ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ, УПРАВЛІННЯ, ПРАВА ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Пояснювальна записка

до кваліфікаційної роботи на здобуття ступеня вищої освіти магістр

на тему: **«Технологія класифікації тексту на основі нейронної мережі»**

Виконав: здобувач вищої освіти
за освітньо-професійною програмою
Інформаційні управляючі системи та технології спеціальності
126 Інформаційні системи та технології ступеня вищої освіти магістр
групи 126ІСТмд_22
Науменко С. С.
Керівник: Слюсар В. І.
Рецензент: Муравльов В. В.

Полтава – 2023 року

ВСТУП

Актуальність теми кваліфікаційної роботи підтверджується необхідністю класифікації тексту за допомогою штучного інтелекту. У сучасному світі нейронні мережі використовуються у багатьох галузях людської діяльності. Вони використовуються у всьому, від розпізнавання обличчя на смартфонах до (але не обмежуючись) аналізу великих даних. Основні функції, які виконують нейронні мережі, включають: стиснення та розпакування даних; класифікація та розпізнавання образів; екстраполяція даних; прогнозування; статистика та ін. Велика кількість публікацій досліджує різні аспекти створення архітектури нейронні мережі та застосування різних методів обробки даних. Важливу роль у реалізації цих методів відіграє програмне забезпечення, що використовується для розробки та навчання нейронні мережі. Найбільш розвинений функціонал для роботи з нейронними мережами та відповідні бібліотеки обробки даних мають фреймворки, що базуються на мові програмування Python, такі як Keras, TensorFlow, Pytorch та інші. Однак, інтеграція нейронних мереж, розроблених на основі Python, з вебдодатками може бути складною та неоптимальною. Це може призвести до сповільнення роботи нейронних мереж та висувати високі вимоги до апаратного забезпечення, на якому ці нейронні мережі розгортаються, для забезпечення їх функціонування у реальному масштабі часу. Одним з основних недоліків використання інтерпретатора Python є обмежена сумісність між різними версіями відповідних бібліотек, що значно скорочує термін служби розробленого програмного забезпечення та вимагає значних ресурсів та часу для підтримки роботи вебдодатків, які використовують нейронні мережі. Використання фреймворку WEKA для класифікації тексту може принести значні переваги для організацій, підприємств та установ, які займаються обробкою та аналізом великих обсягів текстових даних. При цьому, однією з проблем використання WEKA є складність використання фреймворку для навчання моделей на великих наборах даних. Це може призвести до уповільнення процесу навчання та зростання вимог

до ресурсів обчислення. Враховуючи ці проблеми, актуальність досліджень у цій області є очевидною.

Зв'язок роботи з науковими програмами, темами. Робота відповідає дослідженням в рамках науково-дослідної роботи «Управління стратегією інноваційного розвитку підприємств в контексті підвищення їх конкурентоспроможності на аграрному ринку, сталого розвитку та забезпечення продовольчої безпеки держави» (2021 р.), що фінансувалась господарськими договорами із замовниками, Концепції розвитку штучного інтелекту в Україні (розпорядження Кабінету Міністрів України № 1787-р від 29.12.2021), тематиці досліджень навчально-дослідної лабораторії інтелектуальних систем, комп'ютерних мереж та інтернет речей кафедри інформаційних систем та технологій Полтавського державного аграрного університету.

Метою кваліфікаційної роботи є класифікація текстів за допомогою нейронні мережі, що розроблена на мові програмування Java. Така нейронна мережа буде здатна розрізняти позитивний чи негативний відгук надійшов від відвідувача.

Завданнями кваліфікаційної роботи є:

– визначити особливості класифікації тексту;

– розробити технологію класифікації тексту;

– сформувати рекомендації щодо використання технології класифікації тексту на основі нейронної мережі;

– виконати техніко-економічне обґрунтування прийнятих рішень.

Об'єктом дослідження є процес класифікації тексту за допомогою нейронних мереж.

Предметом дослідження є залежність точності класифікації тексту від параметрів класифікатору.

Методи досліджень: аналітичний, дедуктивний, інформаційно-пошуковий, методи синтезу та навчання нейронних класифікації тексту, робота з фреймворком WEKA.

Інформаційна база базується на ресурсах з даними про методи та класифікації тексту, інструментарій для розробки та дослідження нейронних мереж на основі мови Java.

Елементи наукової новизни роботи полягають у створені моделі класифікатору тексту.

Практична значущість роботи полягає в розробці рекомендацій щодо використання технології класифікації тексту на основі нейронної мережі, які можуть бути використані для подальших досліджень за даною тематикою та при проектуванні мобільних додатків.

Апробація результатів відбувалася в рамках в V-ої Міжнародної студентської конференції «Теоретичне та практичне застосування результатів сучасної науки» (жовтень 2023 р., м. Рівне), III-ої Міжнародної студентської конференції «Міждисциплінарні наукові дослідження та перспективи їх розвитку» (листопад 2023 р., м. Дніпро).

За результатами досліджень здійснено 2 публікації тез доповідей.

Структура кваліфікаційної роботи має логічний зв'язок із задачами досліджень, містить вступ, три розділи основної частини, висновки, список використаних джерел, додатки. Загальний обсяг пояснювальної записки кваліфікаційної роботи складає роботи складає 73 сторінки формату А4. Вона містить 35 рисунків і 4 таблиці.

РОЗДІЛ 1
АНАЛІЗ ОСОБЛИВОСТЕЙ КЛАСИФІКАЦІЇ ТЕКСТУ

1.1 Передумови виникнення інтелектуального аналізу даних

Дані. Сучасний світ не просто повністю ґрунтується на даних, він живе даними. Дані присутні всюди, вони проникають у кожен аспект людського життя, і їхня кількість незліченна. Під даними можна також розуміти, у деякому значенні інформацію, але це більше ніж просто інформація. Кількість даних у світі, у людському житті, не можливо точно вирахувати, через те, що кількість інтернет користувачів невпинно зростає, як показано на рисунку 1.1. Згідно звіту Global Digital 2023 [1, с. 30] кількість інтернет користувачів у 2023 році зросла на приблизно 140 мільйонів. Кожен з цих користувачів поширює інформацію, дані, дані про себе, свою сім'ю, свої вподобання, думки, медіа, будь що. Це створює величезний потік даних, який впливає на всі сфери людського життя, від освіти до економіки.

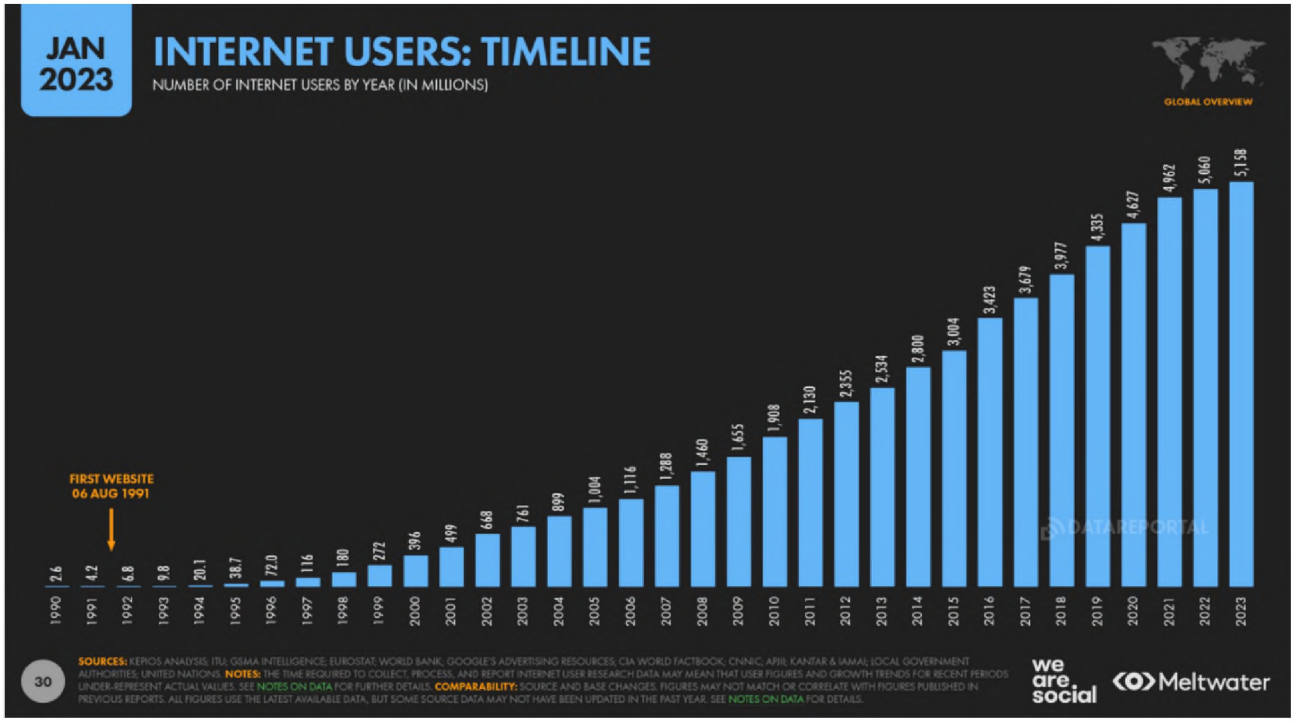


Рисунок 1.1 – Графік зростання кількості інтернет користувачів [1, с. 30]

Персональні комп'ютери не просто дозволяють надто легко зберігати речі, які користувачі звикли зберігати, вони спрощують цей процес і дозволяють накопичувати інформацію, замість того щоб позбутися її. З відносно недорогими накопичувачами, які здатні вміщувати багато інформації, дуже легко відкладати рішення, що з усіма цими даними робити – замість активних дій, просто купувати інший диск і все.

Усюди електроніка записує прийняті людьми рішення, їхній вибір у супермаркеті, фінансові звички, поточні доходи або випадкові витрати. Цей процес створює цифровий відбиток людського життя, який може бути використаний для аналізу людських звичок, вподобань та поведінки.

Для збору та зберігання усієї цієї інформації використовуються не лише тільки персональні комп'ютери. На рисунку 1.2 можна побачити статистику використання інтернету усіма пристроями згідно звіту Global Digital 2023 [1, с. 25].

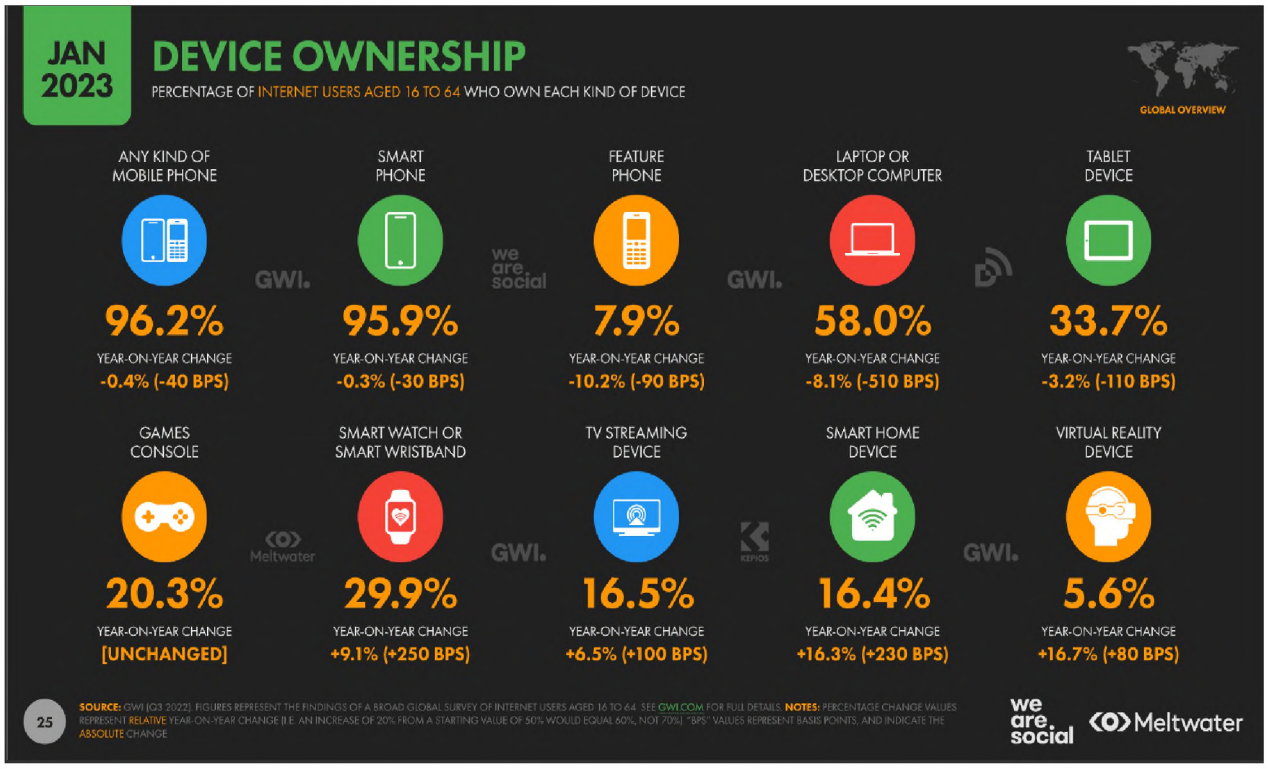


Рисунок 1.2 – Статистика використання інтернету різними пристроями [1, с. 25].

Кожен користувач всесвітньої мережі не просто лишає свій шлях у ній, він створює цифровий відбиток своєї присутності. Кожен рух пальця створює запис у

базі даних, відображаючи людські вибори та вподобання. Незважаючи на те, що всесвітня мережа переповнена інформацією – кожен вибір записується. І всі ці записи, які всього-на-всього зберігають лише особистий вибір, мають величезне значення у світі торгівлі та промисловості. Оскільки обсяг даних збільшується, частка того, що з цих даних може бути зрозумілим людині, зменшується у геометричній послідовності, адже можливості людського мозку не безмежні і на відміну від носіїв інформації не можуть бути збільшені за рахунок вертикального масштабування [2, с. 70]. Це викликає потребу в розробці нових методів та технологій для обробки та аналізу цих величезних обсягів даних.

Усі ці дані містять приховану потенційно корисну інформацію, і основна мета аналізу цих даних – це видобуток цієї інформації і використання її із користю. Цей процес видобутку може включати в себе все, від виявлення тенденцій та шаблонів до прогнозування майбутніх подій. Потрібно враховувати що, ці дані також можуть бути використані із злим умислом, що може призвести до фінансових втрат [3, с. 44]. Це підкреслює важливість захисту даних, а також необхідність розуміння того, як і де дані користувачів використовуються.

Розглянемо один із найпоширеніших прикладів того, як збираються, обробляються та використовуються дані.

Супермаркети використовують різноманітні методи для збору даних про своїх покупців. Один з найпоширеніших способів – це через програми лояльності, які надають покупцям знижки або бонуси за покупки. Коли покупці реєструються в програмі лояльності, вони надають свої контактні дані, такі як електронна пошта або номер телефону, а також інформацію про свої вподобання та інтереси.

Кожен раз, коли покупець робить покупку і використовує свою картку лояльності, супермаркет збирає дані про те, що він купує, скільки він витрачає і коли він робить покупки. Ця інформація допомагає супермаркету створювати профіль покупця, який вони можуть використовувати для персоналізації маркетингових пропозицій [4, с. 127].

Наприклад, якщо супермаркет бачить, що покупець регулярно купує певний бренд молока, співробітники супермаркету можуть надіслати покупцю купон на

знижку на цей продукт. Або, якщо вони бачать, що покупець часто робить покупки в певний день тижня, вони можуть надіслати йому спеціальні пропозиції, які діють лише в цей день.

Ці дані також можуть бути використані для прогнозування майбутньої поведінки покупців. Наприклад, якщо супермаркет бачить, що покупець купує певні продукти разом (наприклад, ковбасу і сир для бутербродів), вони можуть використовувати цю інформацію для створення «кошика покупок», який вони можуть пропонувати покупцю наступного разу, коли він зробить покупку [5, с. 54].

Використання даних про покупців дозволяє супермаркетам не тільки збільшити свої продажі, але і покращити досвід покупок для своїх клієнтів, надаючи їм більш персоналізовані та релевантні пропозиції.

Ілюстрований приклад таких даних можна побачити на рисунку 1.3 який відображає дані щодо статі онлайн покупців у продуктових крамницях, згідно звіту Global Digital 2023 [1, с. 356].

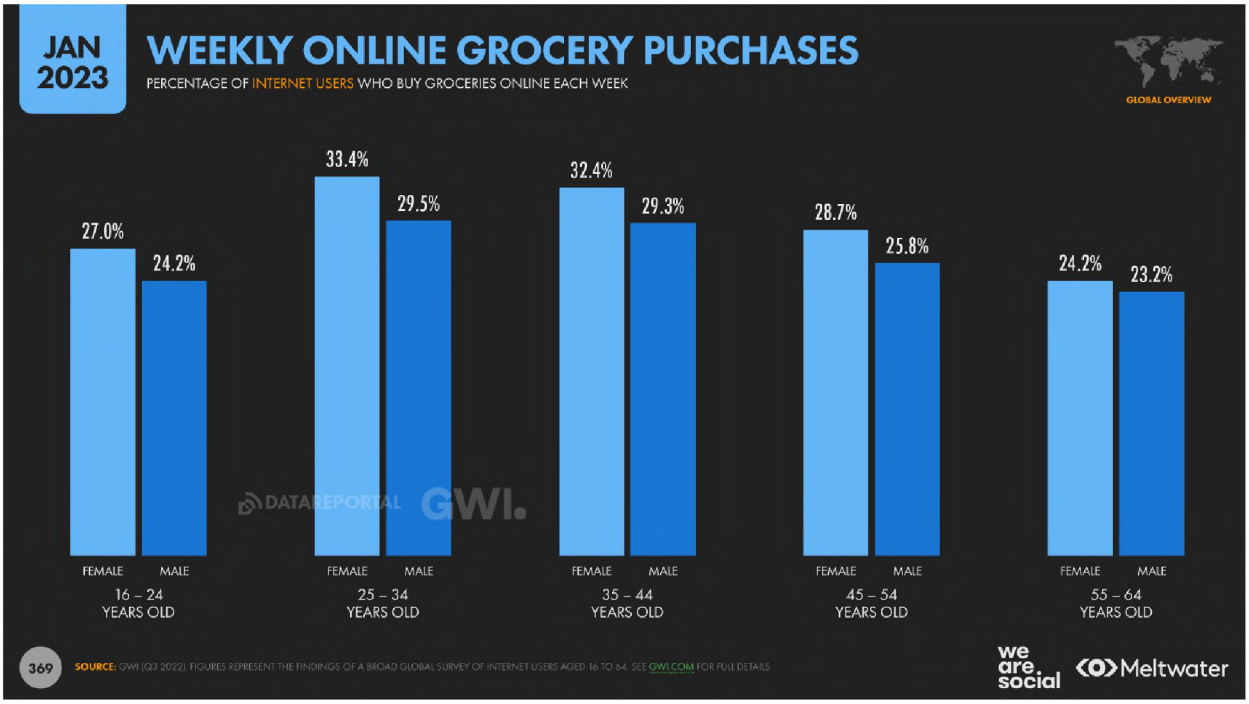


Рисунок 1.3 – Аналіз статі покупців у продуктових магазинах [1, с. 356].

Звісно, для того щоб обробити дані кожної людини, яка робить покупки у супермаркеті, незалежно від того, чи робить вона це особисто в магазині, чи

замовляє товари через інтернет, потрібна величезна кількість людських ресурсів. Однак залучення такої кількості людей для обробки цих даних не є практичним, а навіть якщо б і була така можливість, отримані дані вже втратять свою актуальність і об'єктивність до того часу, як їх оброблять [6, с. 48].

Саме тому виникла потреба в автоматизації аналізу цих даних. Але мова йде не просто про механічну автоматизацію, а про автоматизацію, яка включає логічний аналіз і здатна навчатися та передбачати результати на основі отриманих даних.

