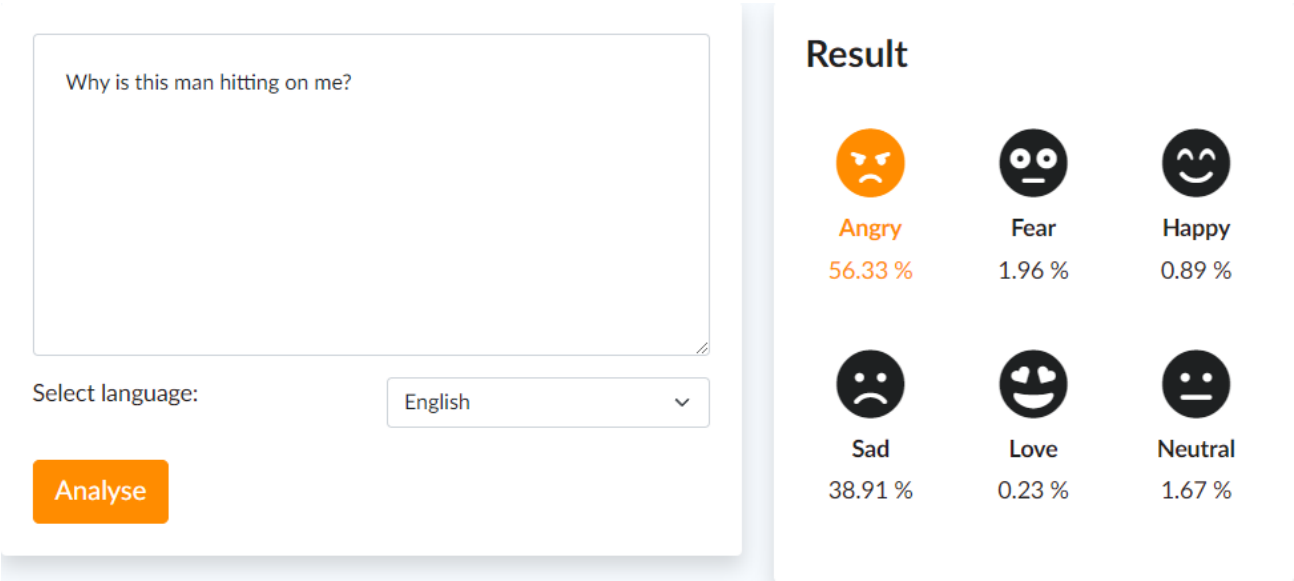

Рисунок 2.4.4 – Порівняння моделі Bytesview

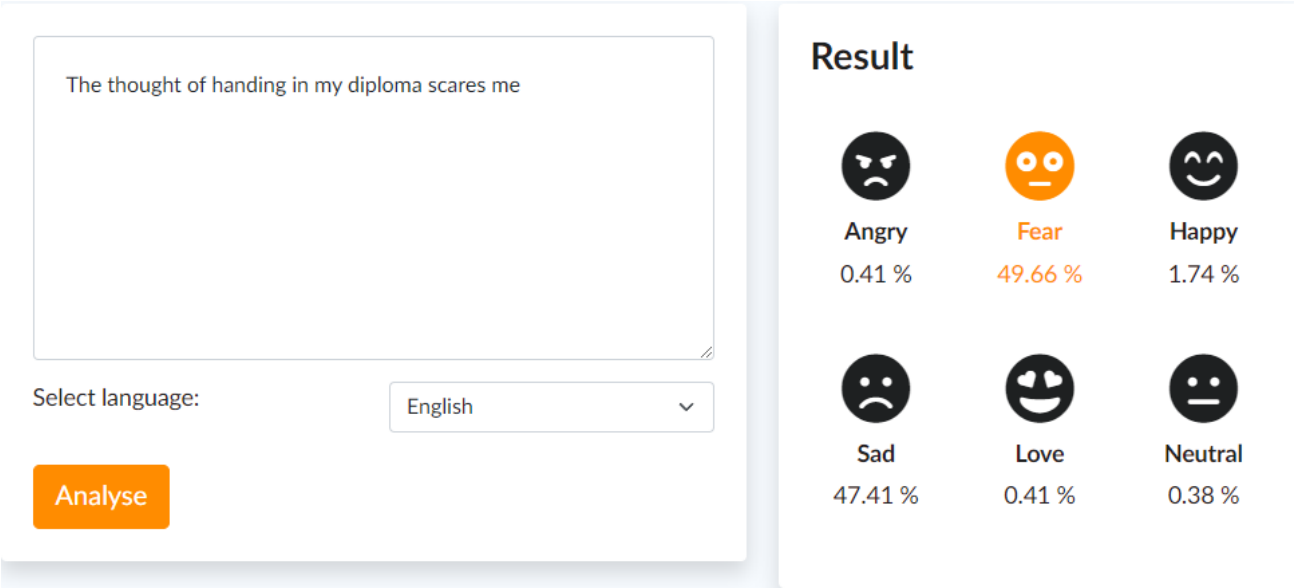

Рисунок 2.4.5 – Порівняння моделі Bytesview

2. Друга модель для порівняння від HuggingFace

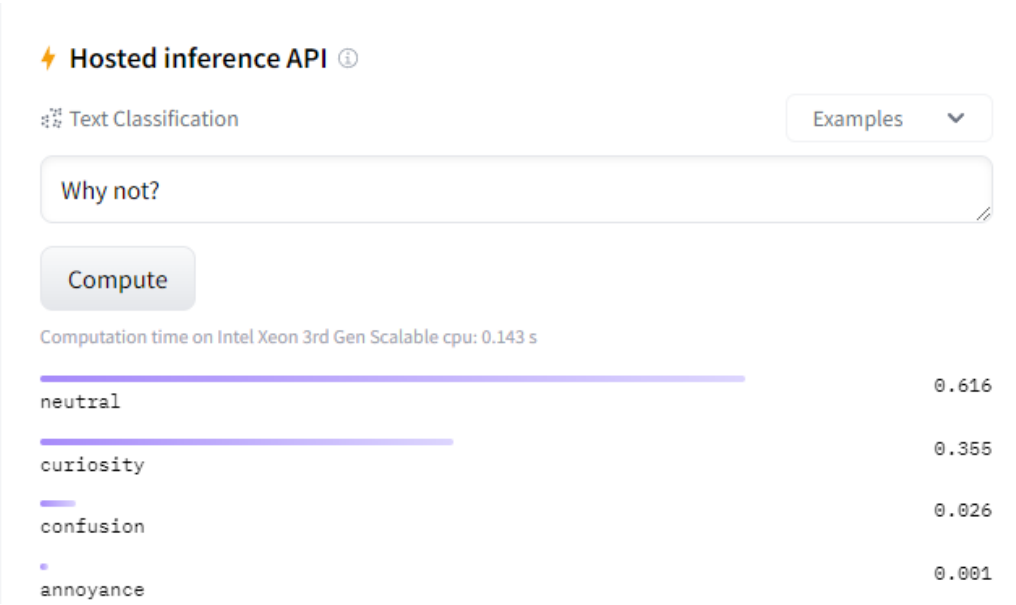

Рисунок 2.4.6 – Порівняння моделі HuggingFace

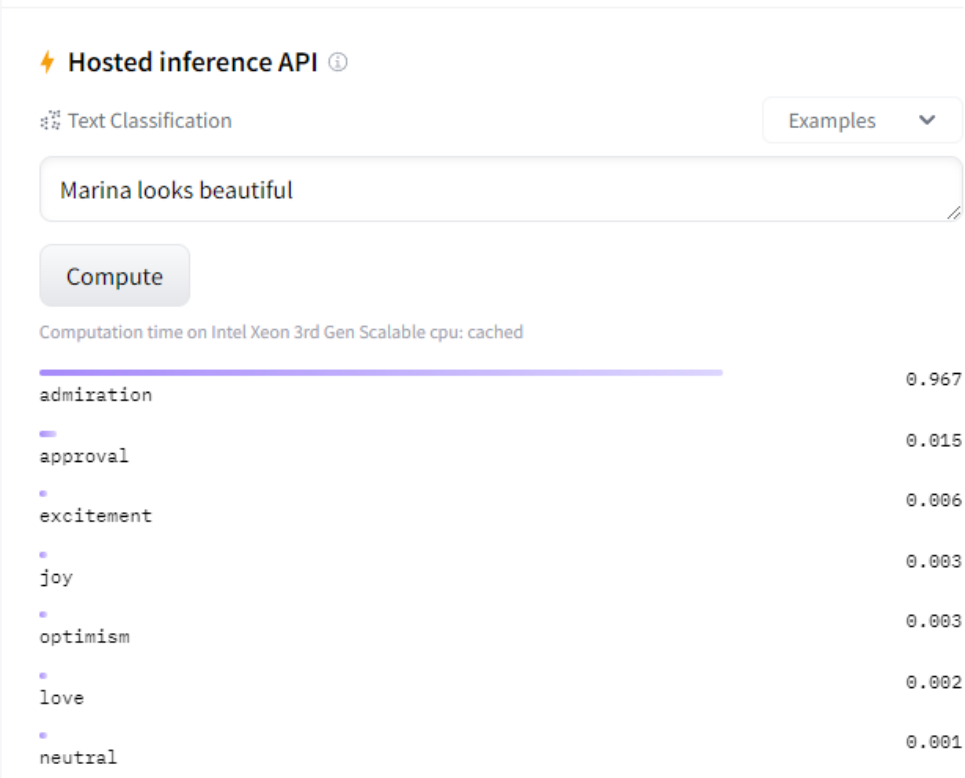

Рисунок 2.4.7 – Порівняння моделі HuggingFace

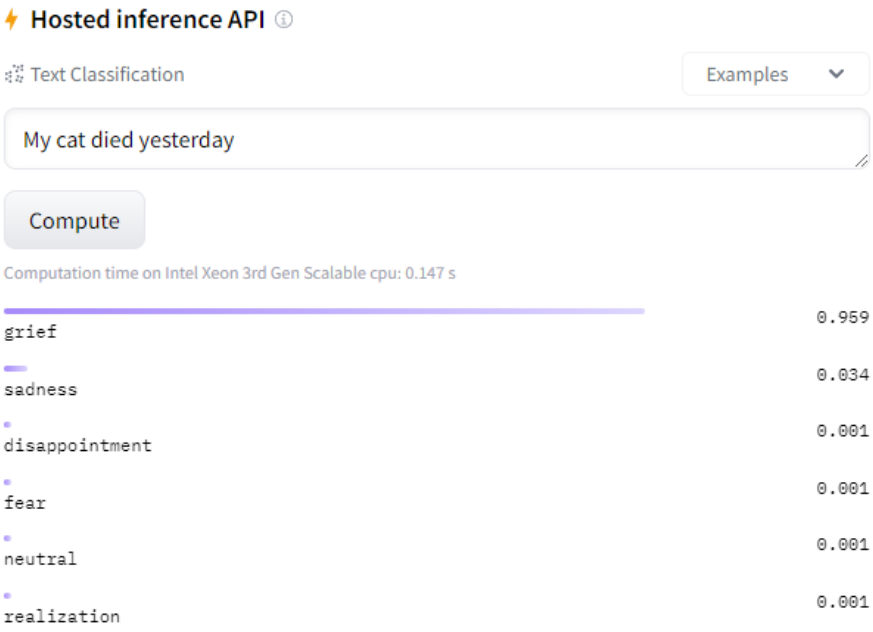

Рисунок 2.4.7 – Порівняння моделі HuggingFace

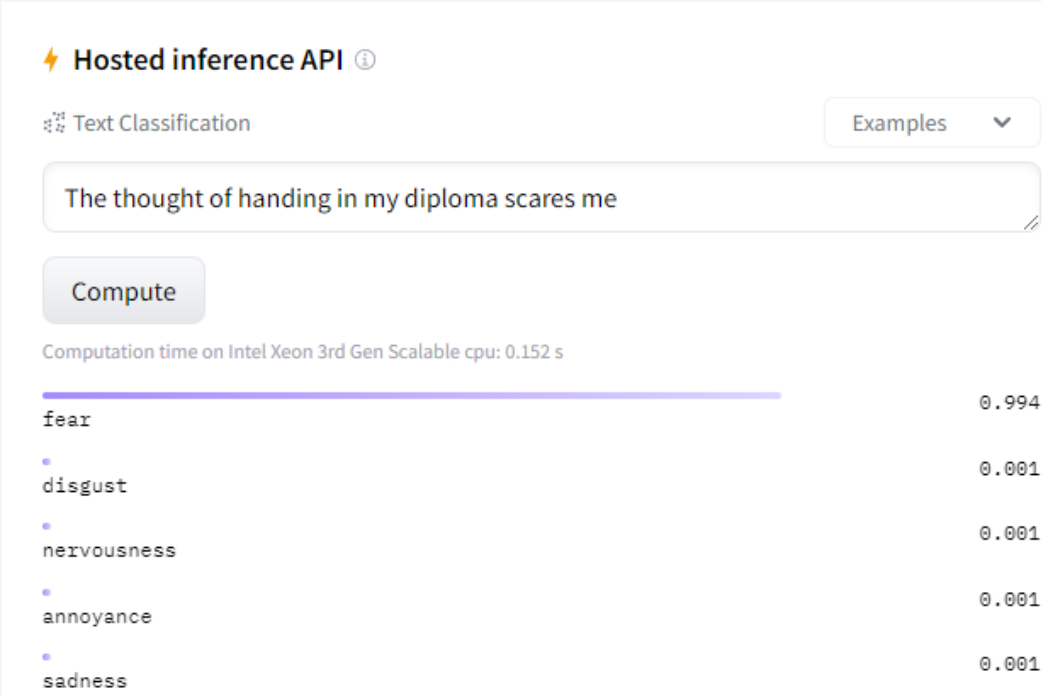

Рисунок 2.4.8 – Порівняння моделі HuggingFace

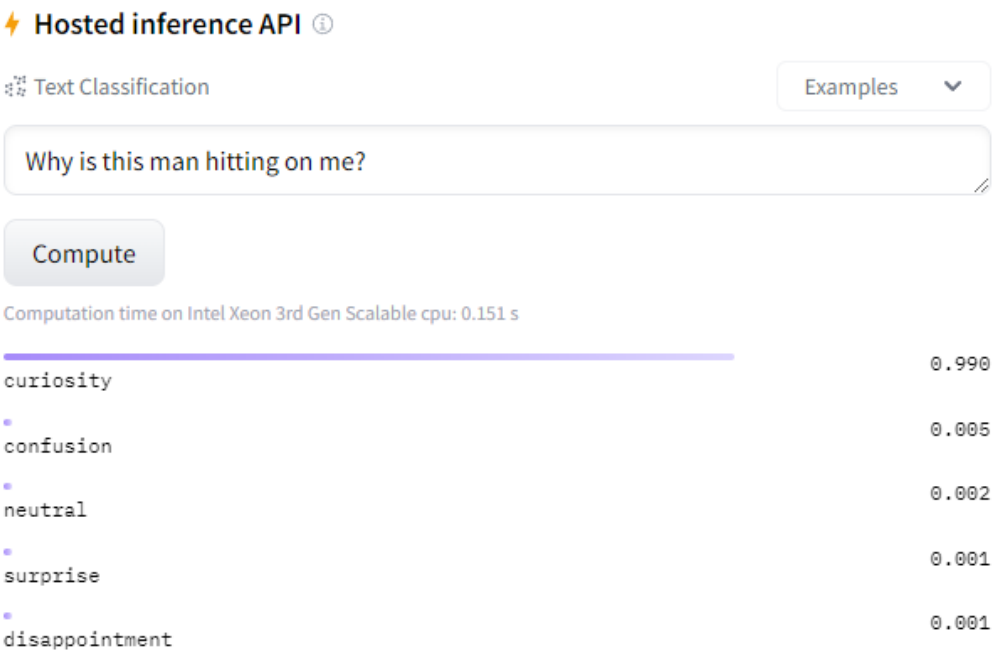

Рисунок 2.4.9 – Порівняння моделі HuggingFace

