Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційних радіотехнологій і технічного захисту інформації
(повна назва)
Кафедра Радіотехнологій інформаційно-комунікаційних систем
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

ГЮІК.ХХХХХХ.001ПЗ
(позначення документа)

АНАЛІЗ ТА ДОСЛІДЖЕННЯ МЕТОДІВ ТА ЗАСОБІВ НАВЧАННЯ БАЙЄСІВСЬКИХ МЕРЕЖ
(тема)

Виконав:
студент II курсу, групи ІКТм-20-1
Болгов Д. Р.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Освітня програма Інформаційно-комунікаційні технології
(повна назва освітньої програми)

Керівник проф. Кузьомін О. Я.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ О.І. Цопа
(підпис) (прізвище, ініціали)

2021 р.

Не містить відомостей заборонених для відкритого публікування.

Керівник _____ Кузьомін О.Я.

Студент _____ Болгов Д. Р.

Харківський національний університет радіоелектроніки

Факультет Інформаційних радіотехнологій і технічного захисту інформації

Кафедра Радіотехнологій інформаційно-комунікаційних систем

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Освітня програма Інформаційно-комунікаційні технології
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри_____
(підпис)

"_____" _____ 2021 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Болгову Дмитру Руслановичу
(прізвище, ім'я, по батькові)

1. Тема роботи Аналіз та дослідження методів та засобів навчання байєсівських мереж затверджена наказом по університету від 05 листопада 2021 р. №1648Ст
2. Термін подання студентом роботи до екзаменаційної комісії 9 грудня 2021 р.
3. Вихідні дані до роботи розробити штучний інтелект, що буде прогнозувати наявність хвороби
4. Перелік питань, що потрібно опрацювати в роботі аналіз предметної області, постановка задачі, аналіз існуючих алгоритмів розв'язання задачі, вибір алгоритму розв'язання задачі, розробка алгоритму функціонування, розробка алгоритму прогнозування хвороби
5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів): діаграма логістичної регресії, діаграма SVM, діаграма розсіювання невеликого надуманого набору даних, листінг коду

6. Консультанти розділів роботи

Найменування розділу	Консультант (посада, прізвище, ім’я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Основна частина	проф. Кузьомін О.Я.		

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Ознайомлення із завданням. Уточнення ТЗ*	01.09.21	вик.
2	*Підбір літератури за темою роботи*	13.09-22.09.21	вик.
3	*Теоретичний розділ*	23.09-10.10.21	вик.
4	*Проектний розділ*	11.10-25.10.21	вик.
5	*Оформлення презентаційного матеріалу,*	26.10-30.11.21	вик.
	підготовка до захисту у ЕК		

Дата видачі завдання __1__ ___вересня___ 2021 р.

Студент _____ Болгов Д. Р.
(підпис) (прізвище та ініціали)

Керівник роботи _____ проф. Кузьомін О.Я.
(підпис) (посада, прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до магістерської кваліфікаційної роботи: 72 с., 9 табл., 17 рис., 2 додатки, 16 джерел інформації.

БАЗА ДАНИХ, AI, БАЙЭС, АЛГОРИТМ, МЕДИЦИНА, ПРОГНОЗУВАННЯ, ХВОРОБА

Об’єктом досліджень є технології штучного інтелекту, а саме алгоритму Байєса для прогнозування захворювання.

Предметом досліджень є використання технології алгоритму Байєса для прогнозування захворювання.

Мета досліджень – розробка штучного інтелекту на основі алгоритму Байєса.

Методи дослідження – створення, тестування та впровадження нових технологій для прогнозування захворюваннь.

В результаті проведених досліджень вирішено задачу створення алгоритму байєса за допомогою мови Python. Отримані результати використовуються, як нові можливості прогнозування захворювання.

Мова програмування – Python. Пропонована розробка є корисною для вирішення різноманітних проблем, пов’язаних з прогнозуванням хвороб за симтомами.

Галузь застосування розробки – ця система буде корисна у медичній сфері.

ABSTRACT

Explanatory note to the master's qualification work: 72 pages, 9 tables, 17 figures, 2 appendixes, 16 sources of information.

DATABASE, AI, BAYES, ALGORITHM, MEDICINE, FORECASTING, ILLNESS

The object of research is artificial intelligence technologies, namely the Bayesian algorithm for disease prediction.

The subject of research is the use of Bayesian algorithm technology for disease prediction.

The purpose of research is to develop artificial intelligence based on the Bayesian algorithm.

Research methods - creation, testing and implementation of new technologies for disease prediction.

As a result of the research, the problem of creating a Bayesian algorithm using the Python language was solved. The obtained results are used as new possibilities for disease prediction.

The programming language is Python. The proposed development is useful for solving various problems related to the prediction of symptoms by symptoms.