Алгоритми класифікації тексту, що досліджуються у цій кваліфікаційній роботі, це і є той самий інтелектуальний аналіз даних, що навчається і що можна розглядати як спосіб дізнатися, які фактори були враховані при прийнятті того чи іншого рішення та наступному його передбаченні. Це включає в себе вивчення шаблонів та тенденцій в даних, виявлення взаємозв'язків між різними факторами, а також розробку моделей, які можуть передбачити майбутнє поведінку на основі цих виявлених шаблонів і тенденцій.

Інтелектуальний аналіз даних є однією із передових технологій [7, с. 69], призначених для обробки великих обсягів даних, і він знаходиться на передньому краї нових бізнес-технологій. Згідно з приблизною оцінкою незалежних експертів, кількість даних, що зберігаються у світових базах даних, подвоюється кожні 20 місяців.

Це надзвичайно швидкий темп зростання, який вимагає розробки нових методів та технологій для ефективного керування та аналізу цих даних. У кількісному сенсі, існує також можливість, зіставити темпи зростання із якісним. Так як потік даних, так само як і можливості персональних комп'ютерів, які можуть шукати та аналізувати, зростають, можливості інтелектуального аналізу даних також зростають.

Розумно проаналізовані дані є цінним ресурсом [8, с. 52]. Вони можуть відкрити нові горизонти для наукових досліджень, інновацій та розуміння світу навколо нас. В комерційних умовах, вони можуть призвести до нових ідей, що можуть дати конкурентні переваги, допомогти в оптимізації процесів, покращити

взаємодію з клієнтами та навіть відкрити нові ринки. Це відкриває нові можливості для бізнесу та науки, а також створює нові виклики, пов'язані з захистом та безпекою даних.

Інтелектуальний аналіз даних представляє собою вирішення проблем шляхом аналізу даних, які вже є в базах даних. Це включає в себе використання розширених алгоритмів та методів для виявлення шаблонів та тенденцій в цих даних, що можуть допоможуть краще зрозуміти та вирішити проблеми.

Інтелектуальний аналіз даних – це технологія, що розвивається, для отримання знань із даних [9, с. 28]. Це технологія, яку багато людей починають сприймати серйозно з ідеєю, що закономірності в даних можна шукати автоматично, ідентифікувати перевірено та використовувати для прогнозування.

Розглянемо старий, але все ще актуальний приклад: вирішення проблеми зниження лояльності клієнтів на висококонкурентному ринку. База даних вибору клієнтів, разом із профілями клієнтів, може містити ключ до вирішення цієї проблеми.

Можна проаналізувати поведінку колишніх клієнтів, щоб визначити відмінні характеристики тих, хто, швидше за все, перейде до конкурентів, і тих, хто, ймовірно, залишиться лояльним. Один раз коли ці характеристики визначено, їх можна використовувати для ідентифікації поточних клієнтів, які, ймовірно, перейдуть в категорію колишніх клієнтів. На рисунку 1.4 зображений схематичний поділ таких даних.

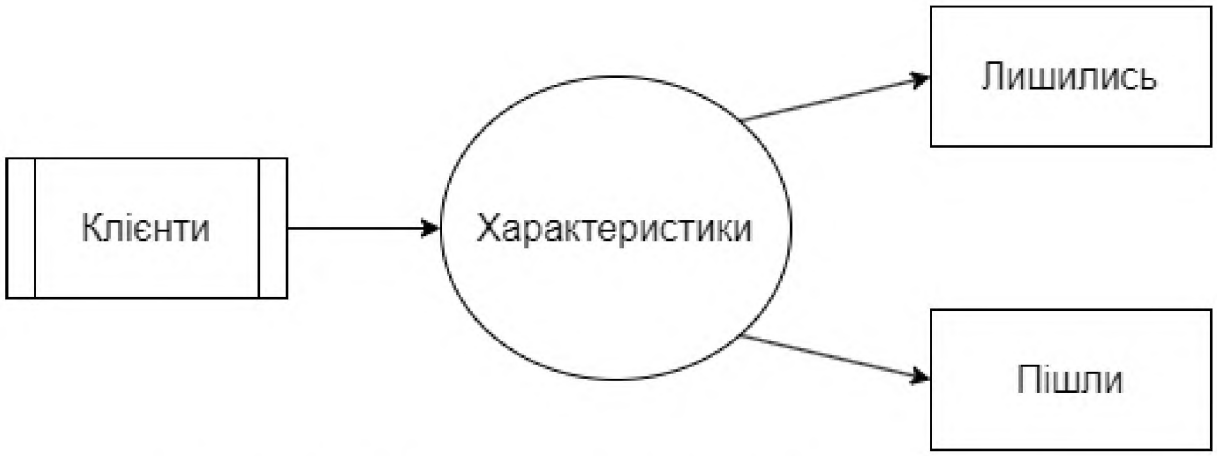


Рисунок 1.4 – Приклад поділу існуючої бази за ключем

На цю групу можна націлити особливий режим, який надто дорогий для застосування до клієнтської бази в цілому. Таким чином, ті самі методи можна використовувати для ідентифікації потенційних клієнтів, які можуть бути залучені до іншої послуги, що надається підприємством, але до якої зараз вони не відносяться.

У сьогоднішньому світі висококонкурентна економіка, орієнтована на клієнта та орієнтована на послуги – це дані, сировина, яка сприяє розвитку бізнесу, якщо тільки її можна видобути і використати правильно. Це вимагає розробки нових методів та технологій для ефективного керування та аналізу цих даних, а також створення нових стратегій та підходів для використання цих даних для досягнення заданих бізнес-цілей.

Прибутковість інтелектуальної обробки даних полягає у визначенні можливостей, тобто моделей у поведінці, яку можна перетворити на прибутковий бізнес, з подальшим використанням. Але це вимагає глибокого розуміння даних та здатності виявляти та використовувати ці моделі ефективно.

Інтелектуальний аналіз даних, як правило, визначається як процес виявлення шаблонів у великих наборах даних [10, с. 21]. Цей процес, як правило, є автоматичним або напівавтоматичним, що означає, що він використовує алгоритми та програмне забезпечення для виявлення цих шаблонів, замість того, щоб вимагати від людини вручну переглядати кожен елемент даних.

Ключі, які виявлені в результаті цього процесу, повинні бути значущими, оскільки вони зазвичай призводять до певної переваги, такої як здатність передбачати майбутню поведінку або виявляти невідомі взаємозв'язки між різними елементами даних.

Є два основних способи представлення шаблонів, які виявлені в результаті інтелектуального аналізу даних: як «чорний ящик», де внутрішня структура шаблону незрозуміла, і як «прозорий ящик», де структура шаблону відкрита для дослідження [11, с. 114]. Це важливі концепції, які використовуються в аналізі даних, і вони мають різні переваги та недоліки.

Обидва варіанти можуть робити хороші прогнози, але вони відрізняються за тим, наскільки легко можна зрозуміти, як саме вони прийшли до своїх висновків. Шаблони «чорного ящика» можуть бути дуже потужними, але вони можуть бути важкими для інтерпретації, тоді як шаблони «прозорого ящика» можуть бути менш потужними, але вони надають більше інформації про те, як саме вони працюють. На цей важливий аспект аналізу даних, варто звернути увагу при виборі методу аналізу.

Ці шаблони, які використовуються у інтелектуальному аналізі даних, називаються структурними, тому що вони фіксують структуру рішення явним чином. Це важливий процес, який включає в себе розуміння та аналіз великої кількості даних. Іншими словами, вони допомагають пояснити щось про дані та впорядкувати ці дані для створення нової структури. Це може бути важливим інструментом для розуміння та використання даних у більш продуктивний спосіб.

1.2 Використання інтелектуального аналізу даних для класифікації даних

З самого початку людського життя, люди завжди шукали закономірності в даних. Це було відображено в різних аспектах їхнього життя. Мисливці, наприклад, шукали закономірності в міграційній поведінці тварин, спостерігаючи за їхніми звичками та шляхами, щоб краще зрозуміти, коли і де вони можуть знайти свою наступну здобич.

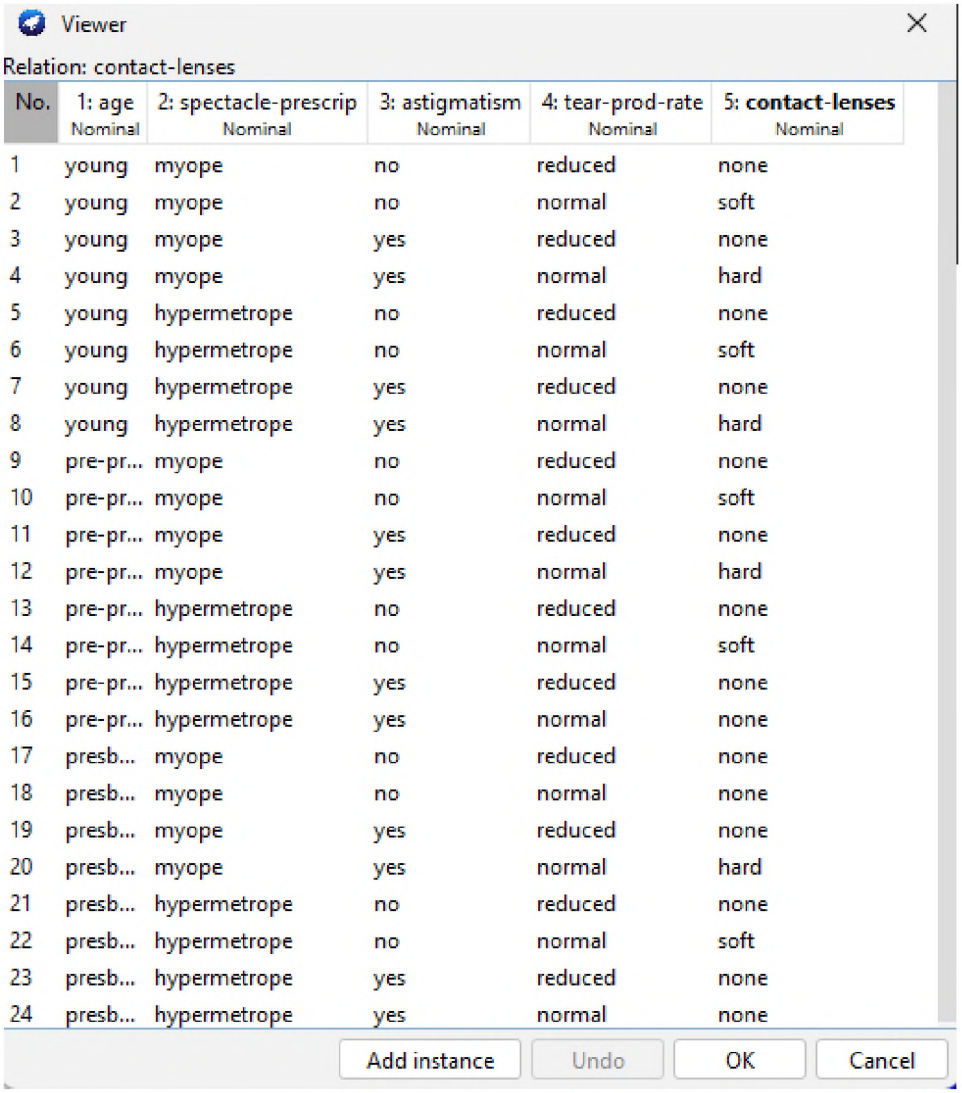
Фермери, з іншого боку, шукали закономірності в посівах зростання. Вони спостерігали за тим, як різні культури реагують на різні умови, такі як погода, тип ґрунту та використання добрив, щоб вони могли оптимізувати свої врожаї та збільшити продуктивність.

Політики шукають шаблони в головах виборців, аналізуючи тренди голосування, демографічні дані та громадську думку, щоб краще зрозуміти своїх виборців та розробити політики, які відповідають їхнім потребам та бажанням.

А закохані шукають шаблони в відповідях своїх партнерів, спостерігаючи за їхніми реакціями, емоціями та поведінкою, щоб краще зрозуміти їхні почуття та потреби.

Так само у інтелектуальному аналізі даних шаблони потрібні для того, щоб зрозуміти дані, виявити закономірності, які керують тим, як працює фізичний світ, і включити їх у теорії, які можна використовувати для прогнозування того, що станеться в новій ситуації. Це відкриває нові горизонти для наукових досліджень та інновацій, оскільки це дозволяє краще зрозуміти світ навколо та розробити нові способи вирішення проблем.

Розглянемо приклад структурного шаблону для набору даних, який зображено на рисунку 1.5, що містить дані про контактні лінзи [12, с. 164].


Viewer

Relation: contact-lenses

No.	1: age Nominal	2: spectacle-prescrip Nominal	3: astigmatism Nominal	4: tear-prod-rate Nominal	5: **contact-lenses** Nominal
1	young	myope	no	reduced	none
2	young	myope	no	normal	soft
3	young	myope	yes	reduced	none
4	young	myope	yes	normal	hard
5	young	hypermetrope	no	reduced	none
6	young	hypermetrope	no	normal	soft
7	young	hypermetrope	yes	reduced	none
8	young	hypermetrope	yes	normal	hard
9	pre-pr...	myope	no	reduced	none
10	pre-pr...	myope	no	normal	soft
11	pre-pr...	myope	yes	reduced	none
12	pre-pr...	myope	yes	normal	hard
13	pre-pr...	hypermetrope	no	reduced	none
14	pre-pr...	hypermetrope	no	normal	soft
15	pre-pr...	hypermetrope	yes	reduced	none
16	pre-pr...	hypermetrope	yes	normal	none
17	presb...	myope	no	reduced	none
18	presb...	myope	no	normal	none
19	presb...	myope	yes	reduced	none
20	presb...	myope	yes	normal	hard
21	presb...	hypermetrope	no	reduced	none
22	presb...	hypermetrope	no	normal	soft
23	presb...	hypermetrope	yes	reduced	none
24	presb...	hypermetrope	yes	normal	none

Add instance | Undo | OK | Cancel

Рисунок 1.5 – Вигляд набору даних з інформацією про контактні лінзи

При застосуванні структурного шаблону, частина структурного опису цієї інформації може бути такою як зображена на рисунку 1.6.

```
if (tearProductionRate == reduced) {
  var recommendation = "none";
} else if (age == "young" && astigmatic == "no") {
  var recommendation = "soft";
}
```

Рисунок 1.6 – Приклад структурного опису

Структурні описи – це важливий аспект аналізу даних, який не обов'язково повинен сформулюватися як правила, такі як ці. Існує багато різних способів представлення цих описів, і одним з них є дерева рішень [13, с. 1], які визначають послідовності рішень, які необхідно прийняти і отримана в результаті рекомендація є ще одним популярним засобом вираження структурного шаблону.

Цей приклад, який зараз розглядається, дуже спрощений. Його мета продемонструвати, що таке структурний шаблон, який використовується у навчанні.

По-перше, можливі всі комбінації значень, що представлені у наборі даних. Є двадцять чотири рядки, що представляють три можливі значення віку та по два значеннях для окулярів і астигматизму, що створює велику кількість можливих комбінацій.

Правила насправді не узагальнюють дані; вони просто підсумовують це. У більшості навчальних ситуацій набір прикладів, наведених як вхідні дані, далеко не завершено, і частина роботи полягає в тому, щоб узагальнити інші, нові приклади.

По-друге, значення вказані для всіх функцій у всіх прикладах. Реальні набори даних незмінно містяться приклади, в яких значення деяких ознак, для деяких

причина чи інші дані, невідомі – наприклад, не проводилися вимірювання або були втрачені. Це важливий аспект аналізу даних, який варто враховувати при виборі методу аналізу.

По-третє, попередні правила класифікують приклади правильно, тоді як часто через помилки або шум у даних, неправильна класифікація трапляється навіть у даних, які використовуються при контрольованому навчанні класифікатора.

1.3 Побудова структури даних на базі шаблонів класифікації та асоціацій

Розглянемо ще один приклад, який має назву «Проблема погоди». Для того, щоб глибше зануритися в цю проблему, потрібен спеціально підготовлений набір даних. Цей набір даних відображає певні погодні ознаки, за умов яких відбувається, чи не відбувається гра. Це допоможе краще зрозуміти, як погода впливає на проведення гри. На рисунку 1.7 зображений цей набір даних, який буде використовуватися для аналізу.

Viewer

Relation: weather.symbolic

No.	1: outlook Nominal	2: temperature Nominal	3: humidity Nominal	4: windy Nominal	5: **play** Nominal
1	sunny	hot	high	FALSE	no
2	sunny	hot	high	TRUE	no
3	overcast	hot	high	FALSE	yes
4	rainy	mild	high	FALSE	yes
5	rainy	cool	normal	FALSE	yes
6	rainy	cool	normal	TRUE	no
7	overcast	cool	normal	TRUE	yes
8	sunny	mild	high	FALSE	no
9	sunny	cool	normal	FALSE	yes
10	rainy	mild	normal	FALSE	yes
11	sunny	mild	normal	TRUE	yes
12	overcast	mild	high	TRUE	yes
13	overcast	hot	normal	FALSE	yes
14	rainy	mild	high	TRUE	no

Рисунок 1.7 – Набір даних для вирішення «Проблеми погоди»

Екземпляри у наборі даних характеризуються значеннями характеристик або атрибутів, які вимірюють різні аспекти екземпляра. У цьому випадку є чотири ознаки: «outlook»(кругозір), «temperature»(температура), «humidity»(вологість) і «windy»(вітер). Результат, який буде прогнозуватися – грати чи ні.

У своїй найпростішій формі, наведеній вище, усі чотири атрибути мають значення у вигляді символічних категорії, а не чисел. «outlook» може набувати значення «overcast», «sunny» або «rainy»; «temperature» може набувати значення «hot», «mild» або «cool»; «humidity» може бути «high» або «normal». «Windy» може приймати значення «true» або «false». Це створює 36 можливих комбінацій, що відображають різноманітність можливих ситуацій, які можна зустріти в реальному світі з яких 14 присутні у наборі вхідних прикладів.

Набір правил, отриманих на основі цієї інформації, може виглядати так, як показано на рисунку 1.8.

```
if (outlook == "sunny" && humidity == "high") {
  var play = "no";
} else if (outlook == "rainy" && windy == true) {
  var play = "no";
} else if (outlook == "overcast") {
  var play = "yes";
} else if (humidity == "normal") {
  var play = "yes";
} else {
  var play = "yes";
}
```

Рисунок 1.8 – Набір правил заснований на наборі даних

Ці правила, які тут розглядаються, призначені для тлумачення по порядку: спочатку перше, потім, якщо ні застосувати друге і так далі. Набір цих важливих

правил, призначених для тлумачення такий як показана вище послідовність, називається списком рішень. Він трактується як перелік рішень, правил які допомагають правильно класифікувати всі окремо взяті приклади. У дещо складнішій формі, ті самі дані показані на рисунку 1.9.

Viewer

Relation: weather

No.	1: outlook Nominal	2: temperature Numeric	3: humidity Numeric	4: windy Nominal	5: **play** Nominal
1	sunny	85.0	85.0	FALSE	no
2	sunny	80.0	90.0	TRUE	no
3	overcast	83.0	86.0	FALSE	yes
4	rainy	70.0	96.0	FALSE	yes
5	rainy	68.0	80.0	FALSE	yes
6	rainy	65.0	70.0	TRUE	no
7	overcast	64.0	65.0	TRUE	yes
8	sunny	72.0	95.0	FALSE	no
9	sunny	69.0	70.0	FALSE	yes
10	rainy	75.0	80.0	FALSE	yes
11	sunny	75.0	70.0	TRUE	yes
12	overcast	72.0	90.0	TRUE	yes
13	overcast	81.0	75.0	FALSE	yes
14	rainy	71.0	91.0	TRUE	no

Рисунок 1.9 – Набір правил заснований на наборі даних

У цьому випадку два атрибути – «temperature» і «humidity» мають числові значення. Це називається числовим атрибутом.

Це означає, що будь-який метод навчання повинен створювати нерівності щодо цих атрибутів, а не прості тести на рівність, як у попередньому випадку. Проблемою – у цьому випадку буде проблема змішаних атрибутів, оскільки не всі атрибути числові. Цю проблему можна вирішити шляхом додавання асоціацій до числових значень.