Отже, наша модель має обмежену кількість класифікацій емоцій і не може здійснювати широкий спектр класифікації емоцій у тексті, у порівнянні з іншими моделями.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Підготовка набору даних

Для підготовки тренувального та тестового наборів даних були використані відкриті ресурси, які містили інформацію у форматі csv. Загальна кількість даних була розділена на дві частини відповідно до відношення 70:30, де 70% даних було призначено для тренування, а 30% - для тестування.

У тренувальному наборі міститься 7934 повідомлення з короткими текстами, що використовуються для тренування моделі, тоді як у тестовому наборі міститься 3393 повідомлення, які використовуються для оцінки ефективності моделі.

Загалом, дані містять інформацію про 5 різних емоцій: радість, смуток, гнів, страх і нейтральний.

```
size of training set: 7934
size of validation set: 3393
Emotion
joy        2326
sadness    2317
anger      2259
neutral    2254
fear       2171
```

Рисунок 3.2.1 – Візуалізація розподіл тренувального та тестового наборів даних з врахуванням окремих емоцій

0	neutral	There are tons of other paintings that I thin...
1	sadness	Yet the dog had grown old and less capable , a...
2	fear	When I get into the tube or the train without ...
3	fear	This last may be a source of considerable disq...
4	anger	She disliked the intimacy he showed towards so...

Рисунок 3.2.2 – Приклад початку даних із набору