Scope of development - this system will be useful in the medical field.

ЗМІСТ

ВСТУП

Питання громадського здоров'я займають чільне місце в обізнаності населення, політичних дебатах і літературі. Інтелектуальний аналіз даних, робота з виявлення закономірностей у даних, має потенціал впливають на громадське здоров’я безліччю способів, від персоналізованої генетичної медицини до досліджень здоров'я навколишнього середовища та епідеміологія, а також багато застосувань між ними. Класифікація нових даних, яка заснована на моделях, які спостерігалися раніше, є перспективними для застосування конкретних досягнень громадського здоров'я в цілому. Алгоритми класифікації, які використовують теорему Байєса і статистику поширеності захворювання мають мету досягти цього з легкістю при доступних даних.

Для цього дослідження ми застосували наївний класифікатор Байєса до надійного набору даних охорони здоров’я з жадібним відбором ознак з метою ефективного визначення того, які атрибути найкраще передбачають вибраний цільовий атрибут без вичерпного пошуку у вхідному просторі. Наприклад, тривалість перебування в лікарні залежить від типу страхування, регіону, типу лікарні тощо. Це дослідження може сприяти застосуванню підходів аналізу даних до даних громадського здоров’я, зокрема, для передбачення атрибутів, які представляють міру результату лікування, для пацієнтів, які отримують медичні послуги в лікарнях США, на основі легкодоступних даних пацієнта.

1. АНАЛІТИЧНИЙ ОГЛЯД. ПОСТАНОВКА ЗАДАЧІ

1.1 Основні поняття

На відміну від регресії, вихідна змінна класифікації є категорією, а не значенням, таким як "Зелений або Синій", "Фрукти чи тварини" тощо. Оскільки алгоритм класифікації є контрольованою методикою навчання, отже, він приймає позначені вхідні дані, які означає, що він містить вхід з відповідним виходом.[1]

В алгоритмі класифікації дискретна вихідна функція (y) відображається на вхідну змінну (x).

Існує два типи класифікаторів:

1. Двійковий класифікатор: Якщо проблема класифікації має лише два можливі результати, то вона називається двійковою класифікатором.
2. Класифікатор для багатьох класів: Якщо проблема класифікації має більше двох результатів, вона називається Класифікатором для багатьох класів.

Отже тепер потрібно вирішити який з алгоритмів ми будемо використовувати для задачі класифікації. У ході пошуків було вирішено розглянути такі алгоритми машинного навчання:

— Логістична регресія

— SVM

— Random Forest

— Наївний байєсівський класифікатор

— Дерево прийняття рішень

Для вибору алгоритму для нашої задачі потрібно розглянути кожен з цих алгоритмів, та оцінити наскільки вони підходять для вирішення нашої задачі, та складність їх реалізації.

1.2 Логістична регрессія

Логістична регресія - один з найпопулярніших алгоритмів машинного навчання, який підпадає під техніку наглядового навчання. Він використовується для прогнозування категоріальної залежної змінної за допомогою заданого набору незалежних змінних[2].

Логістична регресія передбачає вихід категоріальної залежної змінної. Тому результат повинен мати категоричне або дискретне значення. Це може бути або так, або ні, 0 або 1, істина або хибність тощо, але замість того, щоб дати точне значення як 0 та 1, воно дає імовірнісні значення, які лежать від 0 до 1 .

Логістична регресія багато в чому схожа з лінійною регресією, за винятком того, як вони використовуються. Лінійна регресія використовується для вирішення задач регресії, тоді як логістична регресія використовується для вирішення задач класифікації .

У логістичній регресії, замість лінії регресії, ми підходимо до логічної функції у формі "S", яка передбачає два максимальні значення (0 або 1).

Крива з логістичної функції вказує на ймовірність того, що клітини ракові чи ні, миша страждає ожирінням чи ні, залежно від ваги тощо.

Логістична регресія є важливим алгоритмом машинного навчання, оскільки вона має можливість забезпечувати ймовірності та класифікувати нові дані, використовуючи безперервні та дискретні набори даних.

Логістична регресія може бути використана для класифікації спостережень з використанням різних типів даних і може легко визначити найбільш ефективні змінні, які використовуються для класифікації.

Припущення щодо логістичної регресії:

— Залежна змінна повинна мати категоричний характер.

— Незалежна змінна не повинна мати мультиколінеарності.