Тепер перше правило, наведене раніше, може мати наступний вигляд, який зображено на рисунку 1.10.

```
if (temperature == "cool") {
  humidity = "normal";
} else if (humidity == "normal" && windy == false) {
  play = "yes";
} else if (outlook == "sunny" && play == "no") {
  humidity = "high";
} else if (windy == false && play == "no") {
  outlook = "sunny";
  humidity = "high";
}
```

Рисунок 1.10 – Набір правил нормалізації на наборі даних

Усі ці правила, які тут розглянуті, на 100% вірні на наведених даних. Вони не роблять помилкових прогнозів. Перші два застосовуються до чотирьох прикладів у наборі даних, третій – до трьох приклади, а четвертий до двох прикладів. Є багато інших правил: насправді, можна знайти майже шістдесят правил асоціації, які застосовуються до двох або більше прикладів про погоду та абсолютно правильні щодо цих даних. Це так багато, тому що на відміну від правил класифікації, правила асоціації можуть «передбачити» будь-який з атрибутів, а не тільки визначений клас, і навіть може вказувати більше ніж на одну річ. Наприклад, четверте правило передбачає, що «outlook» буде «sunny» і що «humidity» повітря буде «high».

Інформація про контактні лінзи, представлена раніше, дає змогу зрозуміти, який тип контактних лінз слід призначити, враховуючи певну інформацію про пацієнта. Таким чином проаналізувавши дані – можна надати певний прогноз того, які лінзи має отримати пацієнт маючи той чи інший діагноз.

Якщо застосувати розглянуті вище правила класифікації та асоціації, можна сформулювати зразок набору правил, отриманих на основі цієї інформації, як показано на рисунку 1.11.

```
if (tearProductionRate == "reduced") {
  recommendation = "none";
} else if (age == "young" && astigmatic == "no" && tearProductionRate == "normal") {
  recommendation = "soft";
} else if (age == "pre-presbyopic" && astigmatic == "no" && tearProductionRate == "normal") {
  recommendation = "soft";
} else if (age == "presbyopic" && spectaclePrescription == "myope" && astigmatic == "no") {
  recommendation = "none";
} else if (spectaclePrescription == "hypermetrope" && astigmatic == "no" && tearProductionRate == "normal") {
  recommendation = "soft";
} else if (spectaclePrescription ==  "myope" && astigmatic == "yes" && tearProductionRate == "normal") {
  recommendation = "hard";
} else if (age == "young" && astigmatic == "yes" && tearProductionRate == "normal") {
  recommendation = "hard";
} else if (age == "pre-presbyopic" && spectaclePrescription == "hypermetrope" && astigmatic == "yes") {
  recommendation = "none";
} else if (age == "presbyopic" &&  spectaclePrescription == "hypermetrope" && astigmatic == "yes") {
  recommendation = "none";
}
```

Рисунок 1.11 – Набір правил для набору даних з інформацією про контактні лінзи

Це досить великий набір правил, але вони правильно класифікують усі приклади. Ці правила повні та детерміновані: вони надають унікальний рецепт для кожного прикладу.

Іноді ймовірності або ваги можуть бути пов’язані з самими правилами, щоб вказати, що деякі є більшими важливі або надійніші за інші. Але створення такого набору правил на досить невеликому наборі даних займає певний проміжок часу і до того ж кількість атрибутів у цьому наборі дійсно невелика. Яким чином створювати набір правил для набору даних у якому кількість атрибутів може сягати кількасот? До того ж створення цих правил відбувається за участі людини, а не машини – а це не відповідає меті вирішення проблеми.

Успішна сфера досліджень машинного навчання була розпочата як спроба стиснути величезну базу даних можливих шахових партій і їхні результати в структуру даних прийнятного розміру. Структурою даних для вирішення цієї проблеми, було обрано не набір правил, які були розглянуті вище, а дерево рішень.

На рисунку 1.12 показано структурний опис набору даних, що містить інформацію про контактні лінзи у формі дерева рішень, яке для багатьох цілей є більш лаконічним і зрозумілим.

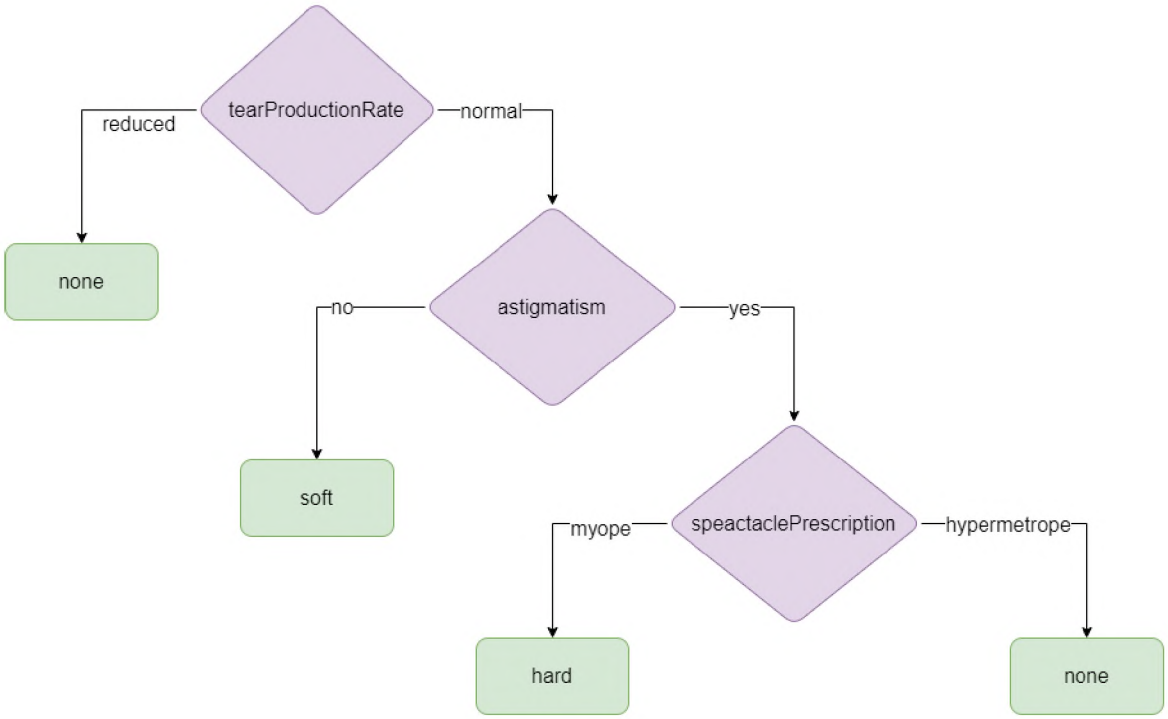


Рисунок 1.12 – Дерево рішень для набору даних з інформацією про контактні лінзи

Саме такі дерева використовуються класифікаторами для навчання та прийняття рішень. На основі такого дерева може бути створена модель НМ яка здатна класифікувати і розрізняти величезну кількість даних. Але у таких дерев існують певні обмеження, які будуть розглянуті в наступному розділі.

Висновки до розділу 1

В першому розділі було розглянуто передумови виникнення інтелектуального аналізу даних, проаналізовані статистичні дані з різних джерел. На основі результатів отриманих із цих даних, був зроблений висновок, що кількість даних у світі зростає, а отже для класифікації текстів, тематика цієї кваліфікаційної роботи - актуальна.

Також було розглянуто поняття шаблонів що використовуються для класифікації тексту, шаблони асоціації та класифікації та структуру даних що називається – дерева прийняття рішень.

РОЗДІЛ 2
РОЗРОБКА ТЕХНОЛОГІЇ КЛАСИФІКАЦІЇ ТЕКСТУ

2.1 Особливості застосування контрольованого навчання для завдань класифікації тексту

У будь-якій ІС поняття того, що таке вхід і вихід, набагато важливіше, ніж знання того, які процеси взаємодії відбуваються між ними, і класифікація тексту не виняток.

Вхідними даними для класифікації тексу є набір екземплярів. Ці екземпляри – це об’єкти, які необхідно класифікувати, асоціювати або групувати. Всі вони характеризується значеннями набору попередньо визначених атрибутів. Кожен набір даних представлено як матрицю екземплярів на атрибути, яка в термінах реляційної бази даних є відношенням один до багатьох [14].

Кожен екземпляр даних характеризується значеннями атрибутів, які вимірюють різні аспекти екземпляра. На рисунку 2.1 можна побачити приклад того як виглядають атрибути для екземпляра «погода».

Viewer

Relation: weather.symbolic

No.	1: outlook Nominal	2: temperature Nominal	3: humidity Nominal	4: windy Nominal	5: **play** Nominal
1	sunny	hot	high	FALSE	no

Рисунок 2.1 – Приклад атрибутів екземпляру

У цьому прикладі атрибутами для екземпляру «погода» є:

- кругозір(«outlook»);
- температура(«temperature»);
- вологість(«humidity»);
- вітряність(«windy»);

– гра(«play»).

Існує багато різних типів атрибутів, хоча типові методи класифікації тексту мають справу лише з числовими та номінальними даними.

Підготовка вхідних даних для класифікації тексту зазвичай займає основну частину зусилля, витрачені на весь процес класифікації даних. Для класифікації тексту WEKA використовує певний формат вхідного файлу, файл зв'язку атрибутів (arff формат) [15], який використовується в пакеті Java.

Приступаючи до роботи над вирішенням проблеми класифікації тексту, спочатку необхідно звести усі дані разом у набір екземплярів. Інтеграція даних із різних джерел зазвичай створює багато проблем через те, що дані потрібно сегментувати а потім заповнювати їх атрибути. На рисунку 2.2 можна побачити приклад *.arff файлу з даними які використані у цій кваліфікаційній роботі.

application.yaml | train.arff | output.arff

```
1  @relation Reviews
2
3  @attribute Text string
4  @attribute Class {positive,negative}
5
6  @data
7  'Wow... Loved this place.',positive
8  'Crust is not good.',negative
9  'Not tasty and the texture was just nasty.',negative
10 'Stopped by during the late May bank holiday off Rick Steve recommendation
11 'The selection on the menu was great and so were the prices.',positive
12 'Now I am getting angry and I want my damn pho.',negative
13 'Honeslty it did not taste THAT fresh.)',negative
14 'The potatoes were like rubber and you could tell they had been made up ah
15 'The fries were great too.',positive
16 'A great touch.',positive
17 'Service was very prompt.',positive
18 'Would not go back.',negative
19 'The cashier had no care what so ever on what I had to say it still ended
20 'I tried the Cape Cod ravoli, chicken, with cranberry...mmmm!',positive
21 'I was disgusted because I was pretty sure that was human hair.',negative
22 'I was shocked because no signs indicate cash only.',negative
23 'Highly recommended.',positive
24 'Waitress was a little slow in service.',negative
25 'This place is not worth your time, let alone Vegas.',negative
26 'did not like at all.',negative
27 'The Burrittos Blah!',negative
28 'The food, amazing.',positive
29 'Service is also cute.',positive
```

Рисунок 2.2 – Начальна вибірка у arff форматі

Більшість прикладів у першому розділі демонструють саме проблеми класифікації тексту. Набір даних для вирішення «Проблеми Погоди» представляє набір днів разом із рішенням для кожного щодо того, грати чи ні. Завдання полягає в тому, щоб навчити ІС класифікувати нові дні, які потрапляють на вхід, як ті в які можна грати чи не можна грати.

Якщо ж взяти дані з набору даних про контактні лінзи, то у цьому випадку завдання буде полягати в тому, щоб навчити ІС приймати рішення щодо того, які рекомендувати лінзи для нового пацієнта. В обох випадках для отримання результату необхідно навчити ІС узагальнювати дані. Для узагальнення існують чотири основних методи:

1. Навчання за класифікаціями. Ця схема навчання представлена набором класифікованих прикладів, на основі яких відбувається навчання з метою класифікувати нові приклади, що не мають цієї класифікації.

2. Асоціативне навчання. В асоціативному навчанні шукають будь-який зв'язок між ознаками, а не просто ті, що передбачають певне значення класу.

3. Кластеризація. У кластеризації екземпляри розділяються по групах, а потім відбувається пошук до якої групи відноситься той чи інший екземпляр.

4. Числове навчання. У числовому навчанні результат, який потрібно передбачити це не дискретний клас, а числова величина.

Навчання за класифікаціями іноді називають контрольованим, оскільки в певному сенсі цей метод працює під наглядом [16, с. 54]. Він класифікує реальний результат для кожного з навчальних екземплярів – грати чи не грати. Цей результат називається класом екземпляра. Також у контрольованого навчання існує синонім – навчання із вчителем. У цій кваліфікаційній роботі побудована ІС використовує саме такий підхід до навчання.

За замовчуванням фреймворк WEKA завжди використовує останнє значення для прогнозу результату класифікації. Наприклад, для екземпляру «погода» прогнозується результат класу «play»: «yes» чи «no», що несе інформацію про те, чи відбудеться гра, чи ні. Але якщо потрібно спрогнозувати інший атрибут, то для отримання цього результату можна змінити цю поведінку за допомогою інтерфейсу

користувача. Там можна вказати який атрибут буде класифікуватися та передбачатися. На рисунку 2.3 зображений приклад такого налаштування через інтерфейс фреймворку WEKA.

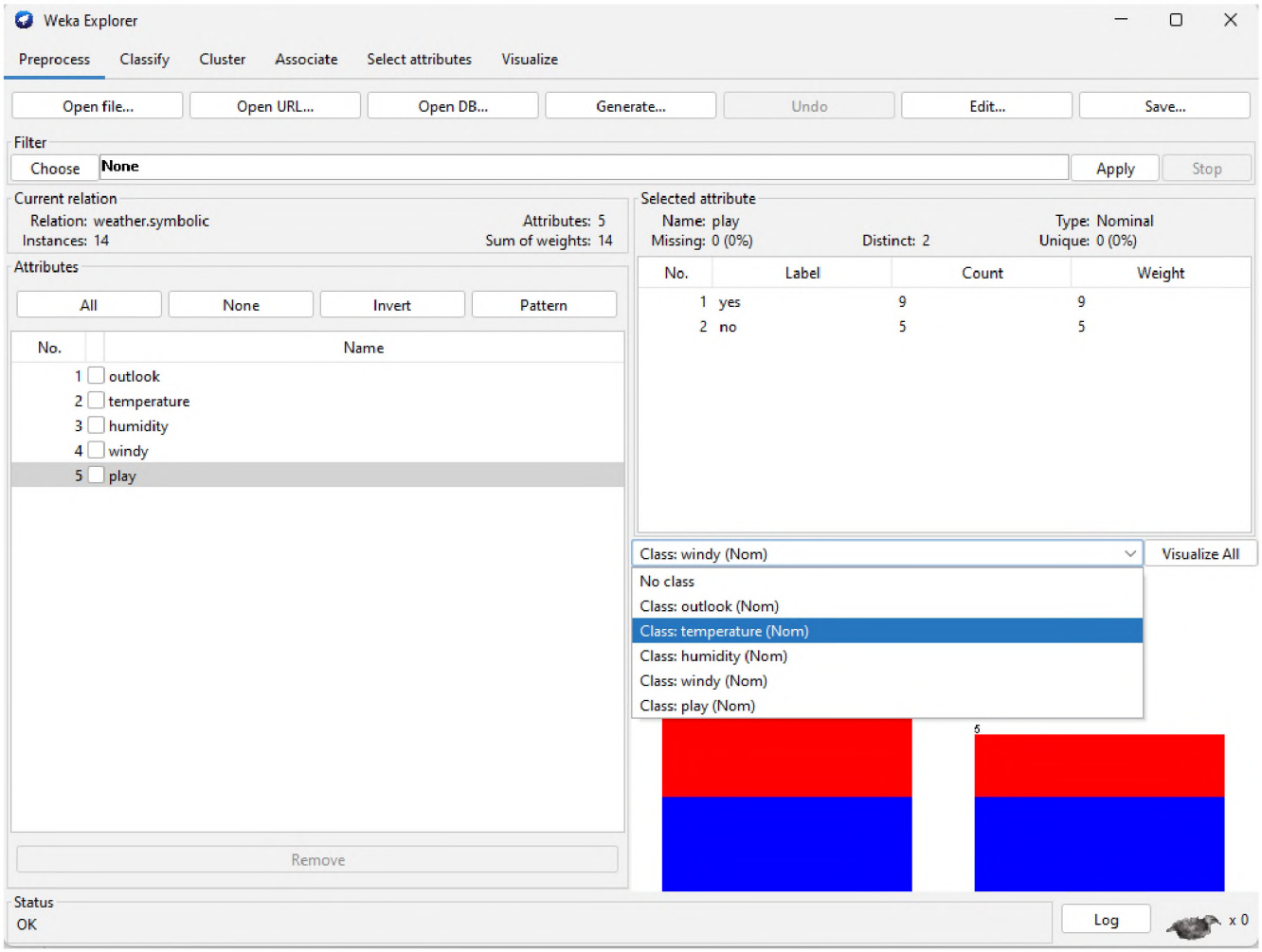


Рисунок 2.3 – Приклад зміни атрибуту для класифікації

Успішність навчання можна оцінити, спробувавши перевірити дані, отримані на незалежному наборі тестових даних, для яких справжня класифікація відома, але при цьому ця інформація не доступна для ІС. Рівень успіху тестових даних забезпечує розуміння того, наскільки добре була засвоєна концепція навчання за певним методом.

Більшість прикладів у першому розділі також можна однаково добре використовувати для асоціативного навчання, в якому немає конкретного

класу [17, с. 19]. Методи асоціативного навчання відрізняються від методів навчання за класифікацією двома пунктами:

– вони можуть спрогнозувати значення будь-якого атрибуту, а не тільки класу;

– вони можуть передбачити більше ніж одне значення атрибута одночасно.

З цієї причини правила методу асоціативного навчання часто обмежено тими, які застосовуються до певної мінімальної кількості прикладів, скажімо 80% набору даних і перевищують певний мінімальний рівень точності скажімо 95%.

Якщо клас не визначено, для групування елементів які з'являються – використовується метод кластеризації [18]. Якщо екземпляр для набору даних про контактні лінзи має атрибут в якому відсутнє значення, тоді розподілення буде проведено по тим групам атрибутів, які є спільними для кожного екземпляру.

Числове прогнозування [19] є варіантом класифікаційного навчання, в якому результатом є числове значення, а не категорія.

Усі ці методи використовують класифікатори для створення моделі для вирішення проблеми класифікації тексту – що безпосередньо і полягає у тому, щоб спрогнозувати значення класу для екземпляра на основі атрибутів попередніх екземплярів.

2.2 Класифікації тексту на базі дерев прийняття рішень

Підхід «розділяй і володарюй», який є досить популярним у сфері машинного навчання, до проблеми навчання з набору незалежних екземплярів природно призводить до стилю представлення, яке називається деревом рішень [20, с. 308]. Вузли в дереві рішень включають тестування конкретного атрибута. Як правило, тест на вузлі порівнює значення атрибута з константою.

Однак деякі дерева, які є більш складними, порівнюють два атрибути один з одним або використовують певну функцію одного чи кількох атрибутів. Листові вузли забезпечують класифікацію, яка застосовується до всіх екземплярів, які

досягають дерева, або набору класифікацій, або розподілу ймовірностей за всіма можливими класифікаціями.

Щоб класифікувати невідомий екземпляр, він направляється вниз по дереву відповідно до значень атрибутів, перевірених на послідовних вузлах, і коли досягає потрібного листа, екземпляр класифікується відповідно до класу, призначеного вузлу.

Якщо атрибут, який перевіряється на вузлі, є номінальним, кількість дочірніх елементів зазвичай дорівнює кількості можливих значень атрибута. У цьому випадку, оскільки існує одна гілка для кожного можливого значення, той самий атрибут не перевірятиметься знову далі по дереву. На рисунку 2.4 зображений приклад побудови дерева для номінального атрибута за допомогою фреймворку WEKA.