Після цього, ми зосереджуємося на обробці набору даних, щоб полегшити його використання для машинного навчання. Для цього застосовується алгоритм обробки даних, який включає наступні кроки:

1. Видалення HTML-розмітки з тексту.
2. Видалення хештегів та імен користувачів, позначених символом "@".
3. Видалення розділових знаків та не-ASCII цифр з тексту.
4. Видалення зайвих пробілів.

Ці кроки допомагають зменшити складність та шум у вихідних даних, а також створюють підготовлені токени, які можуть бути використані для подальшого навчання моделі машинного навчання.

Наступний крок полягає у розділенні тексту на окремі слова або токени (токенізація) та зменшенні слів до їх основи або кореня (стемінг). Для цього використовується бібліотека NLTK (Natural Language Toolkit). Функція word_tokenize() з модуля nltk використовується для розбиття тексту на окремі слова або токени. Наприклад, якщо початковим текстом є речення "I love cats", результатом токенізації буде список ['I', 'love', 'cats'].

Після токенізації застосовується метод стемінгу, щоб зменшити слова до їх основи або кореня. У даному випадку використовується алгоритм Портера (Porter stemming algorithm), який реалізований у класі PorterStemmer() з

бібліотеки NLTK. Цей алгоритм застосовує ряд правил для визначення кореня слів. Наприклад, слово "cats" буде зменшене до "cat", а слово "loved" - до "love".

Отже, після виконання цього фрагмента коду змінна stem_data буде містити список стемованих токенів, тобто коренів слів, які були отримані з вихідного тексту data.