Діаграму логістичної регресії зображено на рисунку 1.1

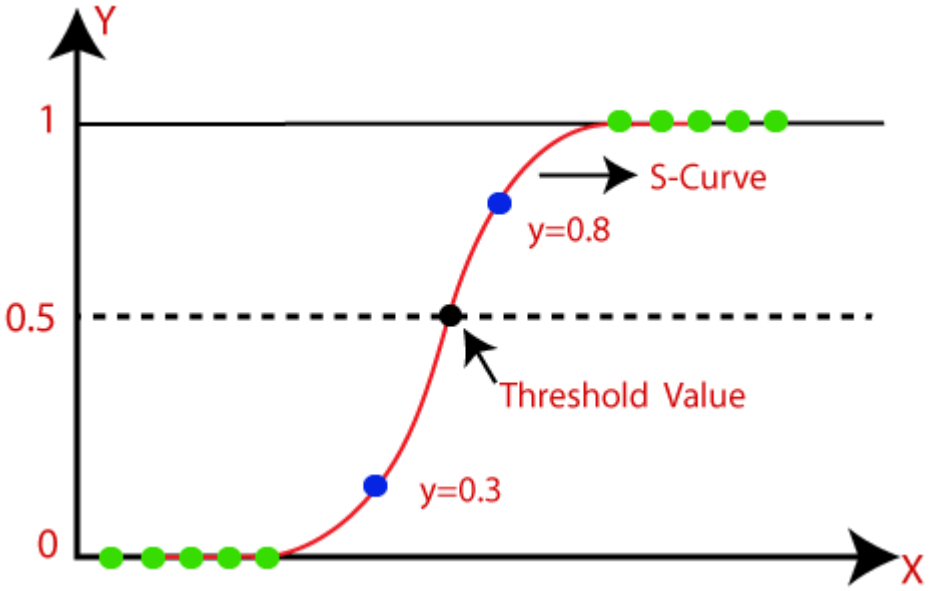

Рисунок 1.1 - Діаграма логістичної регресії

Рівняння логістичної регресії можна отримати з рівняння лінійної регресії.

Наведене нижче рівняння є остаточним рівнянням для логістичної регресії (Формула 1.1).

$$log\left[\frac{y}{1-y}\right] = b_0 + b_1x_1 + b_2x_2 + b_3x_3 + \cdots + b_nx_n \quad (1.1)$$

Розглянемо переваги та недоліки логістичної регресії у таблиці 1.1.

Таблиця 1.1 - Переваги та недоліки логістичної регресії

Переваги	Недоліки
Отримання кількісної оцінки ймовірності у явному вигляді	Неточність при описі змінних
Висока стійкість	Низька чутливість

1.3 SVM

Підтримка векторної машини або SVM - один з найпопулярніших алгоритмів нагляду, який використовується для класифікації, а також для проблем регресії. Однак, насамперед, він використовується для вирішення проблем класифікації в машинному навчанні[3].

Метою алгоритму SVM є створення найкращої лінії або межі рішення, яка може розділити n-вимірний простір на класи, щоб ми могли легко поставити нову точку даних у правильну категорію в майбутньому. Ця межа найкращого рішення називається гіперплощиною.

SVM вибирає крайні точки/вектори, які допомагають у створенні гіперплощини. Ці крайні випадки називаються векторами підтримки, і тому алгоритм називається машиною підтримки вектора. Розглянемо наведену нижче діаграму, на якій є дві різні категорії, які класифікуються за кордоном прийняття рішення або гіперплощиною (Рисунок 1.2).

Алгоритм SVM може бути використаний для виявлення облич, класифікації зображень, категоризації тексту тощо.

SVM може бути двох типів:

— Лінійний SVM: Лінійний SVM використовується для лінійно роздільних даних, що означає, що якщо набір даних можна класифікувати у два класи за допомогою однієї прямої лінії, то такі дані називаються лінійно розділими даними, а класифікатор використовується як лінійний SVM - класифікатор.

— Нелінійний SVM: Нелінійний SVM використовується для нелінійно розділених даних, що означає, що якщо набір даних не можна класифікувати за допомогою прямої лінії, то такі дані називаються нелінійними даними, а класифікатор, що використовується, називається не лінійний класифікатор SVM.