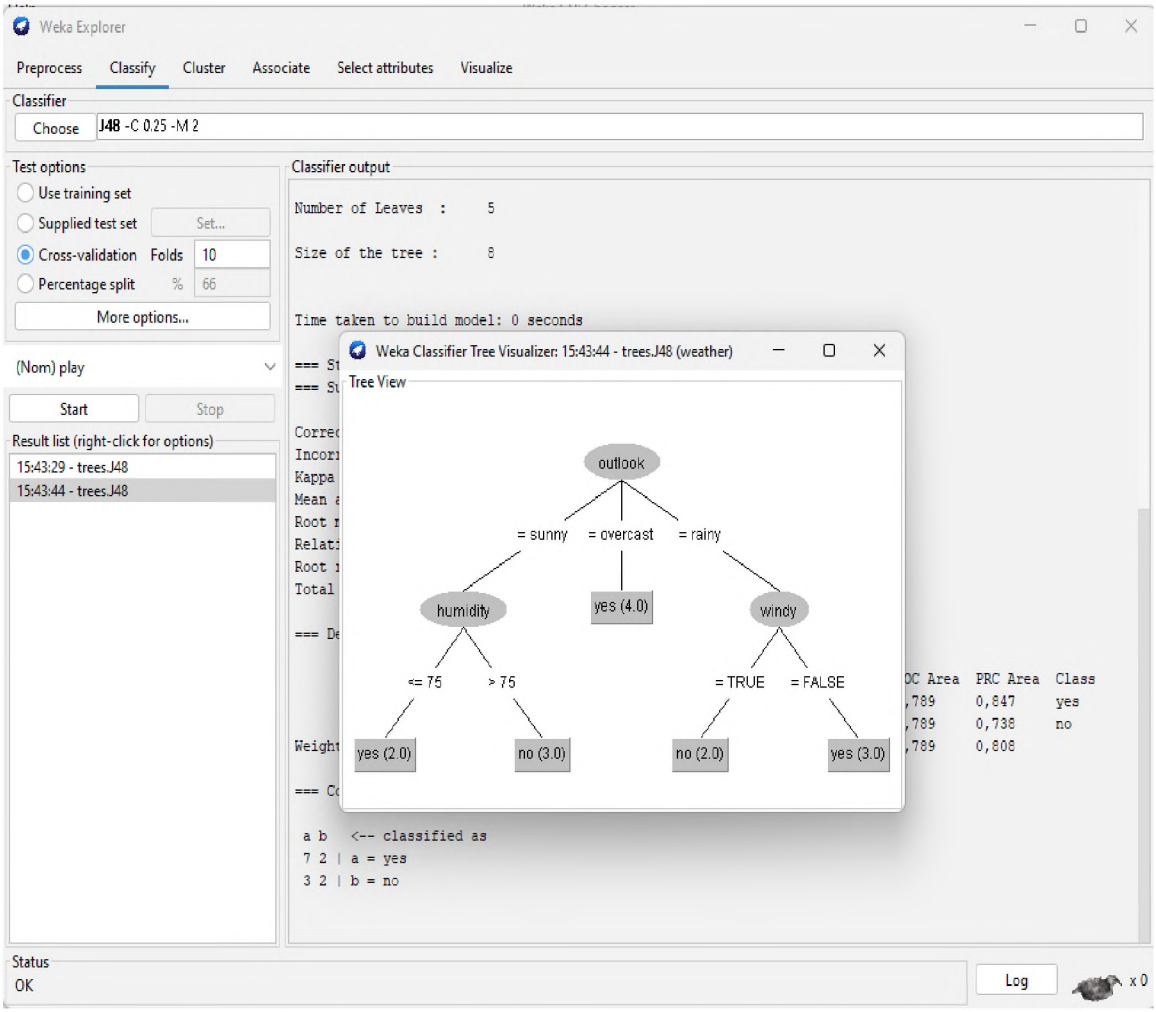


Рисунок 2.4 – Побудова дерева прийняття рішень для номінального атрибута

Іноді значення атрибутів розбиваються на дві підмножини, і дерево розгалужується лише двома шляхами залежно від того, до якої підмножини належить значення в дереві. У цьому випадку атрибут можна перевірити кілька разів.

Якщо атрибут є числовим, перевірка у вузлі зазвичай визначає, чи є його значення більше або менше попередньо визначеної константи, що дає двобічний розподіл. На рисунку 2.5 зображений приклад побудови дерева для числового атрибута за допомогою фреймворку WEKA.

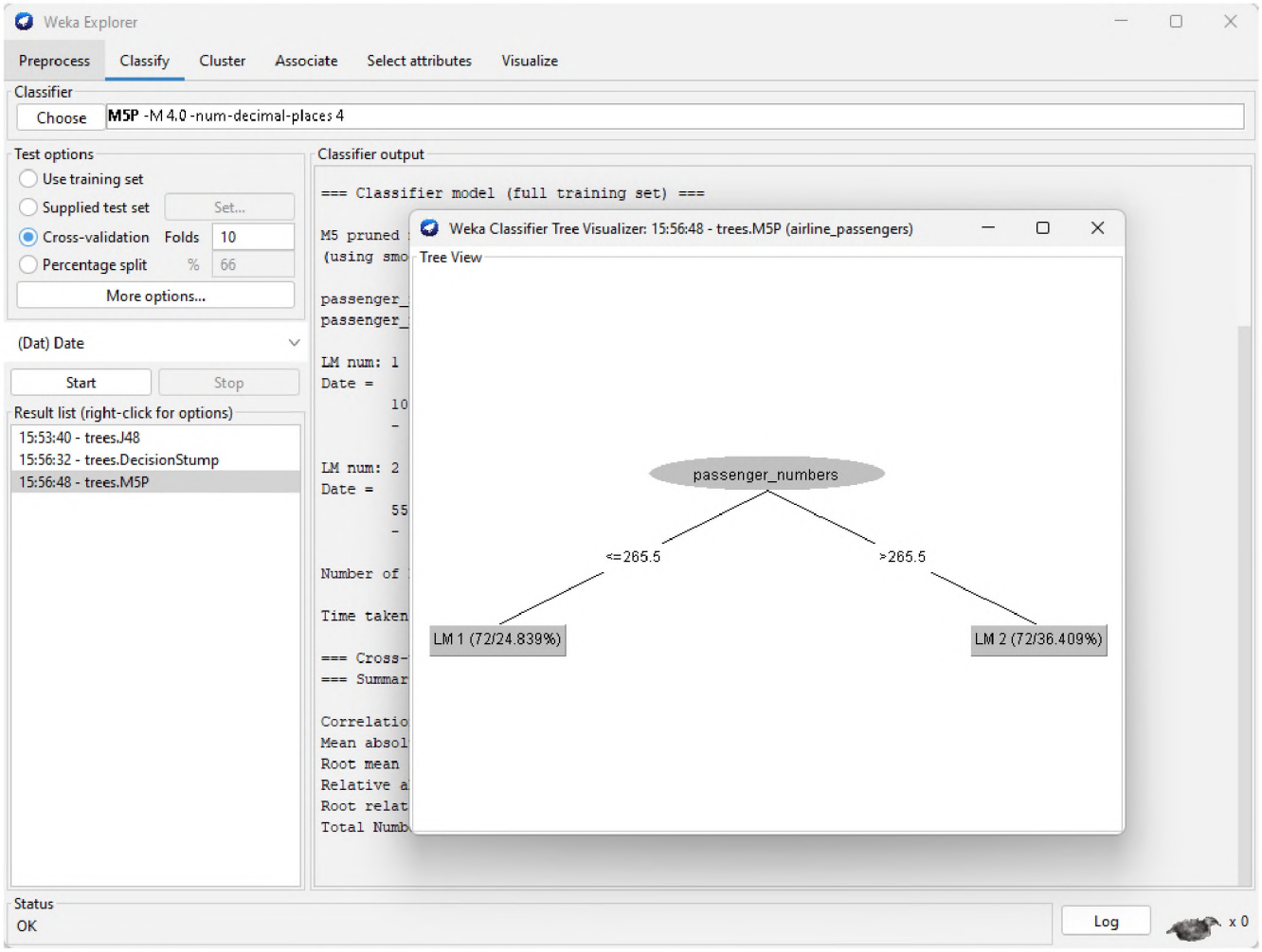


Рисунок 2.5 – Побудова дерева прийняття рішень для числового атрибута

Крім того, можна використовувати тристоронній розподіл [21, с. 129], і в цьому випадку є кілька різних можливостей. Якщо відсутнє значення розглядається як власне значення атрибута, це створить третю гілку. Альтернативою для

цілочисельного атрибута було б тристороннє розбиття на менше, дорівнює та більше.

Альтернативою для атрибута з дійсним значенням, для якого менш значущий параметр є рівним, було б доцільно робити перевірку на інтервалі, а не на одній константі, знову ж таки даючи тристоронній розподіл: нижче, всередині та вище. Числовий атрибут часто перевіряється кілька разів на будь-якому заданому шляху вниз по дереву від кореня до листа, кожен тест включає іншу константу.

Відсутні значення є очевидною проблемою. Незрозуміло, який маршрут повинен бути прийнятим, коли вузол перевіряє атрибут, значення якого відсутнє [22, с. 773]. Іноді, відсутнє значення розглядається як окреме значення атрибута. Якщо це не так, відсутні значення слід обробляти особливим чином і не розглядати їх як інше можливе значення, яке може приймати атрибут. Просте рішення полягає в тому, щоб записати кількість елементів у навчальному наборі, які спускаються до кожної гілки, і використовувати найпопулярнішу гілку, якщо немає значення для екземпляра тесту.

Більш складне рішення, яке вимагає більше обдумувань, полягає в тому, щоб умовно розділити кожний окремий екземпляр на декілька частин та відправити його частину вниз по кожній гілці, а звідти безпосередньо вниз до листів відповідних під дерев. Розподіл здійснюється за допомогою числової ваги від нуля до одиниці, а вага для гілки вибирається пропорційна до кількості екземплярів навчання, що йдуть до цієї гілки, при цьому всі ваги дорівнюють одиниці. Зважений екземпляр можна далі розділити на нижчий вузол. Згодом кожна з різних частин екземпляра досягне кінцевого вузла, і рішення в цих кінцевих вузлах повинні бути повторно об’єднані за допомогою ваг.

Щоб ефективно створити дерево рішень для набору даних, знадобиться хороший спосіб візуалізації даних [23, с. 170], щоб мати можливість вирішити, які дані, ймовірно, будуть найкращими атрибутами для перевірки, а який тест може бути відповідним.

Фреймворк WEKA має інструмент класифікації, який дозволяє користувачам будувати дерево рішень в інтерактивному режимі. Він надає точкову діаграму

даних для двох вибраних атрибутів. Коли користувач знаходить пару атрибутів, які розділяють класи, він може створити біноміальний розподіл, намалювавши багатокутник навколо відповідних точок даних на діаграмі розсіювання.

Для демонстрації візьмемо один із найпопулярніших наборів даних, що містить інформацію про три види ірисів [24], кожен з яких представлений 50 екземплярами. Для кожного екземпляра вказані такі атрибути:

- Довжина чашолистка/sepal length (в см);
- Ширина чашолистка/sepal width (в см);
- Довжина пелюстки/Peral length (в см);
- Ширина пелюстки/petal width (в см);
- Клас (вид ірису: Iris-setosa, Iris-versicolor, Iris-virginica).

На рисунку 2.6 зображений цей набір даних у arff форматі.

Viewer

Relation: iris

No.	1: sepallength Numeric	2: sepalwidth Numeric	3: petallength Numeric	4: petalwidth Numeric	5: class Nominal
1	5.1	3.5	1.4	0.2	Iris-setosa
2	4.9	3.0	1.4	0.2	Iris-setosa
3	4.7	3.2	1.3	0.2	Iris-setosa
4	4.6	3.1	1.5	0.2	Iris-setosa
5	5.0	3.6	1.4	0.2	Iris-setosa
6	5.4	3.9	1.7	0.4	Iris-setosa
7	4.6	3.4	1.4	0.3	Iris-setosa
8	5.0	3.4	1.5	0.2	Iris-setosa
9	4.4	2.9	1.4	0.2	Iris-setosa
10	4.9	3.1	1.5	0.1	Iris-setosa
11	5.4	3.7	1.5	0.2	Iris-setosa
12	4.8	3.4	1.6	0.2	Iris-setosa
13	4.8	3.0	1.4	0.1	Iris-setosa
14	4.3	3.0	1.1	0.1	Iris-setosa
15	5.8	4.0	1.2	0.2	Iris-setosa
16	5.7	4.4	1.5	0.4	Iris-setosa
17	5.4	3.9	1.3	0.4	Iris-setosa
18	5.1	3.5	1.4	0.3	Iris-setosa
19	5.7	3.8	1.7	0.3	Iris-setosa
20	5.1	3.8	1.5	0.3	Iris-setosa
21	5.4	3.4	1.7	0.2	Iris-setosa
22	5.1	3.7	1.5	0.4	Iris-setosa
23	4.6	3.6	1.0	0.2	Iris-setosa
24	5.1	3.3	1.7	0.5	Iris-setosa

Рисунок 2.6 – Набір даних, що містить інформацію про іриси

Наприклад, користувач працює з набором даних із трьох класів, з набору даних, що містить інформацію про іриси і знайшов два атрибути, довжина і ширина чашолистка, які добре справляються з розділенням класів. Для виділення одного з класів (Iris-versicolor) користувач може намалювати прямокутник, так як зображено на рисунку 2.7.

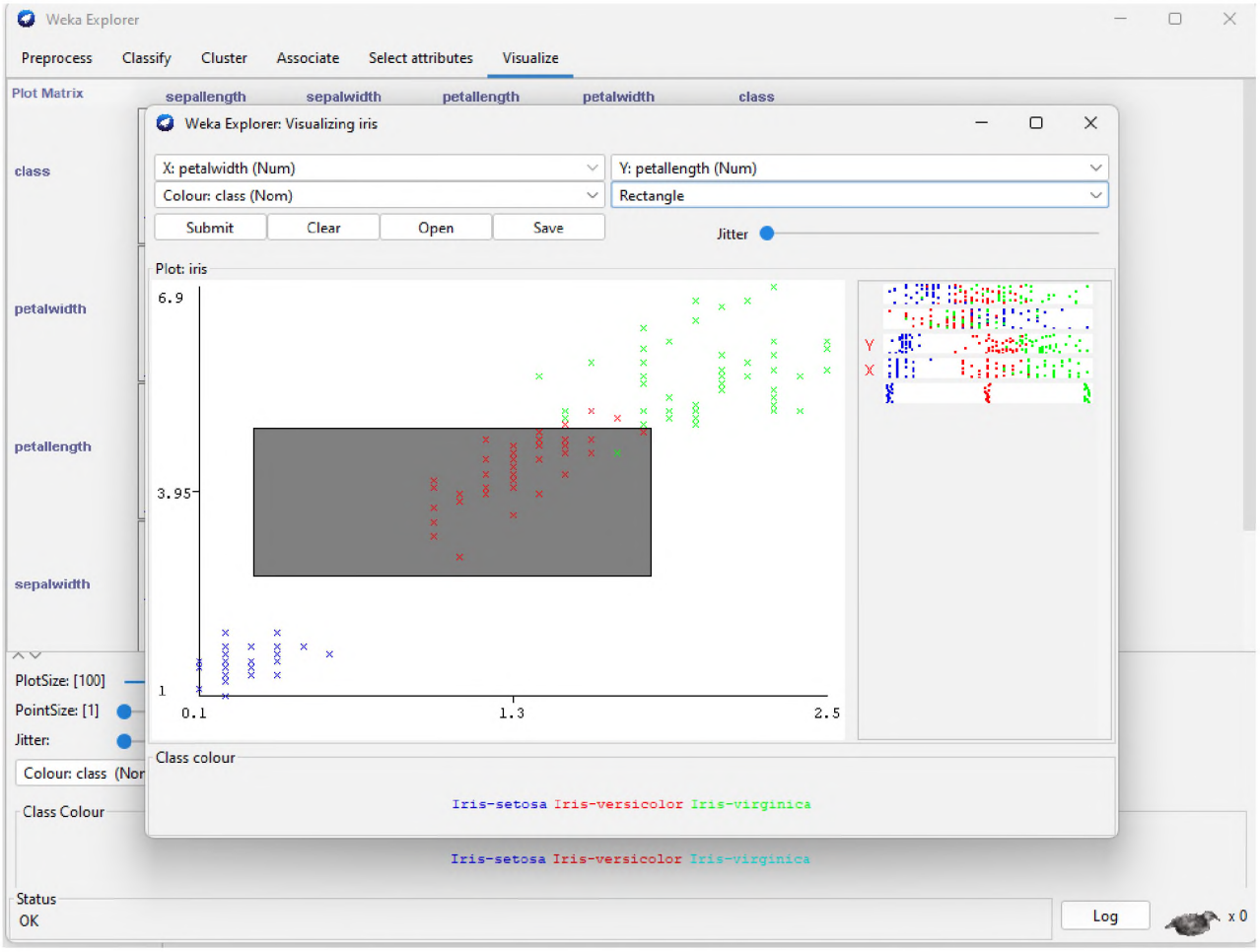


Рисунок 2.7 – Виділення конкретного типу екземпляра

Для такого розподілу дерево прийняття рішень матиме наступний вигляд:

– верхній листковий вузол має нормальне розподілення де переважають іриси одного типу – Iris-setosa, всі наступні віднесені далі;
– середній також має переважаючий вид – Iris-versicolor;
– і останній ряд в якому представлені Iris-virginica і виключення для Iris-setosa.

На рисунку 2.8 зображено отримане дерево прийняття рішень.

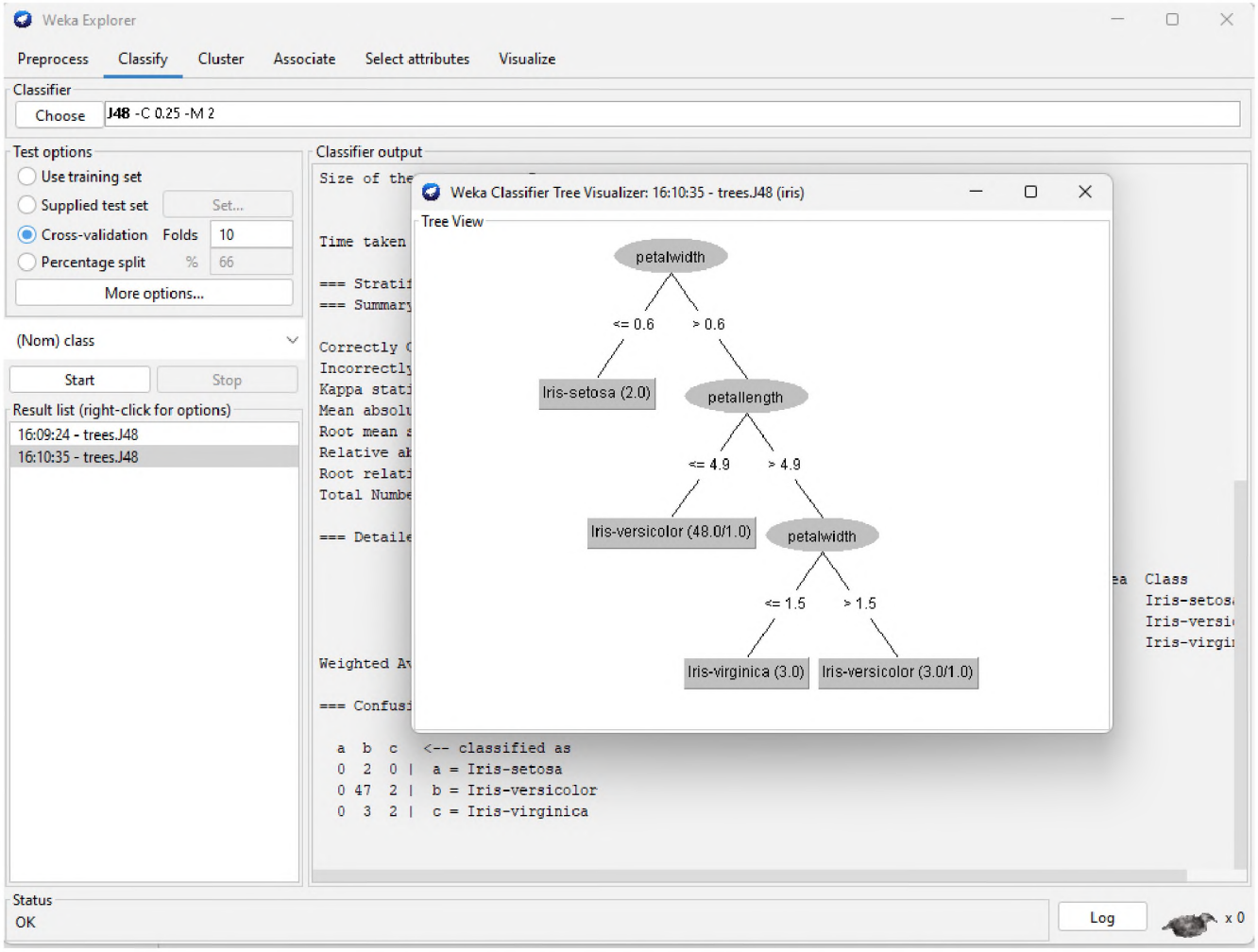


Рисунок 2.8 –Побудоване дерево рішень для обраних користувачем даних

Дерева рішень і правила, які були розглянуті вище, призначені для прогнозування категорій, а не числових величин. Коли справа доходить до прогнозування числових величин, можна використовувати той самий вид представлення дерева або правила, але листові вузли дерева або права частина правил міститимуть числове значення, яке є середнім усіх значень навчального набору, до якого застосовується вузол або правило.