Після проведення токенізації та стемінгу, наступним кроком є векторизація тексту за допомогою техніки терм-частот (TF) - зворотної частоти документу (IDF).

Для цього можна використовувати нашу функцію preprocess_and_tokenize для токенізації тексту. Потім обчислюється TF, що представляє відношення кількості повторень слова в документі до загальної кількості слів у документі. Це дає нам інформацію про важливість кожного слова у конкретному документі.

Далі обчислюється IDF, що враховує частоту використання слова у всіх документах колекції. IDF розраховується за формулою: логарифм відношення загальної кількості документів до кількості документів, що містять дане слово. Це дозволяє виділити слова, які володіють більшою інформативністю через їх низьку частоту використання у всій колекції документів.

Таким чином, в результаті векторизації тексту ми отримуємо числові представлення слів у вигляді значень TF-IDF, які вказують на їх важливість у кожному документі з колекції.

Наступним кроком є використання функції vect.fit_transform(data.Text), яка навчає модель векторизації (у цьому випадку об'єкта vect) на повному корпусі текстів data.Text. Це означає, що модель аналізує особливості тексту у цьому корпусі і створює числові векторні представлення для кожного документа, використовуючи певну техніку векторизації, наприклад, TF-IDF.

Після навчання моделі, застосовується метод vect.transform(X_train) та vect.transform(X_test), де навчена модель vect використовується для

перетворення тренувального та тестового наборів даних у вектори. Це означає, що для кожного документа з тренувального та тестового наборів даних будуть створені векторні представлення, використовуючи вивчені особливості моделі векторизації.

Отже, після виконання цих операцій, змінна X_train_vect буде містити векторні представлення для тренувальних даних, а X_test_vect - для тестових даних. Ці вектори можна використовувати як вхідні дані для подальшого навчання моделей машинного.

3.2 Реалізація функцій для оцінки підходів тренування

Для оцінки методів нам потрібно виконати деякі функції. Спершу ми повинні реалізувати функцію побудови матриці помилок (confusion matrix) під назвою "plot_confusion_matrix". Ця функція використовується для створення графічного представлення матриці помилок для класифікаційних моделей. Вона приймає реальні мітки (y_true) та передбачені мітки (y_pred) і створює графік, на якому вертикальна ось відображає реальні мітки, а горизонтальна ось - передбачені мітки.

На графіку використовується кольорова шкала, де кожна клітинка представляє відповідне значення матриці помилок. Значення помилок можуть бути нормалізовані або ненормалізовані, залежно від встановленого параметра "normalize". Також на графіку відображаються значення помилок в кожній клітинці. Якщо використовується нормалізація, значення помилок відображаються як десяткові числа, і текст стає білим, якщо значення помилки перевищує половину максимального значення. В іншому випадку текст залишається чорним.

Далі ми використаємо функцію "accuracy_score(y_test, y_nb_pred)" для обчислення точності, яка визначає відношення правильно передбачених міток до загальної кількості міток у тестовому наборі (y_test).
Також ми використаємо функцію "f1_score(y_test, y_rf_pred, average='micro')" для обчислення F1-оцінки, яка є комбінацією точності та повноти (recall) передбачення. Параметр "average='micro'" вказує, що мікро-усереднення має бути використане для обчислення F1-оцінки.

Ми використовуємо функції оцінок з метою визначити найкращий метод для визначення емоцій у тексті. Ці функції допомагають нам оцінити ефективність різних методів і порівняти їх результати. За допомогою цих оцінок ми можемо встановити точність передбачень, що вказує на те, наскільки правильно модель класифікує емоції у тексті. Також ми можемо оцінити збалансованість та точність класифікації, що дозволяє нам визначити найбільш ефективний метод для наших потреб у визначенні емоцій.