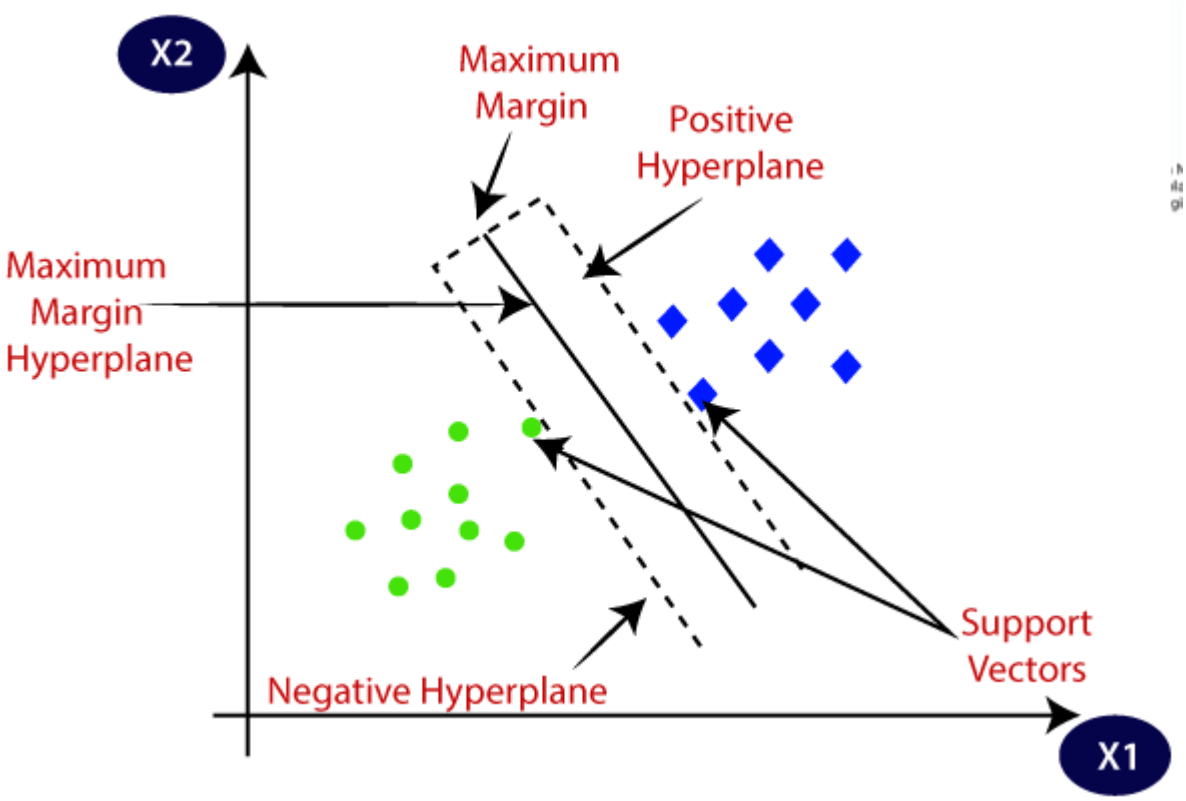

Рисунок 1.2 - Діаграма SVM

Гіперплощина: Може бути кілька ліній/кордонів рішень для відокремлення класів у n-вимірному просторі, але нам потрібно з'ясувати найкращу межу прийняття рішень, яка допомагає класифікувати точки даних. Ця найкраща межа відома як гіперплощина SVM.

Розміри гіперплощини залежать від об’єктів, наявних у наборі даних, а це означає, що якщо є 2 об’єкти (як показано на зображенні), то гіперплощина буде прямою. А якщо є 3 ознаки, то гіперплощина буде двовимірною площиною.

Ми завжди створюємо гіперплощину з максимальним запасом, що означає максимальну відстань між точками даних.

Точки даних або вектори, які є найближчими до гіперплощини і які впливають на положення гіперплощини, називаються опорними векторами.

Оскільки ці вектори підтримують гіперплощину, тому їх називають вектором підтримки.

Розглянемо недоліки та переваги у таблиці 1.2

Таблиця 1.2 - Переваги та недоліки SVM

Переваги	Недоліки
Більш впевнена класифікація	Нестійка
Число нейронів визначається автоматично	Потрібно підбирати константу

1.4 Наївний Байес

Наївний алгоритм Байєса - це керований алгоритм навчання, який базується на теоремі Байєса і використовується для вирішення задач класифікації.

В основному він використовується у класифікації тексту, що включає високомірний навчальний набір даних.

Це один з найпростіших та найефективніших алгоритмів класифікації, який допомагає у створенні моделей швидкого машинного навчання, які дозволяють швидко прогнозувати[4].

Це імовірнісний класифікатор, що означає, що він передбачає на основі ймовірності об’єкта .

Деякі популярні приклади алгоритму наївного Байєса - фільтрація спаму, сентиментальний аналіз та класифікація статей

Алгоритм складається з двох слів Naive та Bayes, які можна описати як:

Naive : він називається наївним, оскільки передбачає, що поява певної ознаки не залежить від появи інших ознак. Наприклад, якщо плід

ідентифікується за кольором, формою та смаком, то червоний, кулястий та солодкий фрукт розпізнається як яблуко. Отже, кожна особливість окремо допомагає визначити, що це яблуко, не залежачи одне від одного.

Bayes : Його називають Байєсом, тому що він залежить від принципу теореми Байєса .

Теорема Байєса також відома як Правило Байєса або закон Байєса , який використовується для визначення ймовірності гіпотези за наявності попередніх знань. Це залежить від умовної ймовірності.