Оскільки статистики використовують термін регресія для процесу обчислення виразу, який передбачає числову величину, дерева рішень із усередненими числовими значеннями на листках називаються деревами регресії [25, с. 164].

Для даного прикладу візьмемо набір даних, що містить інформацію про центральний процесор [26]. На рисунку 2.9 зображена частина цього набору даних.

Viewer

Relation: cpu

No.	1: MYCT Numeric	2: MMIN Numeric	3: MMAX Numeric	4: CACH Numeric	5: CHMIN Numeric	6: CHMAX Numeric	7: **class** Numeric
1	125.0	256.0	6000.0	256.0	16.0	128.0	198.0
2	29.0	8000.0	32000.0	32.0	8.0	32.0	269.0
3	29.0	8000.0	32000.0	32.0	8.0	32.0	220.0
4	29.0	8000.0	32000.0	32.0	8.0	32.0	172.0
5	29.0	8000.0	16000.0	32.0	8.0	16.0	132.0
6	26.0	8000.0	32000.0	64.0	8.0	32.0	318.0
7	23.0	16000.0	32000.0	64.0	16.0	32.0	367.0
8	23.0	16000.0	32000.0	64.0	16.0	32.0	489.0
9	23.0	16000.0	64000.0	64.0	16.0	32.0	636.0
10	23.0	32000.0	64000.0	128.0	32.0	64.0	1144.0
11	400.0	1000.0	3000.0	0.0	1.0	2.0	38.0
12	400.0	512.0	3500.0	4.0	1.0	6.0	40.0
13	60.0	2000.0	8000.0	65.0	1.0	8.0	92.0
14	50.0	4000.0	16000.0	65.0	1.0	8.0	138.0
15	350.0	64.0	64.0	0.0	1.0	4.0	10.0
16	200.0	512.0	16000.0	0.0	4.0	32.0	35.0
17	167.0	524.0	2000.0	8.0	4.0	15.0	19.0
18	143.0	512.0	5000.0	0.0	7.0	32.0	28.0
19	143.0	1000.0	2000.0	0.0	5.0	16.0	31.0
20	110.0	5000.0	5000.0	142.0	8.0	64.0	120.0
21	143.0	1500.0	6300.0	0.0	5.0	32.0	30.0
22	143.0	3100.0	6200.0	0.0	5.0	20.0	33.0
23	143.0	2300.0	6200.0	0.0	6.0	64.0	61.0
24	110.0	3100.0	6200.0	0.0	6.0	64.0	76.0

Рисунок 2.9 –Набір даних із інформацією про процесор

Для даних продуктивності процесора наведено наступну формулу:

$$\mathrm{Y} = b_0 + b_n \times var_n \ldots + b_{n+1} \times var_{n+1}, \qquad (2.1)$$

де Y – це залежна змінна (продуктивність процесора);

b_0 – константа;

b_n – коефіцієнти регресії;

var_n – порядковий незалежний атрибут екземпляра(MMIN, MMAX, CACH, CHMIN, CHMAX.

На рисунку 2.10 показано дерево регресії, яке було утворено за допомогою рівняння.

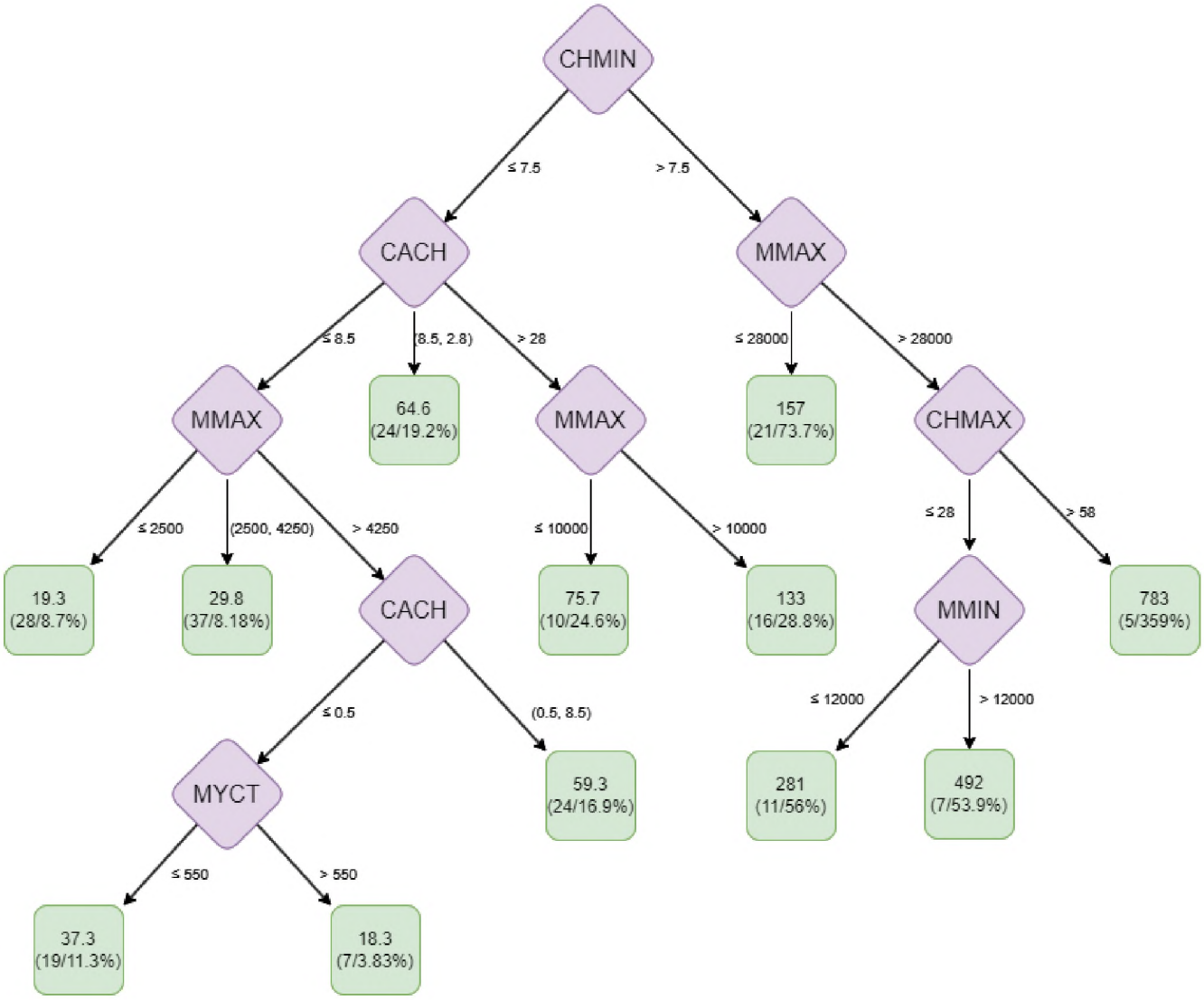


Рисунок 2.10 – Регресійне дерево прийняття рішень

Листя дерева – це числа, які представляють середній результат для екземплярів, які досягають листка. Дерево є набагато більшим і складнішим, ніж рівняння регресії, і якщо обчислити середнє абсолютних значень похибок між прогнозованою та фактичною продуктивністю процесора, воно виявиться значно меншим для дерева, ніж для рівняння регресії. Дерево регресії точніше, оскільки проста лінійна модель погано представляє дані в цій задачі [27, с. 56]. Однак дерево громіздке і його важко інтерпретувати через великі розміри.

Можна комбінувати рівняння регресії з деревами регресії. Таке дерево, листя якого містить лінійні вирази, тобто рівняння регресії, а не окремі прогнозовані значення, називається деревом моделі [28, с. 6]. Дерево моделі апроксимує безперервні функції лінійними «патчами», більш складним представленнями, ніж

лінійна регресія або регресійні дерева. Незважаючи на те, що модельне дерево є меншим і більш зрозумілим, ніж дерево регресії, середні значення помилок у навчальних даних нижче.

2.3 Класифікатори на основі Байєсівських нейронних мереж

Штучна нейронна мережа Байєса є розширенням класифікатора Байєса і спочатку була створена Ланснером і Екебергом у групі SANS Королівського технологічного інституту як рекурентна одношарова мережа. Байєсова нейронна мережа навчається відповідно до правила навчання Байєса, де нейрони в мережі представляють стохастичні події, а ваги обчислюються на основі кореляції між ними.

Топологічно Байєсовська нейронна мережа, яка використовується в цій дипломній роботі, нагадує архітектуру штучної нейронної мережі прямого зв'язку.

Одношарова нейронна мережа Баєса побудована на основі класифікатора Байєса, який передбачає незалежність між вхідними атрибутами. Це припущення не завжди вірне. Цю проблему вирішено в багатошаровій Байєсівській нейронній мережі шляхом введення прихованих стовпців, що представляють комбінації вхідних атрибутів.

Властивості пересилання сигналу нейронної мережі моделюються як зважена сума вхідних сигналів, за формулою:

$$s_j = b_j + \sum_i w_{ij} o_i, \tag{2.2}$$

де b_j – біас,

w_{ij} – ваги між вхідним сигналом i та нейроном j.

Функція активації часто є нелінійною та антисиметричною. Нейрон розділяє вхідний простір на дві підмножини за допомогою гіперплощини, де вхідні дані від однієї підмножини активують нейрон, а вхідні дані від іншої підмножини його деактивують.

У штучній нейронній мережі нейрони з'єднані один з одним. Існує кілька відповідних топологічних рішень або ж архітектур для вирішення різних проблем.

Зазвичай найпоширенішою архітектурою є шарова архітектура, де нейрони розміщені шарами. Нейрони одного шару живлять нейрони наступного шару.

Штучна нейронна мережа із такою архітектурою називається мережею прямого зв'язку. Можливості навчання штучного нейрона розглядаються під час навчання.

Ще однією популярною архітектурою побудови є наївний Байєсівський класифікатор та його реалізація у спрямованій двошаровій або багато направленій одношаровій Байєсівській нейронній мережі (БНН). Подібним чином напівнаївний Байєсівський класифікатор може бути також реалізований у спрямованій або багато направленій багатошаровій БНН.

Спрямована БНН насправді є наївним Байєсівським класифікатором, реалізованим як орієнтований ациклічний граф. Вхідні нейрони відповідають усім можливим (дискретним або дискретизованим) значенням атрибутів, а вихідні – міткам класів. Мережа отримує на вхід значення вхідних нейронів і за один крок обчислює значення активації для вихідних нейронів. Таким чином, питання про стабільність під час виконання не потрібно піднімати, на відміну від різноспрямованих нейронних мереж.

Багато спрямована БНН аналогічна нейронній мережі Хопфілда, а спрямована БНН аналогічна перцептрону. Нейронна мережа Хопфілда використовує узагальнене правило навчання Хебба, тоді як БНН використовує базове правило навчання Гебба.

В обох випадках складність навчання пропорційна кількості навчальних прикладів. У БНН навчання відбувається поетапно, і достатньо одного проходу через навчальний набір, на відміну від перцептрона, який використовує правило навчання дельта та повторює навчальний набір кілька разів. Що стосується наївності, спрямований БНН аналогічний двошаровому перцептрону, оскільки вони обидва можуть обчислювати лише лінійні функції.

У порівнянні з методами символічного навчання, одним із найсерйозніших недоліків нейронних мереж є їх нездатність зрозуміло пояснити свої рішення (класифікації). Ця проблема найбільш загострюється при розгляді нейронів прихованого шару багатошарових нейронних мереж. На відміну від нейронних мереж Хопфілда, двошарових і багатошарових перцептронів, усі БНН можуть зрозуміло пояснювати свої рішення. Також інтерпретація нейронів із прихованих шарів багатошарових нейронних мереж особливо складна, у багаторівневих БНН відбувається простіше.

2.4 Використання фреймворку WEKA для класифікації тексту

Проблему побудови дерева рішень можна виразити за допомогою рекурсивної функції. Рекурсією називається виконання одного і того ж самого процесу на вкладених об'єктах сумісного типу. Спочатку потрібно вибрати атрибут для розміщення в кореневому вузлі та зробити одну гілку для кожного можливого значення цього атрибуту [29, с. 4]. Це розбиває набір прикладів на підмножини, по одній для кожного значення атрибута. Тепер процес можна повторювати рекурсивно для кожної гілки, використовуючи лише ті екземпляри, які фактично досягають гілки.

Єдине, що залишається вирішити, це як визначити, який атрибут потрібно розділити, враховуючи набір прикладів з різними класами. Для прикладу розглянемо набір даних для вирішення, який найкраще зможе проілюструвати цей підхід, «Проблеми погоди». Кількість класів на листах не вказується. Будь-який лист із лише одним класом – так чи ні – не потребуватиме подальшого розбиття, і рекурсивний процес у цій гілці завершиться.

Оскільки мета пошуку – це невеликі дерева [30, с. 2], цей процес має статися якнайшвидше. Якби міра чистоти кожного вузла була доступною, була б доступна можливість обрати атрибут, який створює найчистіші дочірні вузли.

Є чотири варіанти для кожного розділу, і на верхньому рівні вони створюють дерева, подібні до тих, що зображені на рисунку 2.11.

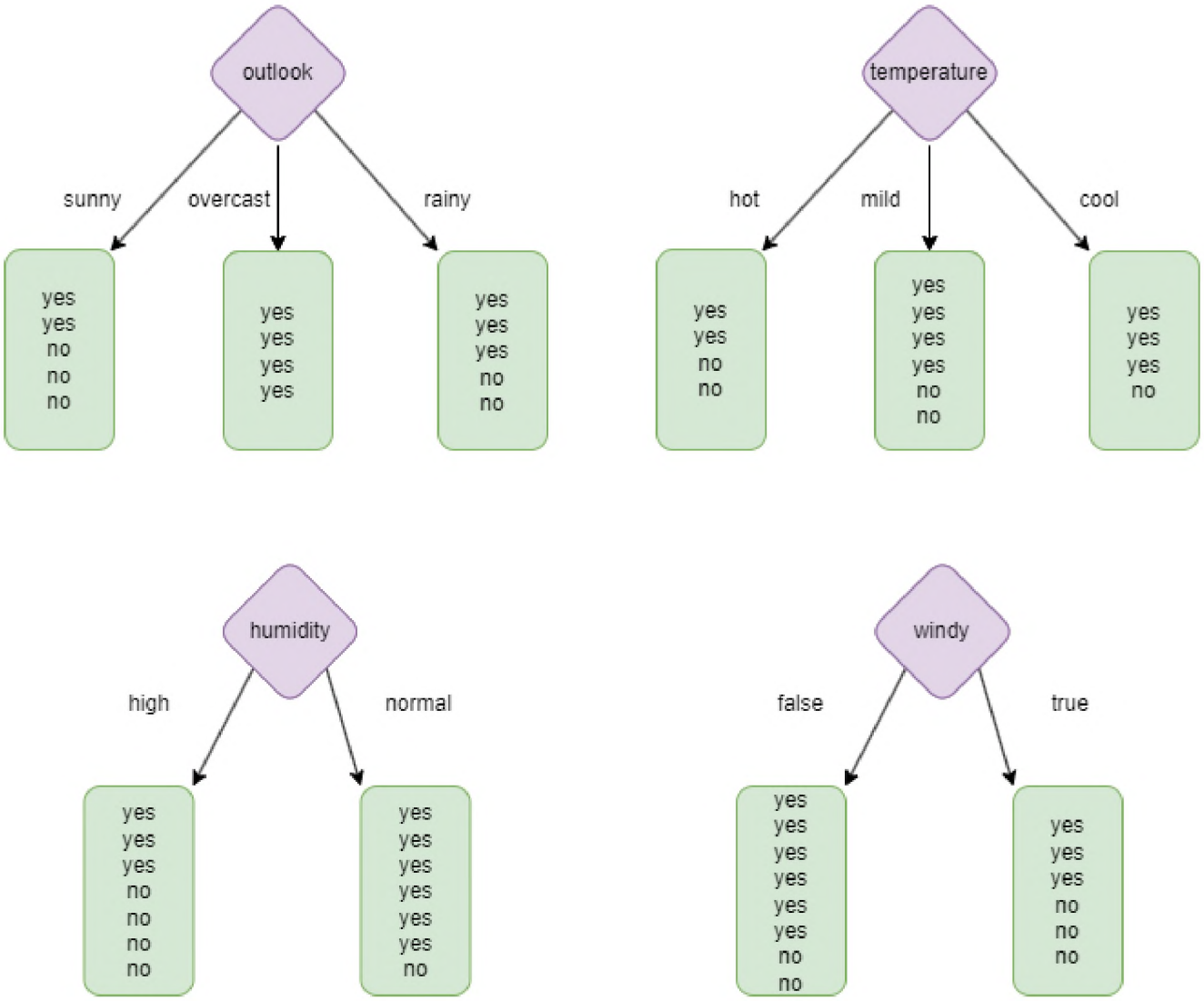


Рисунок 2.11 – Приклад рекурсивної побудови дерев прийняття рішень

Міра чистоти, яка буде використовуватися, називається інформацією та вимірюється в одиницях, які називаються бітами. Пов'язаний із вузлом дерева, біт представляє очікуваний обсяг інформації, яка знадобиться, щоб визначити, чи слід класифікувати новий екземпляр як «yes» чи «no», враховуючи, що екземпляр досяг цього вузла. Його обчислення відбувається на основі кількості класів «yes» і «no» у вузлі.

Під час оцінки першого дерева кількість класів «yes» і «no» у листових вузлах дорівнює:

– два «yes», три «no»;

– чотири «yes»;

– три «yes», два «no».

Перш ніж обчислювати кількість інформації, необхідної для визначення класу прикладу, враховуючи, що він досягає вузла дерева з певною кількістю «yes» і «no», потрібно розглянути спочатку тип властивостей цієї величини:

– Якщо кількість «yes» чи «no» дорівнює нулю, інформація дорівнює «0».

– При рівній кількості «yes» і «no» інформація досягає максимуму.

Інформаційна міра [31, с. 4] стосується кількості інформації, отриманої під час прийняття рішення, а більш детальну властивість інформації можна отримати, дивлячись на природу рішень. Рішення можуть прийматися як в один, так і в кілька етапів, і в обох випадках обсяг залученої інформації однаковий.

Для знаходження інформаційної міри використаємо формулу, яка задовольняє всі ці властивості:

$$H(X) = -p_n \log p_n - p_{n+1} \log p_{n+1}, \quad (2.3)$$

де p_n – це вага доказів для кожного конкретного випадку (частка конкретної величини від загальної).