3.3 Реалізація моделі Multinomial Naive Bayes

Модель Multinomial Naive Bayes є однією з найпростіших та широко використовуваних моделей для класифікації текстових даних. У цій моделі використовується наївний баєсовський підхід з припущенням мультиноміального розподілу. Перед початком роботи модель навчається на тренувальних даних шляхом використання функції fit(X_train_vect, y_train), де X_train_vect - векторизовані тренувальні дані, а y_train - мітки класів для тренувальних даних. Після навчання модель може прогнозувати класи для тестових даних за допомогою функції predict(X_test_vect), де X_test_vect - векторизовані тестові дані. Результати класифікації оцінюються за допомогою метрик точності (Accuracy), F1-показника (F1 Score) та матриці плутанини (Confusion Matrix).

3.4 Реалізація моделі Random Forest

Модель Random Forest є ансамблевою моделлю, яка базується на рішеннях дерев рішень. У цій моделі використовується декілька дерев рішень, які працюють разом для зроблення прогнозів. У данному випадку, модель Random Forest складається з 100 дерев рішень (n_estimators=100). Після навчання модель може прогнозувати класи для тестових даних так само, як і у моделі Multinomial Naive Bayes. Результати класифікації оцінюються тими ж метриками, що й у попередній моделі.

3.5 Реалізація моделі Logistic Regression

Модель Logistic Regression є лінійною моделлю для класифікації. У цій моделі використовується логістична функція для моделювання ймовірності належності до певного класу. Після навчання модель може прогнозувати класи для тестових даних так само, як і у попередніх моделях. Результати класифікації оцінюються за тими ж метриками.

3.6 Реалізація моделі Linear SVM

Модель Linear SVM (Support Vector Machine) є моделлю, яка шукає гіперплощину, що найкраще розділяє дані різних класів. У данному випадку використовується лінійне ядро (linear kernel) для розв'язання задачі класифікації. Після навчання модель може прогнозувати класи для тестових даних так само, як і у попередніх моделях. Результати класифікації оцінюються за тими ж метриками.

3.7 Реалізація моделі комбінованого класифікатора з м'яким голосуванням

Для покращення результатів класифікації, ми будуємо комбінований класифікатор з використанням м'якого голосування. У даному випадку, ми використовуємо чотири моделі: Multinomial Naive Bayes, Random Forest, Logistic Regression та Linear SVM. Комбінований класифікатор (ensemble_model) будується за допомогою класу VotingClassifier, який приймає список оцінювачів (estimators) та тип голосування (voting). Оцінювачі вказуються у форматі кортежів (ім'я, модель). В даному випадку, тип голосування встановлено як "soft", що означає, що прогнози моделей враховують ймовірності належності до кожного класу. Після навчання комбінованого класифікатора, ми можемо прогнозувати класи для тестових даних та оцінювати результати за тими ж метриками, що й у попередніх моделях.

В результаті порівняння та комбінування різних моделей класифікації, ми можемо оцінити їх ефективність та вибрати найкращу модель для нашої задачі. Також комбінований підхід може дати кращі результати, ніж окремі моделі, оскільки враховує різні підходи та переваги кожної моделі.

3.8 Результати тестування моделі комбінованого класифікатора

Для проведення тестування ми використовуємо власні дані, які складаються з двох речень для кожної емоції, що не були враховані під час навчання нашої машини. Після навчання моделі машинного навчання найкращий результат зберігається у файлі "model.sav", який можна використовувати для подальшого використання моделі без необхідності повторного навчання.

Оскільки ми вибрали ансамблеву модель, саме цю модель було використано для проведення тестування.