Формула теореми Байєса зображено у формулі 1.2.

$$P(B) = \frac{P(B|A)P(A)}{P(B)} \tag{1.2}$$

Застосування класифікатора наївного Байєса:

— Він використовується для кредитного скорингу .
— Він використовується в класифікації медичних даних .
— Його можна використовувати для прогнозування в режимі реального часу, оскільки класифікатор охоче навчається.
— Він використовується в класифікації тексту, такі як фільтрація спаму та аналіз настроїв .

Розглянемо переваги та недоліки наївного байєса у таблиці 1.3.

Таблиця 1.3 - Переваги та недоліки наївного байеса

Переваги	Недоліки
Наївний Байєс - один з швидких і простих алгоритмів ML для передбачення класу наборів даних.	Наївний Байєс припускає, що всі функції незалежні або не пов'язані між собою, тому він не може визначити зв'язок між ознаками.
Його можна використовувати як для двійкових, так і для багатокласових класифікацій.	
У порівнянні з іншими алгоритмами, він добре працює у передбаченні багатокласності.	

1.5 Дерево рішень

Дерево рішень - це наглядова техніка навчання, яка може бути використана як для задач класифікації, так і для регресії, але переважно вона є кращою для вирішення проблем класифікації. Це деревно-структурований класифікатор, де внутрішні вузли представляють особливості набору даних, гілки представляють правила прийняття рішень, а кожен листовий вузол представляє результат[5].

У дереві прийняття рішень є два вузли, це вузол прийняття рішень і листовий вузол. Вузли прийняття рішень використовуються для прийняття будь -якого рішення і мають декілька гілок, тоді як вузли «Листя» є результатом цих рішень і не містять жодних подальших гілок.

Рішення або перевірка виконуються на основі особливостей даного набору даних.

У машинному навчанні існують різні алгоритми, тому вибір найкращого алгоритму для даного набору даних та проблеми - це головне, що слід пам’ятати під час створення моделі машинного навчання. Нижче наведено дві причини використання дерева рішень:

— Дерева рішень зазвичай імітують здатність людини мислити під час прийняття рішення, тому це легко зрозуміти.

— Логіку за деревом рішень можна легко зрозуміти, оскільки він показує деревоподібну структуру.

У дереві рішень для прогнозування класу даного набору даних алгоритм починається з кореневого вузла дерева. Цей алгоритм порівнює значення кореневого атрибута з атрибутом запису (реального набору даних) і на основі порівняння слідує за гілкою і переходить до наступного вузла.

Для наступного вузла алгоритм знову порівнює значення атрибута з іншими підвузлами і рухається далі. Він продовжує процес, поки не досягне листового вузла дерева.

Розглянемо переваги та недоліки дерева рішень у таблиці 1.4.

Таблиця 1.4 - Переваги та недоліки дерева рішень

Переваги	Недоліки
Легко для розуміння, оскільки воно слідує за тим самим процесом, яким керується людина, приймаючи будь-яке рішення в реальному житті.	Дерево рішень містить багато шарів, що робить його складним.
Це може бути дуже корисним для вирішення проблем, пов'язаних з прийняттям рішень.	Можливо, у нього є проблема з переобладнанням, яку можна вирішити за допомогою алгоритму Випадкового лісу.
Це допомагає продумати всі можливі наслідки проблеми.	Для більшої кількості міток класів обчислювальна складність дерева рішень може зрости.
У порівнянні з іншими алгоритмами вимоги до очищення даних є меншими.	

1.6 Випадковий ліс

Випадковий ліс - популярний алгоритм машинного навчання, який належить до техніки навчання під наглядом. Він може бути використаний як для проблем класифікації, так і для регресії. Він базується на концепції ансамблевого навчання, яке являє собою процес об’єднання кількох класифікаторів для вирішення складної проблеми та покращення продуктивності моделі[6].

Випадковий ліс працює на двох етапах, по-перше, це створення випадкового лісу шляхом об'єднання N дерева рішень, а по-друге, це передбачення для кожного дерева, створеного на першому етапі.

В основному існує чотири сектори, де переважно використовуються випадкові ліси:

— Банківська справа: банківський сектор здебільшого використовує цей алгоритм для ідентифікації кредитного ризику.
— Медицина: За допомогою цього алгоритму можна визначити тенденції захворювання та ризики захворювання.
— Використання землі: за цим алгоритмом ми можемо визначити зони подібного землекористування.
— Маркетинг: За допомогою цього алгоритму можна визначити маркетингові тенденції.

Розглянемо переваги та недоліки випадкового лісу у таблиці 1.5:

Таблиця 1.5 - Переваги та недоліки випадкового лісу

Переваги	Недоліки
Випадковий ліс здатний виконувати як класифікаційні, так і регресійні завдання.	Хоча випадковий ліс можна використовувати як для задач класифікації, так і для регресії, він менш підходить для завдань регресії.
Він здатний обробляти великі набори даних з високими розмірами.	
Підвищує точність моделі та запобігає проблемі надмірної комплектації.	