Таким чином інформаційні значення вузлів першого дерева дорівнюють:

$$\text{Вузол}_1 = -(2 \div 5) \times \log(2 \div 5) - (3 \div 5) \times \log(3 \div 5) = 0.971;$$

$\text{Вузол}_2 = 0$, (тому що для цього випадку кількість «ні» рівна нулю);

$$\text{Вузол}_3 - (3 \div 5) \times \log(3 \div 5) - (2 \div 5) \times \log(2 \div 5) = 0.971.$$

Ми можемо обчислити середню інформаційну цінність цих даних, взявши до уваги кількість екземплярів, які йдуть вниз по кожній гілці – п’ять вниз по першій і третій і чотири вниз по другій за формулою:

$$IV = \sum_{i=1}^{n} p_i \times H(X_i), \quad (2.4)$$

де p_i – це вага доказів для кожного окремого випадку;

$H(X_i)$ – інформаційна міра для кожного випадку.

Таким чином рівняння для знаходження середньої інформаційної цінності буде виглядати наступним чином:

$$(5 \div 14) \times 0.971 + (4 \div 14) \times 0 + (5 \div 14) \times 0.971 = 0.693 \text{ біти}.$$

Це середнє значення представляє кількість інформації, яка, знадобиться для визначення класу нового екземпляра, враховуючи структуру рекурсивного дерева прийняття рішень.

Тепер потрібно розрахувати який із атрибутів представляє найбільш значущій коефіцієнт. Це можна зробити за формулою розрахунку коефіцієнтів ваги:

$$WEI = H(X) - IV, \tag{2.5}$$

де $H(X)$ – вага доказів для конкретного атрибуту,

IV – середня інформаційна цінність.

Рівняння розрахунку матиме наступний вигляд:

$$«outlook» = -(9 \div 14) \times log(9 \div 14) - (5 \div 14) \times log(5 \div 14) - 0.693 = 0.247;$$

$$«temperature» = 0.029 \text{ біта};$$

$$«humidity» = 0.152 \text{ біта};$$

$$«windy» = 0.048 \text{ біта}.$$

Тому для рекурсивної побудови дерева перший обирається «outlook» як атрибут розбиття в корені дерева [32]. Це єдиний вибір, для якого один дочірній вузол є повністю чистим, і це дає йому значну перевагу перед іншими атрибутами. «humidity» є наступним найкращим вибором, оскільки вона створює більший дочірній вузол, який є майже повністю чистим.

Для випадку коли значення для двох вузлів будуть рівними, можна використати правило найменшого коефіцієнту, для того значення яке передбачається. Два вузли вважаються рівними, якщо вони мають однакові атрибути та однакові набори дочірніх вузлів. Це означає, що вони мають однакову структуру та однакові значення. Перевірка рівності вузлів включає рекурсивну перевірку рівності їх дочірніх вузлів. Якщо всі дочірні вузли відповідають, вузли вважаються рівними. Порядок дочірніх вузлів не враховується при порівнянні вузлів. Тобто, якщо два вузли мають однакові дочірні вузли, але в різному порядку, вони все одно вважаються рівними [33, с. 602].

На рисунку 2.12 показані можливості для подальшої побудови гілки у досягнутому вузлі, коли «outlook» - «sunny».

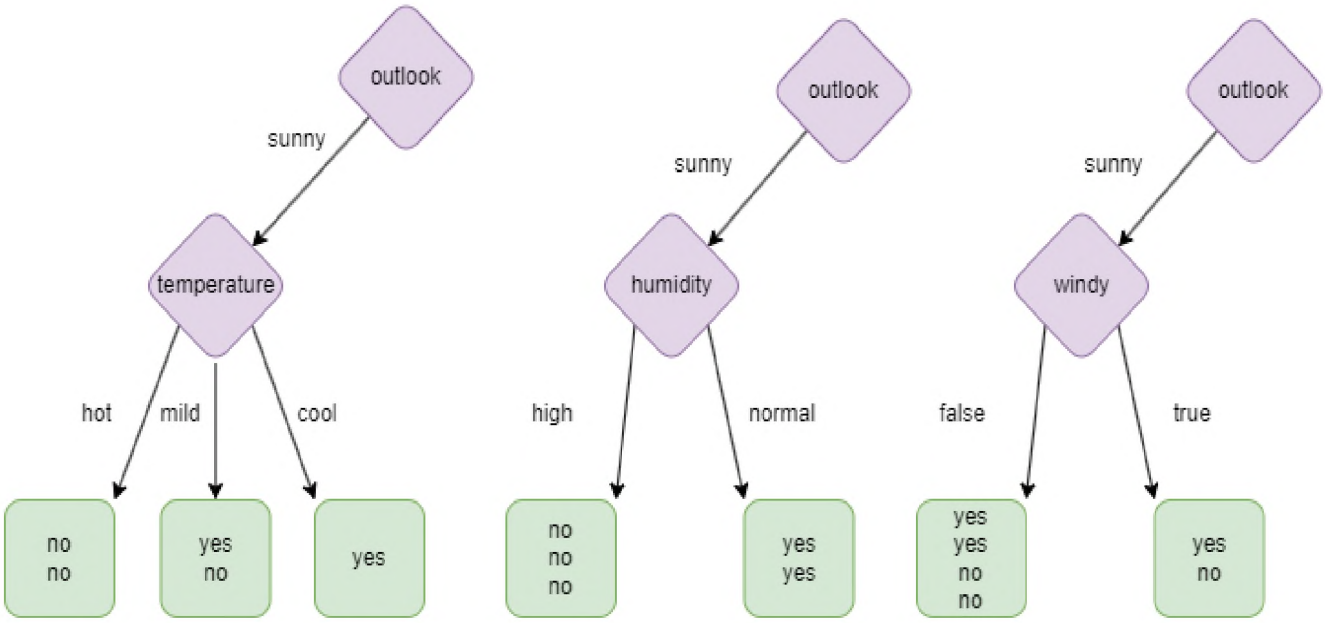


Рисунок 2.12 – Подальша побудова рекурсивного дерева

З нього стає зрозумілим те, що подальше розгалуження не дасть нічого нового.

Інший спосіб побудови дерева прийняття рішень називається «1R» і генерує однорівневе дерево рішень, виражене як набір правил, які перевіряють один конкретний атрибут [34]. «1R» є простим, дешевим методом, який часто забезпечує досить хороші правила для характеристики структури набору даних. Це тому, що структура, яка лежить в основі багатьох реальних наборів даних, є досить рудиментарною, і лише одного атрибута достатньо, щоб досить точно визначити клас екземпляра.

В основі лежить наступна ідея: створюються правила, які перевіряють один атрибут і відповідно розгалужуються. При цьому розгалуження може проводитися як за правилом алгоритмічних дерев так і за довільною структурою. Кожна гілка відповідає різному значенню атрибута. Найкраща класифікація для кожної гілки очевидна: потрібно використовувати клас, який найчастіше зустрічається в тренувальних даних. Тоді коефіцієнт помилок правил можна легко визначити, підрахувавши помилки, які виникають у навчальних даних, тобто кількість екземплярів, які не мають мажоритарного класу.

Кожен атрибут створює окремий набір правил, одне правило для кожного значення атрибута.

Для класифікації останнього стовпця, «play», «1R» розглядає чотири набори правил, по одному для кожного атрибута. Ці правила наведено у таблиці 2.1.

Таблиця 2.1 – Правила для побудови дерева за «1R»

Номер	Атрибут	Правило	Частка помилок	Загальна частка помилок
1	«outlook»	sunny = no	2/5	4/14
		overcast = yes	0/4	
		rainy = yes	2/5	
2	«temperature»	hot = no*	2/4	5/14
		mild = yes	2/6	
		cool = yes	1/4	
3	«humidity»	high = no	3/7	4/14
		normal = yes	1/7	
4	«windy»	false = yes	2/8	5/14
		true = no*	3/6	

Зірочка означає, що було зроблено випадковий вибір між двома рівно імовірними результатами. Для кожного правила вказано кількість помилок разом із загальною кількістю помилок для набору правил у цілому. «1R» обирає атрибут, який створює правила з найменшою кількістю помилок, тобто перший і третій набори правил.

Хоча це дуже елементарний метод тренування, «1R» вміщує як відсутні значення, так і числові атрибути. Він справляється з ними простими, але ефективними способами. Відсутнє розглядається як інше значення атрибута, тому, наприклад, якщо дані про погоду містили відсутні значення для атрибута «outlook», набір правил, згенерований у «outlook», визначить чотири можливі значення класу: по одному для «sunny», «overcast» та «rainy», а четверте за відсутнє.

Можна перетворити числові атрибути в номінальні за допомогою простого методу дискретизації [35]. Спочатку відсортувати навчальні приклади за значеннями числового атрибута від більшого до меншого. Це створює послідовність значень класу, або простіше кажучи числовий ряд. Наприклад, сортування числової версії даних погоди за значеннями температури дає набори послідовностей, які зображено у таблиці 2.2.

Таблиця 2.2 – Послідовність значень для «1R»

Номер послідовності	Значення температури	Значення класу
1	64	yes
2	65	no
3	68	yes
4	70	yes
5	71	yes
6	72	no
7	72	no
8	75	yes
9	75	yes
10	75	yes
11	80	no
12	81	yes
13	83	yes
14	85	yes

Дискретизація передбачає розділення цієї послідовності шляхом розміщення в ній точок розриву. Однією з можливостей є розміщувати точки зупину всюди, де змінюється клас, створюючи вісім категорій. Таким чином буде утворено співвідношення діапазонів до точок розриву.

Вибір точок розриву на півдорозі між прикладами з обох сторін розміщує їх на 64.5, 66.5, 70.5, 72, 77.5, 80.5 і 84. Однак два екземпляри зі значенням 72 викликають проблему, оскільки вони мають однакові значення температури, але поділяються на різні класи.

Найпростішим виправленням є переміщення точки розриву на 72 на один приклад вище, на 73.5, створюючи змішаний розділ, у якому значення «no» є мажоритарним класом. Це можна побачити в таблиці 2.3 у якій наведено отримані діапазони значень для класифікації набору даних, що була розглянута раніше.

Таблиця 1.3 – Таблиця розривів для діапазонів

Номер дискретної групи	Значення температури	Значення класу
1	64.5	yes
2	66.5	no
3	70.5	yes
4	72	no
5	73.5	yes
6	77.5	no
7	80.5	yes
8	84	no

Серйознішою проблемою є те, що ця процедура має тенденцію створювати велику кількість категорій. Метод «1R» природно схилятиметься до вибору атрибута, який розділено на багато категорій, оскільки це розділить набір даних на багато класів, що збільшить ймовірність того, що екземпляри матимуть той самий клас, що й більшість у своєму розділі.

Насправді граничний випадок – це атрибут, який має різне значення для кожного екземпляра, тобто атрибут ідентифікатор якого однозначно ідентифікує екземпляри, і це призведе до нульової частоти помилок у навчальному наборі, оскільки кожен розділ містить лише один екземпляр. Звичайно, сильно розгалужені атрибути зазвичай погано працюють у тестових випадках, тому ідентифікатор атрибуту ніколи не зможе правильно передбачити будь-які приклади за межами навчального набору. Це явище відоме як перенавчання [36].

Для «1R» перенавчання, ймовірно, відбудеться кожного разу, коли атрибут має велику кількість можливих значень. Тому при дискретизації числового атрибута приймається правило, яке диктує мінімальну кількість прикладів

мажоритарного класу в кожному розділі. Скажімо, встановлено максимум три. Це видаляє всі попередні розділи, крім двох.

Метод «1R» використовує єдиний атрибут як основу для своїх рішень і вибирає той, який працює найкраще. Інший простий трюк полягає в тому, щоб використовувати всі атрибути та дозволити їм приймати рішення, які є однаково важливими та незалежними один від одного, враховуючи клас. Це призводить до простої схеми, яка знову напрочуд добре працює на практиці.

Якщо уявити зведення погодних даних, отриманих шляхом підрахунку кількості разів, коли кожна пара атрибут-значення зустрічається з кожним значенням («yes» і «no») для «play», то можна побачити, що прогноз є позитивним для п'яти прикладів, два з яких мають значення класу «yes» і три з яких мають значення класу «no». Із дев'яти таких зіграних днів «outlook» має значення «sunny» на два, що дає шанси 2/9.

Тепер припустимо, що утворюється новий приклад зі значеннями. Загалом розглядається п'ять атрибутів, як однаково важливі незалежні змінні. В такому випадку при перемноженні відповідних дробів буде отримано наступний вираз для класу при значенні «yes»:

$$\frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14} = 0.0053.$$

Частки беруться відповідно до значень атрибутів для нового дня, а остаточне 9/14 – це загальна частка, яка представляє частку днів, у які клас «play» набуває значення «yes». Подібний розрахунок результату для «no» призводить до утворення нового рівняння:

$$\frac{3}{5} \times \frac{1}{5} \times \frac{4}{5} \times \frac{3}{5} \times \frac{5}{14} = 0.0206.$$

Це вказує на те, що для нового дня «no» більш імовірно, ніж «yes», при чому у чотири рази більш ймовірно. Числа можна перетворити на ймовірності, нормалізувавши їх так, щоб їх сума дорівнювала 1:

$$\frac{0.0053}{0.0053+0.0206} = 20.5\% \text{ - для ймовірності «yes»},$$

$$\frac{0.0206}{0.0053+0.0206} = 79.5\% \text{ - для ймовірності «no»}.$$

Цей простий та інтуїтивно зрозумілий метод базується на формулі умовної ймовірності Баєса. Правило Баєса [37] говорить, що якщо у вас є гіпотеза H і докази E, які підтверджують цю гіпотезу, то:

$$\Pr[H \mid E] = (\Pr[E \mid H] \times \Pr[H]) / \Pr[E], \tag{2.6}$$

де $\Pr[H]$ – ймовірність настання події H,

$\Pr[E|H]$ – позначає ймовірність E, залежну від іншої події H,

$\Pr E$ – кількісна ймовірність усіх можливих подій

Гіпотеза H полягає в тому, що значення «play» буде, скажімо, «yes», і $\Pr[H|E]$ становитиме 20,5%, як було визначено раніше. Свідчення E – це конкретна комбінація значень атрибутів для нового дня, «outlook» = «sunny», «temperature» = «cool», «humidity» = «high» та «windy» = «true». Назвемо ці чотири докази $E1$, $E2$, $E3$ та $E4$ відповідно. Припускаючи, що ці докази є незалежними (з урахуванням класу) їх кумулятивна ймовірність виходить множенням ймовірностей:

$$\Pr[yes|E] = \frac{\Pr[E_1 \,|yes] \times \Pr[E_2 \,|yes] \times \Pr[E_3 \,|yes] \times \Pr[E_4 \,|yes] \times \Pr[yes]}{\Pr[E]}.$$

$\Pr[yes]$ у кінці – це ймовірність результату «yes» без знання будь-яких доказів E, тобто без знання нічого про конкретний день, на який посилається, це називається попередньою ймовірністю гіпотези H. У цьому випадку це просто 9/14, тому що 9 із 14 прикладів навчання мали значення «yes» для «play». Підставляючи дроби для відповідних ймовірностей доказів, ми отримаємо:

$$\Pr[yes|E] = \frac{\frac{2}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{3}{9} \times \frac{9}{14}}{\Pr[E]}.$$

Тобто те саме значення, яке було розраховано раніше.

Цей метод має назву «Наївний Баєс» [38, с. 925], оскільки він заснований на правилі Баєса та «наївно» передбачає незалежність – множити ймовірності можна лише тоді, коли події незалежні. Припущення, що атрибути є незалежними (враховуючи клас) у реальному житті, безумовно, є спрощеним. Але, незважаючи на зневажливу назву, «Наївний Баєс» працює дуже добре, коли тестується на фактичних наборах даних, особливо в поєднанні з деякими процедурами вибору атрибутів, які усувають надлишкові, а отже, незалежні атрибути.

Одна річ, яка може піти не так з використанням цього методу, полягає в тому, що якщо конкретне значення атрибута не зустрічається в навчальному наборі в поєднанні з кожним значенням класу, все піде не так. У прикладі припустимо, що тренувальні дані відрізняються, і значення атрибута «outlook» = «sunny» завжди було пов'язане з результатом «no». Тоді ймовірність «outlook» = «sunny» для події «yes», тобто:

$$Pr[\text{«outlook»} = \text{«sunny»}|yes],$$

буде дорівнювати нулю, і оскільки інші ймовірності множаться на це, остаточна ймовірність також буде дорівнювати нулю, незалежно від того, наскільки різні результати ми отримали. Імовірності, які дорівнюють нулю, мають право вето на інші. Але помилка легко виправляється незначними коригувами в методі обчислення ймовірностей з частот.

Наприклад, для «play» = «yes», «outlook» = «sunny» для двох прикладів, «outlook» = «overcast» для чотирьох і «outlook» = «rainy» для трьох, тобто це дає події ймовірності

$$\frac{2}{9}, \frac{4}{9} \text{ і } \frac{3}{9}.$$

Замість цього можна додати одиницю до кожного чисельника та компенсувати, додавши трійку до знаменника, отримавши наступні ймовірності

$$\frac{3}{12}, \frac{5}{12} \text{ і } \frac{4}{12}.$$

Це гарантує, що значення атрибута, яке зустрічається нуль разів, отримує ймовірність, відмінну від нуля, хоча й малу. Стратегія додавання одиниці до кожного числа є стандартною технікою, яка називається оцінкою Лапласа на честь видатного французького математика XVIII століття П'єра Лапласа. Хоча це добре працює на практиці, немає особливої причини додавати одиницю до підрахунків: замість цього можемо вибрати невелику константу μ і використовувати її:

$$\frac{2+\mu/3}{9+\mu}, \frac{4+\mu/3}{9+\mu}, \frac{3+\mu/3}{9+\mu}.$$

Значення μ, яке було встановлено рівним трьом, фактично забезпечує вагу, яка визначає, наскільки впливовими є апріорні значення

$$\frac{1}{3}, \frac{1}{3} \text{і} \frac{1}{3}.$$

для кожного з трьох можливих значень атрибута. До того ж, немає особливої причини ділити μ на три рівні частини в чисельниках: замість цього можна використати інші константи p_1, p_2 і p_3 сума яких дорівнює одиниці.

По суті, ці три числа є апріорними ймовірностями значень атрибута «outlook» при «sunny», «overcast» та «rainy» відповідно.

$$\frac{2 + \mu/p_1}{9 + \mu}, \frac{4 + \mu/p_2}{9 + \mu}, \frac{3 + \mu/p_3}{9 + \mu}.$$

Тепер це повністю Байєсівське формулювання, де попередні ймовірності приписуються всьому, що знаходиться в полі зору [39]. Його перевага полягає в тому, що він повністю точний, але недолік у тому, що зазвичай незрозуміло, як ці попередні ймовірності слід призначати. На практиці попередні ймовірності не мають великого значення за умови, що існує достатня кількість екземплярів навчання, і все можна просто оцінити за допомогою оцінки Лапласа, ініціалізуючи всі підрахунки до одиниці замість нуля.

Однією з дійсно приємних речей Байєсівського формулювання є те, що відсутні значення взагалі не є проблемою. Наприклад, якби значення прогнозу було відсутнє, обчислення просто пропустило б цей атрибут, даючи результат на ці два числа окремо значно вищі, ніж були раніше, тому що один із дробів відсутній. Але це не проблема, тому що в обох випадках є відсутня частка, і ці ймовірності підлягають подальшому процесу нормалізації. Це дає ймовірність

$$\text{«}yes\text{»} = 41\%;$$

$$\text{«}no\text{»} = 59\%.$$

Якщо значення відсутнє в навчальному екземплярі, воно просто не включається при підрахунку частоти, а відношення ймовірності базується на кількості значень, які фактично зустрічаються, а не на загальній кількості екземплярів.