Результатом тестування є правильне призначення емоцій кожному реченню.

```
Test1: Marina looks beautiful ['joy']
Test2: I was passing my diploma ['joy']
Test3: Why not? ['neutral']
Test4: Hello, doc ['neutral']
Test5: My cat died yesterday ['sadness']
Test6: She was sad ['sadness']
Test7: The thought of handing in my diploma scares me ['fear']
Test8: I hate working at night ['fear']
Test9: He made me angry because he didnt do his job ['anger']
Test10: Why is this man hitting on me? ['anger']
```

Рисунок 4.1 – Результат роботи Linear SVM

ВИСНОВОК

Головною метою цього дослідження було створення комбінованого підходу, який здатний сам класифікувати емоції в тексті за допомогою попередньо зазначених класів у наборі даних. Для досягнення цієї мети було виконано наступні кроки:

- Вибір методів класифікації для розв'язання задачі.
- Вибір відповідних інструментів для реалізації обраних методів і оволодіння ними.
- Визначення та реалізація необхідних кроків попередньої обробки даних.
- Тренування моделі машинного навчання.
- Створення комбінованої моделі м’якого голосування
- Порівняння результатів різних моделей.

Завдяки чіткому виконанню цих завдань, які були поставлені на початку дослідження, нам вдалося отримати працюючу програму, яка досить точно класифікує повідомлення за емоційними категоріями.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. A Gentle Introduction to Scikit-Learn: A Python Machine Learning Library by Jason Brownlee on April 16, 2014 in Python Machine Learning. - Режим доступу до статті: https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/
2. How To Prepare Your Data For Machine Learning in Python with Scikit-Learn by Jason Brownlee on May 18, 2016 in Python Machine Learning. - Режим доступу до статті: https://machinelearningmastery.com/prepare-data-machine-learning-python-scikit-learn/
3. Learn classification algorithms using Python and scikit-learn. By Samaya Madhavan, Mark Sturdevant. Published December 3, 2019. - Режим доступу до статті: https://developer.ibm.com/tutorials/learn-classification-algorithms-using-python-and-scikit-learn/

   F1 Score in Machine Learning: Intro & Calculation. December 16, 2022 by Rohit Kundu. - Режим доступу до статті: https://www.v7labs.com/blog/f1-score-guide#:~:text=for%20Machine%20Learning-,What%20is%20F1%20score%3F,prediction%20across%20the%20entire%20dataset.
4. Scikit-learn. - Режим доступу: https://scikit-learn.org/stable/
5. NLTK. - Режим доступу: https://www.nltk.org/api/nltk.corpus.html
6. PYTHON. - Режим доступу: https://www.python.org/
7. Перша модель для порівняння від BytesView. - Режим доступу: https://www.bytesview.com/emotion-analysis

8. Друга модель для порівняння від HuggingFace. - Режим доступу: https://huggingface.co/arpanghoshal/EmoRoBERTa
9. Ensemble learning using the Voting Classifier. Eryk Lewinson Published in Level Up Coding Feb 9, 2020
   - Режим доступу до статті: https://levelup.gitconnected.com/ensemble-learning-using-the-voting-classifier-a28d450be64d

Додаток А

Фрагмент коду навчання комбінованого моделі та її оцінки:

```
# Побудова комбінованого класифікатора з м'ягким голосуванням
ensemble_model = VotingClassifier(
    estimators=[
        ('nb', nb),
        ('rf', rf),
        ('log', log),
        ('svc', svc)
    ],
    voting='soft'  # Голосування за більшість
)

# Навчання комбінованого класифікатора
ensemble_model.fit(X_train_vect, y_train)

# Прогнозування за допомогою комбінованого класифікатора
y_pred_ensemble = ensemble_model.predict(X_test_vect)

# Оцінка результатів комбінованого підходу
print("Ensemble Model (Voting):")
print("Accuracy: {:.2f}%".format(accuracy_score(y_test, y_pred_ensemble) * 100))
print("F1 Score: {:.2f}".format(f1_score(y_test, y_pred_ensemble, average='micro') *
100))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_ensemble))
plot_confusion_matrix(y_test, y_pred_ensemble, classes=class_names,
normalize=True, title='Normalized confusion matrix Ensemble Model (Voting)')
plt.show()
```