1.7 Прийняття рішення щодо використання

Після певного аналізу було вирішено використовувати алгоритм Наївного Байєса. Це обумовлено простотою його реалізації, та тим що він нам підходить за принципом дії. Головна його відмінність від інших розглянутих алгоритмів це те що він обробляє вхідні дані як незалежні один від одного. Це нам підходить через те що симптоми можно розглядати саме таким способом. Саме через це, цей метод часто використовується у медицині для прогнозування захворювання.

2. АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ВИРІШЕННЯ ПРОБЛЕМИ

2.1 Опис методу байєса

Баєсова мережа є однією з найбільш класичних імовірнісних графічних моделей. Він широко використовується в багатьох областях, таких як штучний інтелект, розпізнавання образів та обробка зображень. У цьому дослідженні було досліджено шість типових алгоритмів навчання параметрів. Для повноти набору даних існують в основному дві категорії методів для оцінки параметрів у байєсівських мережах: один підходить для роботи з повними даними, а інший — для роботи з неповними даними. Було вирішено зосередилися на двох алгоритмах першої категорії: оцінка максимальної правдоподібності та байєсівський метод. Для другої категорії обговорювалися алгоритм очікування – максимізації, надійна баєсова оцінка, метод Монте – Карло та метод гаусової апроксимації. В експерименті, всі ці алгоритми були застосовані на класичному прикладі для реалізації виведення параметрів. Результати моделювання виявляють притаманні відмінності цих шести методів і вплив виведених параметрів мережі на подальший обчислення ймовірності. Це дослідження дає уявлення про стратегії виведення параметрів байєсівської мережі та їх застосування в різних ситуаціях.

Байєсівська мережа є ймовірнісної моделлю, визначеною над ациклічним орієнтованим графом. Він розраховується за допомогою одного умовного розподілу ймовірності для кожної змінної в моделі, розподіл якої наведено на графіку залежно від її батьків. Змінні, які відокремлені в графі, все ще незалежні, але просте поділ графів, що використовується для неорієнтованого графа, замінено на більш складне d-розділення, яке враховує вплив конкуруючих пояснень для спостережуваних значень.

Байєсівську мережу можна перетворити у ненаправлену графічну модель, з'єднавши всі вузли, які беруть участь у кожному факторі. Це вимагає,

щоб батьки кожного вузла були об’єднані. Процес приєднання батьків відомий як моралізація, а неорієнтований граф, що відповідає даній байєсівській мережі, називається моральним графом. Обчислення морального графа є першим кроком багатьох обчислювальних алгоритмів для байєсівських мереж.

Ланцюгові графи використовують суміш спрямованих і неорієнтованих графічних ребер для опису відношення незалежності між змінними. Ланцюговий граф розбивається на серію блоків; всередині блоку всі ребра ненаправлені, а між блоками ребра спрямовані. Блоки часто відповідають етапам в експерименті; наприклад, один блок може включати вимірювання, зроблені до початку навчального втручання, а інший — вимірювання, зроблені після. Третій блок може представляти фонові змінні про учнів і школи. Спрямовані ребра протікають у тимчасовому напрямку: від заходів до втручання до заходів після втручання та від фонових змінних до будь-яких інших.

Діаграми шляху мають очевидну схожість з графічними моделями (особливо спрямованими байєсівськими мережами), і, справді, робота над діаграмами шляху була натхненням для подальшої роботи над графічними моделями. Однак є деякі тонкі та важливі відмінності. Деякі з них очевидні: діаграми шляху включають явні вузли для помилок, тоді як вони зазвичай неявні в графічних моделях і байєсівських мережах. Діаграми шляхів допускають деякі види виразів (подвійні стрілки та взаємні зв'язки), які не допускаються в графічних моделях. Інші відмінності більш тонкі. Зокрема, дисципліна моделювання структурних рівнянь зосереджується на моделюванні коваріаційної матриці, тоді як графічні моделі зосереджені на моделюванні оберненої коваріаційної матриці. Підсумок полягає в тому, що припущення про умовну незалежність, неявні в моделі структурного рівняння , не завжди чітко виражені; отже, властивість Маркова може не виконуватися для діаграми шляху. Більшість діаграм шляху мають еквівалентне представлення графічної моделі, але є деякі винятки (як правило, моделі, які важко оцінити).

2.2 Використання наївного байєса у медицині

Не дивно, що багато аналізу даних проводиться у сфері охорони здоров’я, особливо у клінічній медицині, та потенційний вплив такої роботи є широким і переконливим[7]. Збір медичних даних створює унікальні проблеми: неоднорідність медичних даних, етичні, правові та соціальні обмеження щодо використання цих даних, статистичні підходи, які розглядають гетерогенність і ці обмеження, а також особливий статус медицини як шанована і ретельно досліджувана сфера, відповідальна за рішення щодо життя і смерті, які можуть вплинути на всіх нас[8].