Числові значення зазвичай розглядаються, припускаючи, що вони мають «нормальний» або «гаусівський» розподіл ймовірностей. В такому випадку для номінальних атрибутів відбудеться обчислення кількості, як і раніше, а для

числових просто підрахуються значення, які виникли. Тоді, хоча кількість номінальних атрибутів нормалізована у ймовірності, буде обчислине середнє значення та стандартне відхилення для кожного класу та кожного числового атрибута.

Таким чином, середня «temperature» в таких випадках становить 73, а її стандартне відхилення – 6,2. Середнє – це просто середнє попередніх значень, тобто сума, поділена на кількість значень. Стандартне відхилення – це квадратний корінь із вибіркової дисперсії, який можна обчислити шляхом віднімання середнього від кожного значення, зведення результату в квадрат, додавання їх разом, а потім ділення на одиницю менше, ніж кількість значень.

Розглянемо ще одне доповнення до попереднього алгоритму. Однією із найпоширеніших класифікацій є класифікація документів, де кожен екземпляр представляє документ, а клас екземпляра є темою документа. Документи можуть бути новинами, а класи – внутрішніми новинами, закордонними новинами, фінансовими новинами та спортом. Документи характеризуються словами, які в них з'являються, і один із способів застосувати класифікацію тексту до класифікації документів – розглядати наявність або відсутність кожного слова як логічний атрибут. Наївний Байєс є популярним методом для цієї концепції, оскільки він дуже швидкий і досить точний.

Однак він не враховує кількість входжень кожного слова, що є потенційно корисною інформацією при визначенні категорії документа. Натомість документ можна розглядати як мішок слів – набір, який містить усі слова в документі, де слово повторюється кілька разів (технічно набір включає кожного зі своїх членів лише один раз, тоді як мішок може мати повторювані елементи). Частоти слів можна застосувати, використовуючи модифіковану форму наївного Баєса, яку іноді описують як багато іменну наївну Байєсовську мову.

Припустимо, що $n_1, n_2, \ldots n_i$ – це кількість разів, коли слово зустрічається в документі, а $P_1, P_2, \ldots P_k$ – це ймовірність отримання слова і під час вибірки з усіх документів категорії H. Припустимо, що ймовірність не залежить від контексту слова та позиції в документі. Ці припущення призводять до мультиноміального

розподілу ймовірностей документів. Для цього розподілу ймовірність документа E з урахуванням його класу H – іншими словами, формула для обчислення ймовірності $Pr[E|H]$ за правилом Баєса:

$$\Pr[E|H] = N! \times \prod_{i=1}^{k} \frac{p_k}{n_i!}, \quad (2.7)$$

де $N!$ – кількість слів у документі;

n_i – кількість повторення слова;

p_k – ймовірність отримання слова.

Причиною факторіалів є врахування того факту, що порядок появи кожного слова не має значення відповідно до моделі сумки слів. P_i оцінюється шляхом обчислення відносної частоти слова і в тексті всіх навчальних документів, що належать до категорії H. Фактично, має бути додатковий атрибут, який дає ймовірність того, що модель для категорії H генерує документ, довжина якого однакова, як і довжина E.

Наприклад, припустімо, що в словнику є лише два слова, «жовтий» і «синій», і певний клас документа H має:

$$Pr[yellow|H] = 75\%;$$
$$Pr[blue|H] = 25\%.$$

Припустимо, що E є синьо-жовто-синім документом довжиною $N = 3$ слова.

Є чотири можливі «патчі» з трьох слів. Перший це {жовтий; жовтий; жовтий}, і його ймовірність згідно з попередньою формулою дорівнює

$$\Pr[\{\text{жовтий; жовтий; жовтий;}\}]\,|H = 3! \times \frac{0.75^3}{3!} \times \frac{0.25^0}{0!} = \frac{27}{64}.$$

Ймовірність появи інших трьох, можна розрахувати за допомогою аналогічного рівняння. Так для «патча» {синій; синій; синій}, рівняння прийматиме наступний вигляд:

$$\Pr[\{\text{синій; синій; синій}\}|H] = \frac{1}{64}.$$

Для «патча» {жовтий; жовтий; синій}, це рівняння відповідно виглядатиме так:

$$\Pr[\{\text{жовтий; жовтий; синій}\}|H] = \frac{27}{64}.$$

Для «патча» {жовтий; синій; синій}, це рівняння виглядатиме незмінно і матиме вигляд:

$$\Pr[\{\text{жовтий; синій; синій}\}|H] = \frac{9}{64}.$$

Тут E відповідає останньому слову (в мішку слів порядок не має значення), тому ймовірність того, що він буде згенерований жовто-синьо-синьою моделлю документа, становить $9/64$, або 14%.

Висновки до розділу 2

У другому розділі були розглянуті проблеми які виникають під час класифікації тексту при контрольованому навчанні. Був більш детально розглянутий формат фалів arff, який використовується фреймворком WEKA та було проведене первинне ознайомлення із набором даних який використовуватиметься у при навчанні моделі ІС для класифікації тексту.

Було розглянуто рішення, які використовуються для вирішення цих проблем, такі як дерева прийняття рішень. Проведений детальний аналіз побудови рекурсивних дерев рішень, та розглянуті математичні структури які використовуються фреймворком під час класифікації тексту за допомогою дерев прийняття рішень.

Також, у розділі наведена топологія побудованої Байєсівської нейронної мережі із використанням наївного Байєсівського класифікатора. Наведені особливості даного типу нейронних мереж.

За допомогою аналітичного методу була перевірена дієвість наївного Байєсівського методу для класифікації документів, тексту та створення текстових словників. Цей метод використовується під час створення словників деревами прийняття рішень.

РОЗДІЛ 3
РЕКОМЕНДАЦІЇ ЩОДО ПРАКТИЧНОЇ РЕАЛІЗАЦІЇ ТЕХНОЛОГІЇ КЛАСИФІКАЦІЇ ТЕКСТУ

3.1 Використання самостійної версії фреймворку WEKA

Цей розділ присвячено класифікації тексту із використанням фреймворку WEKA. Фреймворк можна використовувати як самостійний додаток, що містить колекцію алгоритмів машинного навчання, які можуть бути використані для вирішення різних завдань так і у якості вбудованого API який розробник може використати у власному вебдодатку.

Розглянемо як виглядає інтерфейс користувача у самостійному додатку. На рисунку 3.1 зображено загальний інтерфейс за допомогою якого можна дізнатися інформацію про набір даних.

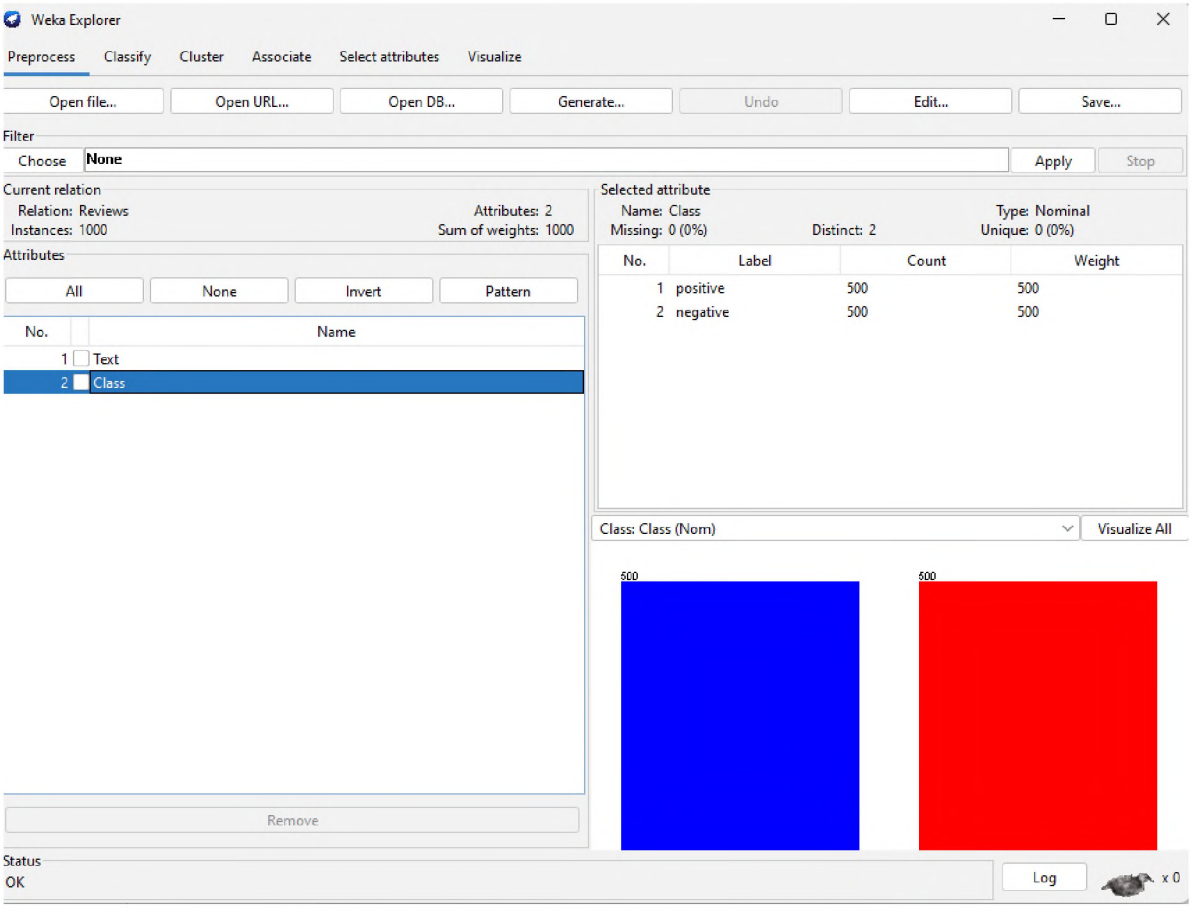


Рисунок 3.1 – Вигляд інтерфейсу WEKA

У правому нижньому куті інтерфейсу зображено інформацію про кількість даних у наборі (два стовпчики, які містять по п’ятсот відгуків відвідувачів кожний). Кожний стовпчик має власний колір, який означає емоційне забарвлення відгука. Синій – відгук позитивний, тоді як червоний колір означає негативний відгук.

Набір даних для кваліфікаційної роботи представляє собою файл із розширенням .csv, який було завантажено із безкоштовного сховища даних KAGGLE [40], яке містить реальні набори даних для навчання НМ. Набір даних було конвертовано у файл формату arff за допомогою інструментів фреймворку.

Набір даних складається із двох колонок. В першій колонці міститься текст відгуку, у другій колонці міститься забарвлення відгуку, яке має значення «positive» та «negative». На рисунку 3.2 відображено набір даних, відтворений за допомогою вбудованого переглядача arff файлів, який вбудовано у самостійний додаток.

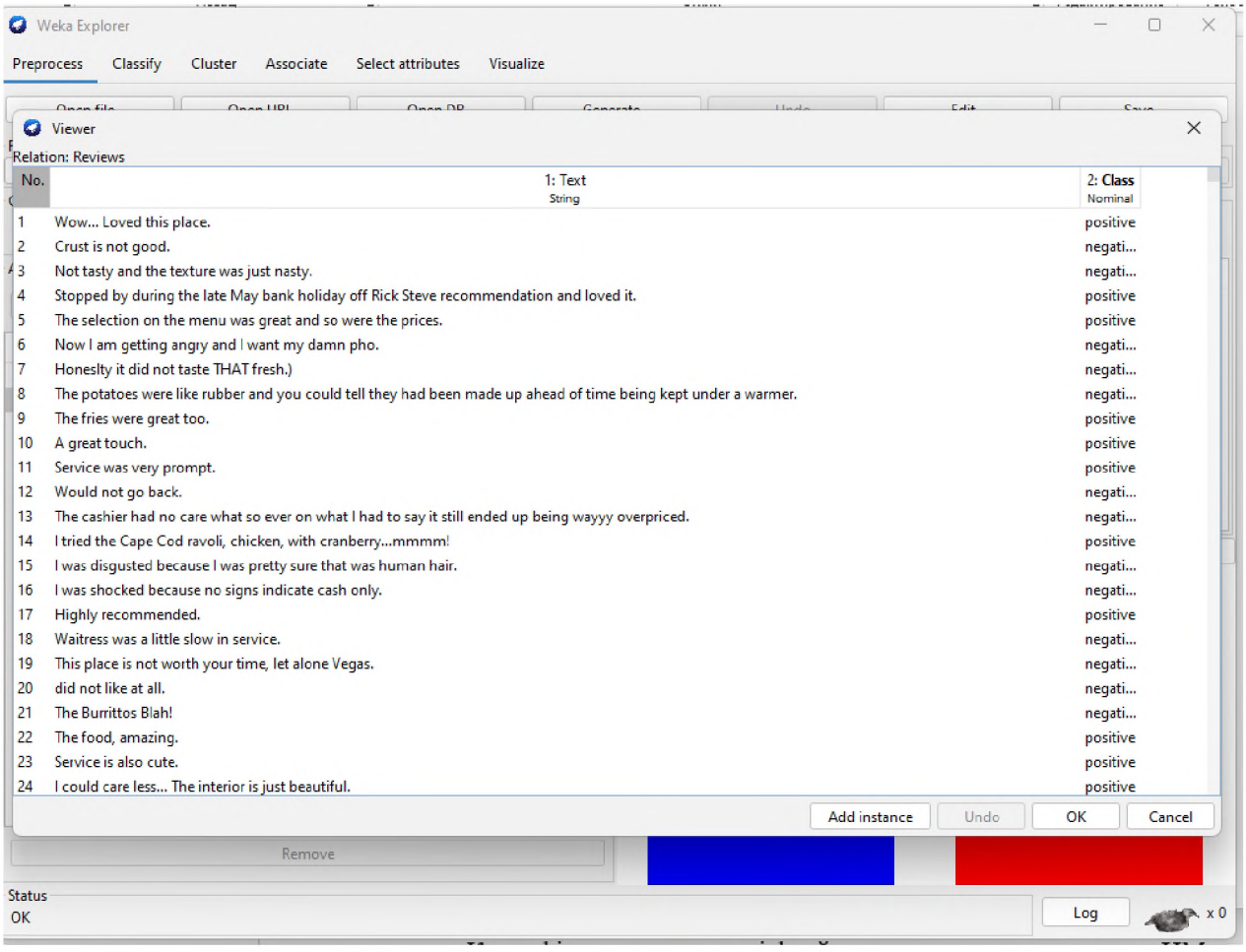


Рисунок 3.2 – Представлення набору даних

Для того щоб, належним чином обробити набір даних, користувач може скористатися кнопкою «Classify» (варто звернути увагу, що фреймворк не підтримує інших мов ані для інтерфейсу, ані для наборів даних, окрім англійської) яку зображено на рисунку 3.3.

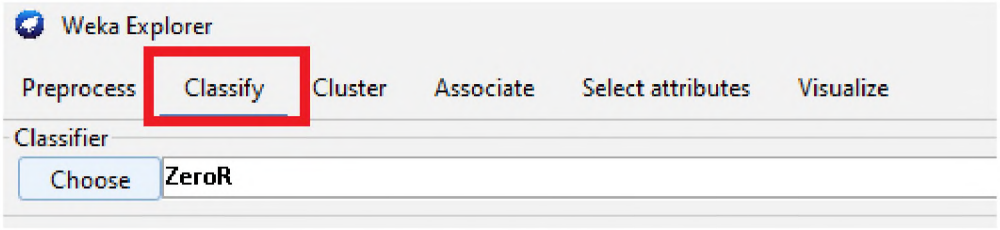


Рисунок 3.3 – Кнопка вибору класифікатора

Класифікатор у контексті фреймворку варто розглядати як модель НМ, яку потрібно навчити та протестувати.

Після входу до меню вибору, потрібно вибрати класифікатор, який найбільше підходить під конкретну задачу. Для даної кваліфікаційної роботи буде використаний класифікатор, основу якого було розглянуто в попередньому розділі.

Це наївний байєсівський класифікатор, який в своїх алгоритмах використовує поєднання техніки Лапласа та метод класифікації Байєса. На рисунку 3.4 зображено вікно вибору доступних класифікаторів.

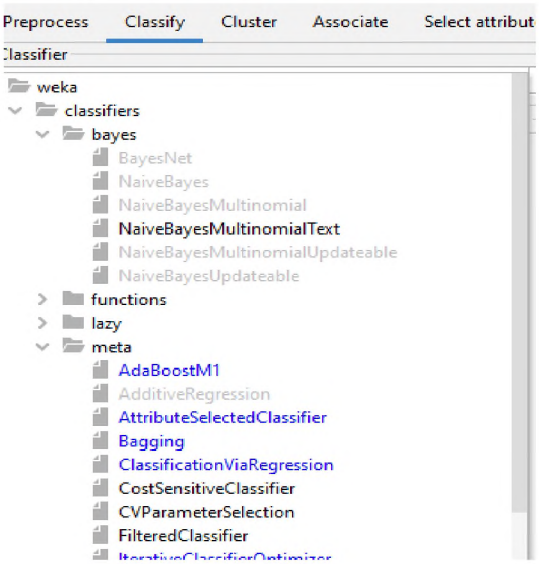


Рисунок 3.4 – Вікно із переліком класифікаторів

Для класифікації або ж сегментації тексту [41, с. 148], WEKA може використовувати майже будь-який класифікатор із переліку вище. Обравши класифікатор, користувач може викликати його, натиснувши на ньому лівою клавішею мишки. Фреймворк WEKA дуже швидко навчає класифікатор на наборі даних.

Класифікатор NaiveBayesMultinomialText, який використовується для класифікації тексту, створює так звану «торбу слів», так як дерево рішень не достатньо оптимізоване для вирішення цієї задачі [42, с. 152]. Інше слово яке теж можна використати, для пояснення процесу, це – словник. У цій «торбі» класифікатор відмічає скільки разів, яке слово зустрічається під час процесу класифікації і надає йому значення ваги відповідно до атрибуту «positive» чи «negative». На рисунку 3.5 можна побачити загальну інформацію по створеному словнику.

```
=== Run information ===

Scheme:       weka.classifiers.bayes.NaiveBayesMultinomialText
Relation:     Reviews
Instances:    1000
Attributes:   2
              Text
              Class
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

Dictionary size: 792

The independent frequency of a class
--------------------------------------
positive        501.0
negative        501.0
```

Рисунок 3.5 – Результат навчання класифікатора

Отриману інформацію можна проаналізувати, та виділити наступні найважливіші аспекти:

– Scheme: назва класифікатору який було використано для класифікації. У даному випадку – це NaiveBayesMultinomialText.

– Relation: назва набору даних – Reviews (з англ. «*відгуки*»).

– Instances: кількість екземплярів для класифікації (кількість відгуків користувачів) – загальна кількість становить одну тисячу.

– Attributes: кількість атрибутів. Загалом набір даних містить п’ятсот входжень для атрибуту «positive» і стільки ж для атрибуту «negative» відповідно.

– Dictionary size: розмір «торби слів». Усього набір даних містить 792 слова.

Далі у цьому ж вікні є можливість побачити як були розподілені слова у «торбі слів». На рисунку 3.6 зображений фрагмент списку класифікованих слів.

```
The frequency of a word given the class
----------------------------------------
   positive        negative
<laplace=1>             3.0     half
        4.0             5.0     chips
<laplace=1>             3.0     awful
        3.0     <laplace=1>     owners
       10.0            15.0     your
<laplace=1>             3.0     building
<laplace=1>             3.0     without
        3.0     <laplace=1>     setting
<laplace=1>             4.0     these
        3.0             3.0     tea
        5.0             3.0     Everything
        3.0     <laplace=1>     music
       11.0            27.0     would
<laplace=1>             6.0     poor
        5.0            11.0     because
<laplace=1>             3.0     Maybe
       11.0     <laplace=1>     excellent
        5.0             7.0     &
<laplace=1>             3.0     fairly
       10.0             9.0     -
<laplace=1>             6.0     1
        3.0             7.0     2
<laplace=1>             3.0     3
        8.0             3.0     selection
        3.0     <laplace=1>     stop
<laplace=1>             3.0     4
```

Рисунок 3.6 – Класифіковані слова і значення їх частоти

На основі цього списку класифікатор буде приймати подальші рішення щодо класифікації того чи іншого слова. Цей словник є обов'язковим елементом для розгляду результатів, так як спираючись на них, можна корегувати налаштування, тим самим контролюючи переміщення слів між атрибутами.