Було показано, що наївна класифікація Байєса краща ніж декілька інших класифікаційних методів, які застосовуються спеціально до медичних даних [9]. Автори оцінюють наївний Байєс і п'ять інших методів на п'ятнадцяти наборах медичних даних UCI.

Вони віддають перевагу наївному Байєсу як за його прогнозну продуктивність, так і за його прозорість та інтерпретацію. Вони визначають наївне припущення про незалежність Байєса як найбільш актуальну область для майбутньої роботи та припускають, що гібридні методи можуть допомогти.

У попередньому дослідженні порівнювали наївні методи Байєса та сім інших методів, а також чотири гібридні методи прогнозування смертності від пневмонії. Вісім методів мали абсолютний рівень помилок в межах 1%, чотири гібридні методи були визнані перспективними, але не статистично надійними.

У дослідженні 2007 року пропонують комбінувати декілька методів[10]. В їхньому випадку вихід з трьома різними класифікаторами були вхідними для нейронної мережі, і всі вони використовувалися для класифікації даних мамографії. Хоча це, можливо, дало хороші результати на їхніх тестових даних, така модель була неінтуїтивною для прийняття рішень у реальному світі.

Важливість попередньої обробки часто посилається в дослідженнях, заснованих на медичних даних. Попеску та Халіліа використовують ієрархію, неявну в кодах МКБ-9 як міра подібності серед пацієнтів, в тому числі призначених подібних пацієнтів під час класифікації покращила прогнозну продуктивність випадкового лісу та машини опорних векторів[11]. Майбутня робота авторів передбачає розширення їхньої роботи на більшу кількість захворювань, як і раніше, використовуючи дані національних стаціонарних вибірок від проекту Агентства з досліджень у сфері охорони здоров'я та якості.

У нашій роботі хотілось би, щоб їхній підхід поєднувався з іншими методами класифікації.

Кілька досліджень зосереджені на виборі функцій як ключовому етапі попередньої обробки. Вибір функції був використовується для визначення найбільш помітних атрибутів у наборі даних і, таким чином, підвищення точності та ефективність класифікації за кількома методами: наївним Байєсом, IB1 і C4.5 [12]. У цьому дослідженні об'єкти були обрані на основі їх вирівнювання з цільовим атрибутом. Автори зосередилися на прогнозуванні стану контролю цукрового діабету у пацієнтів при цукровому діабеті 2 типу; однак вибір ознак на основі ймовірної сили передбачення може розглядатися для даних охорони здоров'я ширше.

Одна серія статей описує комбінацію кількох кроків попередньої обробки для підвищення рівня точності класифікації медичних даних за допомогою Байєса: дискретизація на основі ентропії та вибір ознак, що включає фільтрацію ознак з низьким рівнем статистика хі-квадрат і жадібний вибір серед інших функцій[13]. Була заснована робота над наборами медичних даних зі сховища машинного навчання UCI. Жадібний вибір функцій використання інших мір імовірної сили прогнозу варто було б дослідити.

Ми вважаємо цікавими ще два аспекти цього дослідження, однак: використання 3-значних кодів МКБ-9 для консолідації вхідного простору та використання схожості випадковості трьох окремих показників супутньої патології у виборі ознак.

2.3 Методологія

З огляду літератури випливає кілька питань: важливість медичних даних як основи для будь-яких висновків, що стосуються сфери охорони здоров’я, обіцянка наївної байєсової класифікації для прогнозування у сфері охорони здоров’я та впливу вибору ознак на класифікаційну точність. У роботі був застосований наївний класифікатор Байєса до надійного набору даних громадського здоров’я з жадібним вибором об’єктів таким чином, що може бути n атрибутів, які найкраще передбачають вибраний цільовий атрибут. Гіпотеза полягала в тому, що цей підхід, детально описаний нижче, допоможе у прогнозуванні у сфері охорони здоров'я.

Методологія, що була використана у нашій роботі поділяється на компоненти, визначені Домінгосом [14].

Було використано дані опубліковані в обстеженні виписки з лікарні (Національний центр статистики охорони здоров’я [15]. Це щорічне опитування включає демографічну інформацію, допуск та виписку інформації, діагнози та процедури. Дослідження 2010 року включало 151 551 пацієнта у 203 лікарнях короткочасного перебування. У карті кожного пацієнта включаються ваги, які дозволяють зробити екстраполяцію статистичних даних на національні і регіональні рівні. Для простоти ми виключили дітей віком до року (залишилося 135 418 пацієнтів).