Останньою частиною, яку демонструє WEKA у вікні огляду результатів є загальний підсумок навчання моделі, як зображено на рисунку 3.7.

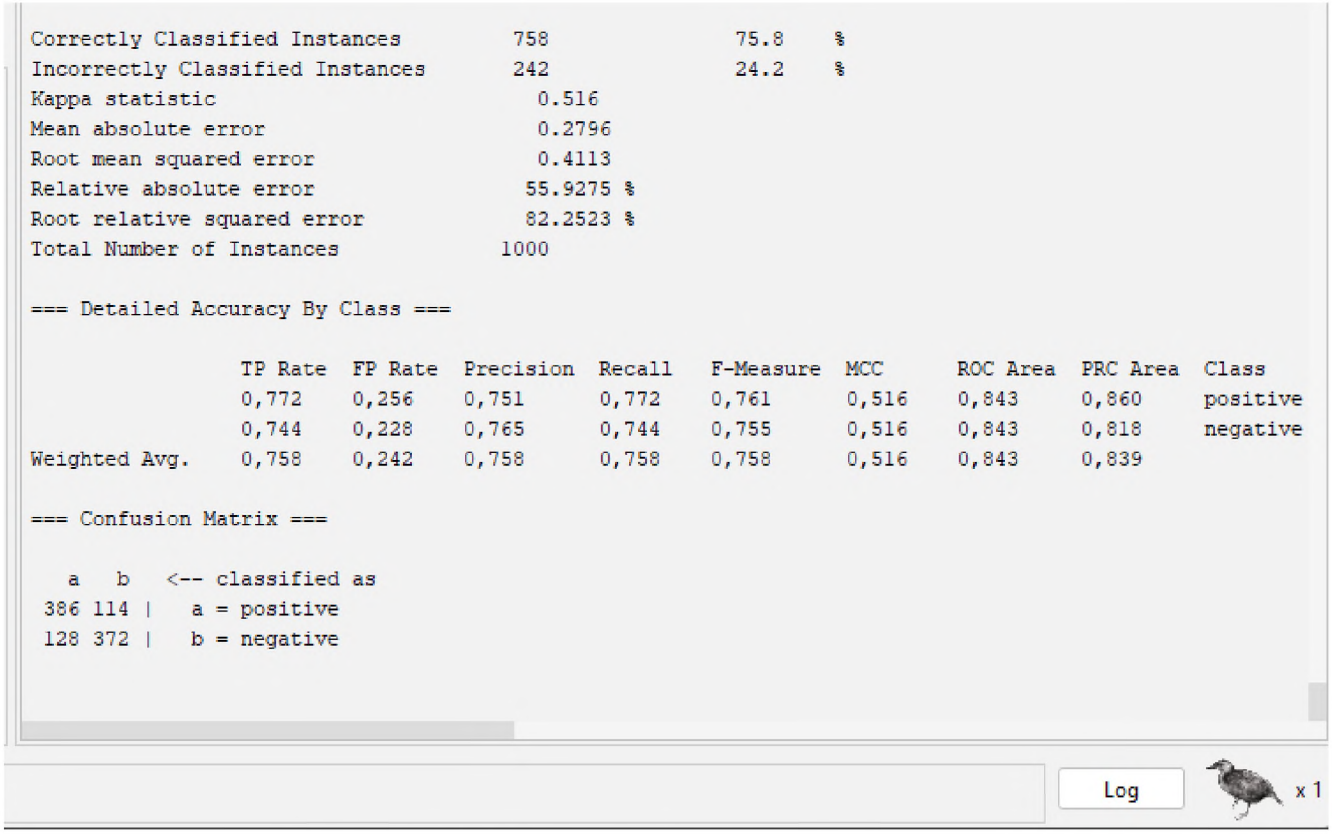


Рисунок 3.7 – Інформація щодо навчання моделі

Аналізуючи останню частину, користувач має можливість визначити наскільки точно було навчену модель яку він використовує. Проаналізувавши вивід ПЗ можна зробити висновок, що точність з якою модель була навчена становить сімдесят п'ять цілих, вісім десятих відсотка. В той час як кількість помилок становить двадцять чотири і дві десятих відсотка.

Також проаналізувавши матрицю помилок, яка має назву «Confusion Matrix», можна зробити висновок, що із п'ятисот екземплярів, які відносяться до

позитивних відгуків сто чотирнадцять, були класифіковані як негативні. І навпаки, частка сто двадцять вісім негативних відгуків були класифіковані як позитивні.

Одним із поширених варіантів аналізу даних, які були класифіковані НМ [43, с. 7] є аналіз за допомогою візуалізації даних, який надає можливість візуально побачити розподіл даних по визначених атрибутах.

На рисунку 3.8 зображено аналіз розподілу слів для набору даних із відгуками відвідувачів ресторанів, що дає змогу передбачити відхилення та адекватність моделі [44, с. 31].

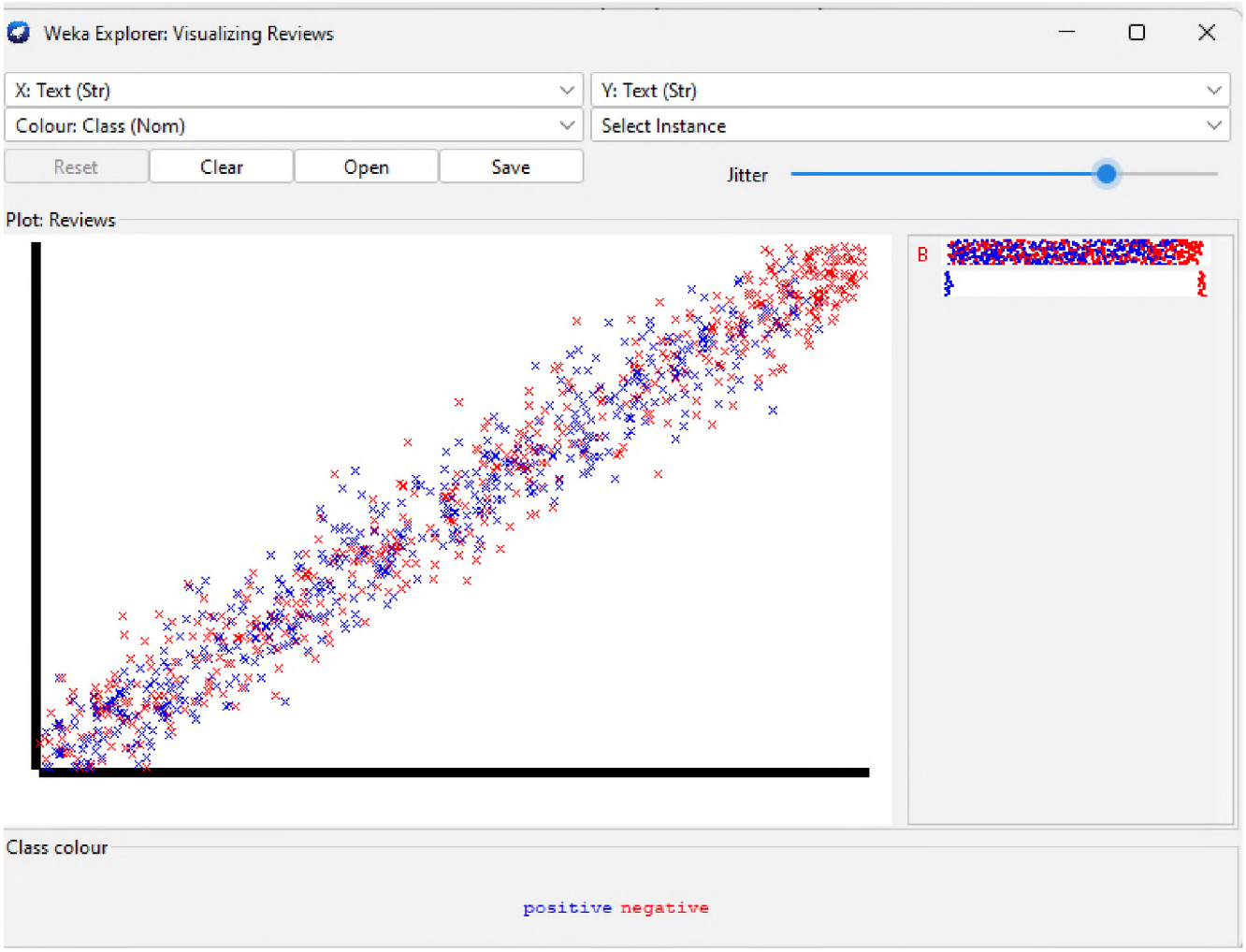


Рисунок 3.8 – Візуалізація розподілу даних навчання НМ

Тестування моделі НМ, яка була навчена можна провести за допомогою вбудованого меню, яке надає можливість розділити набір навчальних даних за пропорціями, використати тестовий набір навчальних даних [45, с. 45], або ж

зробити перехресну валідацію на навчальному наборі даних. Всі ці способи дозволяють отримати протестований варіант НМ.

Проаналізувавши можливості які надаються самостійною версією фреймворку WEKA, можна зробити висновок, що це дуже потужний інструмент для навчання та класифікації НМ.

3.2 Вебдодаток для класифікації тексту

Вебдодаток що використовує фреймворк WEKA, було розроблено на мові Java, для надання можливостям які надаються фреймворком більшої гнучкості та можливості бути використаними без попереднього встановлення самостійної версії.

На рисунку 3.9 зображено головне вікно вебдодатку, через яке виконується введення відгуку та в якому ж і відбувається отримання результату.

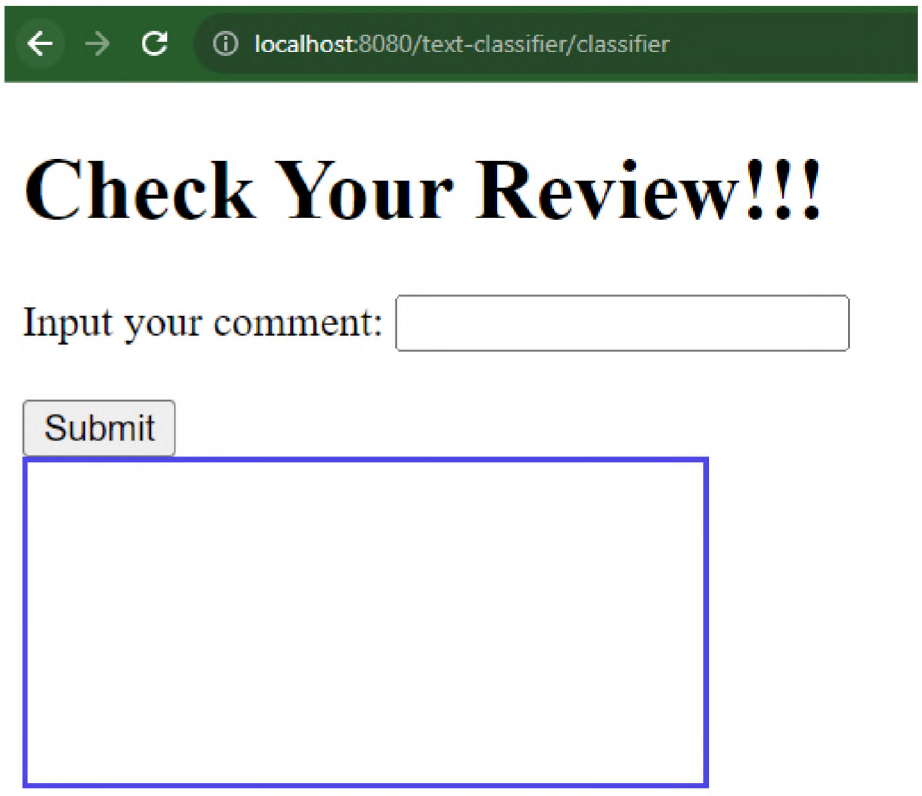


Рисунок 3.9 – Головне вікно веб додатку

Отримання результатів відбувається одразу, після введення відгуку на англійській мові та натискання кнопки «Submit». Усе опрацювання, навчання, обробка та видача результатів відбуваються автоматично, таким чином користувачу не потрібно виконувати зайві або непотрібні дії і таким чином швидкість виконання основної задачі стає значно простішою.

Реалізація вебдодатку за допомогою API досить складна, через те що фреймворк WEKA має свої специфічні алгоритми, типи даних та формат файлів.

Вивід результатів(недоступний для користувача, але доступний для розробника) майже не відрізняється від того, що надається фреймворком. На рисунку 3.10 зображено частковий вивід логів роботи вебдодатку.

```
Classifier...:\nweka.classifiers.bayes.NaiveBayesMultinomialText

===========

-P 0 -M 3.0 -norm 1.0 -lnorm 2.0 -stopwords-handler weka.core.stopwords.Null
 weka.core.stemmers.NullStemmer

===========

Dictionary size: 792

The independent frequency of a class
------------------------------------
positive    501.0
negative    501.0

The frequency of a word given the class
---------------------------------------
   positive    negative
<laplace=1>         3.0 half
        4.0         5.0 chips
<laplace=1>         3.0 awful
```

Рисунок 3.10 – Вивід логів роботи застосунку.

На рисунку 3.11 зображено результат розпізнавання відгуку відвідувача з використанням розробленого веб додатку.

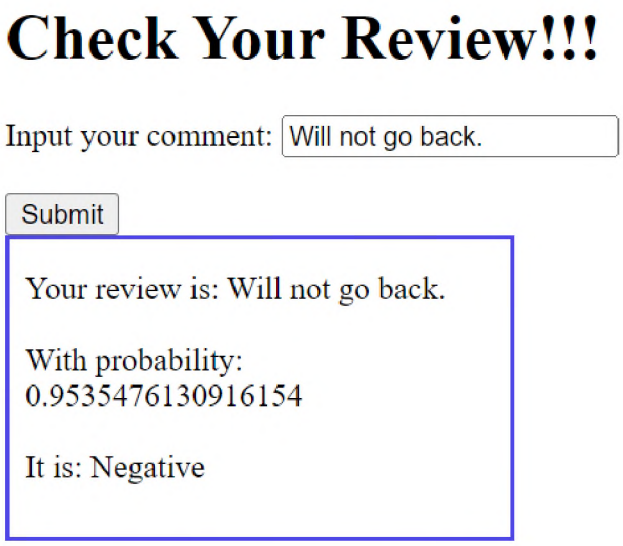


Рисунок 3.11 – Вивід результатів роботи веб додатку

Користувач буде отримувати інформацію про те який відгук був введений – з метою уникнення ситуації коли відсутнє розуміння того, який відгук був відправлений попередньо.

Також результат повертає ймовірність із якою НМ класифікувала відгук, який було надіслано користувачем та який сам це відгук з урахуванням результатів отриманих після класифікації тексту та навчанні НМ на тренувальному наборі даних.

3.3 Техніко економічне обґрунтування прийнятих рішень

Однією із причини для вибору фреймворку WEKA було те, що даний фреймворк надається безкоштовно, без використання прихованих підписок чи абонентської плати. З урахуванням поточної цінової політики Jet Brains на ринку [46], автору знадобиться 614,48 грн., ($16.90 за курсом НБУ на 01.10.2023), на придбання ліцензії, для середовища розробки IntelliJ Idea та ноутбук чи персональний комп’ютер на якому можна розробити вебдодаток. Прямі витрати на впровадження запропонованої інформаційної технології можна обрахувати за формулою:

$$В_п = В_{тз} + В_{ппз} + В_{псп} + В_у, \quad (3.1)$$

де $В_п$ – прямі витрати на проект впровадження інформаційної технології;

$В_{тз}$ – витрати на придбання технічного забезпечення;

$В_{ппз}$ – витрати на придбання програмного забезпечення;

$В_{псп}$ – витрати за послуги які виконують стороні підприємства;

$В_у$ – витрати на управління інформаційними технологіями.

Таблиця 3.1 – Кошторис створення вебдодатку

Перелік складових	Вартість у гривнях
1. Ліцензія IntelliJ Idea	615
2. Персональний комп`ютер	23485
3. Робота програміста	10000
4. Хостинг вебсерверу	1000
Разом	35100

Підставивши у рівняння дані наведені в таблиці можна провести остаточний розрахунок. Таким чином, обрахована сумарна вартість впровадження інформаційної технології становить тридцять п’ять тисяч сто гривень.

Висновки до розділу 3

У розділі було розглянуте використання самостійної версії фреймворку та її основні функції, які потрібні для початку класифікації тексту та подальшого аналізу даних.

Також самостійна версія фреймворку набагато легше піддається налаштуванню та має вбудовані функції повернення даних та налаштувань, у випадку, якщо щось піде не так і налаштування виявляться не коректними.

Також у розділі були розглянуті можливості розробленого веб додатку та продемонстрована його працездатність та можливості. Був проведені фінансові розрахунки та наведений кошторис, який потрібен для того щоб розуміти економічну собівартість.

ВИСНОВКИ

Для навчання НМ у цій кваліфікаційній роботі був в використаний фреймворк WEKA. Основна відмінність в бібліотек, полягає в тому, що даний фреймворк використовує свій формат даних для навчання НМ. Також він набагато краще працює із файлами даних аніж із одинарними входженнями.

Кваліфікаційна робота присвячена розгляду проблеми класифікації тексту – а саме проблемі передбачення результатів після навчання НМ.

Основним недоліком фреймворку є специфічний формат даних, який працює лише із кодуванням UTF-8, що робить неможливим роботу із символами кирилиці. На додаток трансформація у arff формат має певні недоліки, адже потрібно контролювати процес перетворення, через те що символом для розділу атрибутів є кома. Тобто для коректної реалізації потрібно враховувати цей аспект та змінювати налаштування безпосередньо у самому фреймворку чи в коді вебдодатку, що розробляється.

Також до недоліків фреймворку можна віднести високий поріг входження, який збільшує час на розуміння того як саме фреймворк працює.

Зручність фреймворку розкривається саме в самостійній версії яка надає можливості для навчання та тестування, аналізу та візуалізації. При використанні API для розробки коду, ці всі речі потрібно додатково корегувати, так як типи даних дуже специфічні і побудова логіки за рахунок цього може бути дуже сильно ускладнена.

Під час дослідження проведеного у даній кваліфікаційній роботі був розроблений веб додаток який здатен розрізняти відгуки які користувач залишає на веб сайті.

Для розробки вебдодатку були використані такі інструменти як: мова програмування Java, фреймворк Spring, шаблонізатор mustache, стандартні бібліотеки Java.

Сам вебдодаток є можливість завантажувати на веб сервер, після чого він стає доступним для користувачів інтернету. Для додатку відсутні будь-які регіональні

обмеження. Мова вебдодатку – англійська, мова відгуків які приймає вебдодаток теж англійська.

Набори даних які були використані для навчання НМ знаходяться у вільному доступі, та складаються із файлів у форматі arff. В кваліфікаційній роботі були надані приклади того який вигляд ці файли мають, опис конвертації файлів із формату json та csv знаходиться на офіційному сайті фреймворку. Для кваліфікаційної роботи був використаний набір даних, що знаходиться у вільному доступі.

Веб додаток має налаштовану конфігурацію нейронної мережі, так як даний спосіб є найпростішим і використання параметрів налаштування без попереднього ознайомлення із можливостями фреймворку не бажане. Саме тому можливості будь яких користувацьких налаштувань у вебдодатку було відключено.

В кваліфікаційній роботі представлений результат роботи класифікації тексту із використанням наївного байєсівського класифікатору та методу Лапласа.

Побудована НМ працює із класифікацією тексту та може бути використана у інших додатках, що використовують фреймворк WEKA. Розроблена НМ не є сумісною із іншим ПЗ, через те що використовується унікальний тип притаманний лише фреймворку WEKA.

Таким чином, результатом роботи є: створена модель класифікатору тексту; розроблені рекомендацій щодо використання технології класифікації тексту на основі нейронної мережі. Вони можуть бути використані для подальших досліджень за даною тематикою та при проектуванні мобільних додатків.