Було обрано наївний класифікатор Байєса для передбачення будь-якого з атрибутів набору даних, усі з яких є дискретними або їх можна легко дискретизувати.

Наївне припущення Байєса про незалежність атрибутів часто піддається критиці. Хоча незалежність майже напевно не є випадком між атрибутами в даних охорони здоров'я, численні дослідження показали, що цей підхід, тим не менш, є точним та ефективним. Теоретики Ріш, Hellerstein і Thathachar вважають наївну класифікацію Байєса найбільш точною, коли атрибути або незалежні, або найбільш тісно корелюють[16].

Точність класифікації це простий, ефективний показник оцінки, який узгоджується з наївним підходом Байєса. Більшість атрибутів набору даних можна вибрати як цільовий атрибут. У роботі треба було зосередилися на тривалості перебування в лікарні, як потенційному показнику якості та вартості медичної допомоги, та статус звільнення через його очевидні гуманітарні наслідки. Також було врахувано число діагностичних кодів, як ще один потенційний показник вартості та тяжкості захворювання. Набір даних було розділено на навчальні та тестові дані, використовуючи 10-кратну перехресну перевірку. Перехресна перевірка є методом за допомогою якого дані випадковим чином поділяються на підмножини, у нашому випадку на десять. Одна підмножина ізольована для використання даних тестування, а решта буде навчальними даними. У послідовних ітераціях модель оброблює кожну підмножину даних тестування, а результати усереднюються. Таким чином, фактичні дані доступні для обчислення точності класифікації, та зміщення зведено до мінімуму.

Для оптимізації було реалізувано жадібний вибір функцій, так що n атрибутів найкраще передбачають вибраний цільовий атрибут. Можна ефективно ідентифікувати його без вичерпного пошуку у вхідному просторі.

Була розрахована точність класифікації цільового атрибута з кожним іншим атрибутом і рейтингом. Можливо, що атрибут ефективний у поєднанні з іншим, але не окремо, однак щоб не виключати таку можливість виходу на поверхню, ми використали випадковий вибір атрибуту з наведеним вище рейтингом як вагові показники, щоб атрибути з високим рейтингом мали більшу ймовірність відбору.

2.4 Вибір інструментів реалізації

Інтелектуальний аналіз даних — це передусім математична вправа, застосування алгоритмів до даних і результуюче обчислення, на відміну від

того, в якому поведінка об'єктів у реальному світ моделюється. З цієї причини ми вирішили використовувати функціональну мову програмування, а саме Clojure, Lisp, який працює на JVM і надає доступ до будь-яких бібліотек Java.

Щоб покращити продуктивність, ми отримали доступ до наївної моделі Байєса у Weka 3.7.7 за допомогою Weka Java API від Clojure, а також методи 10-кратної перехресної перевірки. Це покращить швидкість обчислення.

2.5 Техніко-економічне обґрунтування

На самому початку роботи треба було впровадили наївний класифікатор Байєса в Clojure і застосувати його до NHDS 2010 року. Було вирішено зосередилися на вікі пацієнтів, розділених на повні десятиліття, пов’язані з діагнозом

Група, яка має 730 різних значень і тривалість перебування в лікарні, розділена на шість категорій.

Поширеність кожної категорії тривалості перебування в повному наборі даних окремо та для кожної комбінації віку та діагнозу, використовувався для прогнозування тривалості перебування кожного пацієнта. Для зразка 50 000 пацієнтів, отриманих з регулярних інтервалів повного (невпорядкованого) набору даних, 53% були передбачені правильно. Ці результати були продубльовані за допомогою Weka 3.6.6 Explorer.

2.6 Аналіз результатів

Ця комбінація наївного байесового класифікатора з жадібним вибором ознак виявила найкраще прогнозовані атрибути для заданого цільового атрибута. Однак досягнуте підвищення точності класифікації залежало від цільових атрибутів. В деяких випадках, збільшення було помірним; в інших

це натякало на (розчаруючу) реальність. Зрештою, прогнозні відносини можуть не ховатися в даних.

Оскільки жадібний алгоритм вибору функцій має елемент випадковості, атрибути було обрано як найбільш прогнозовані після 100 ітерацій, або деякої кількості ітерацій, менших за вхідний простір. Однак точність мало змінювалася від пробігу до пробігу.

Для дослідження цього підходу було обрано в якості цільового атрибута тривалість перебування в стаціонарі, виписку, статус і кількість діагностичних кодів, у свою чергу.

Для тривалості перебування в стаціонарі було розділено дані на наступні шість категорій: один день, два дня, один тиждень, один місяць, два місяці і довгострокові (діапазон становив від 0 до 497 днів).

Таким чином, сліпе припущення може становити 1/6 або 17% точності класифікації. Оцінка на основі даних та відправна точка для наївного підходу Байєса — це передбачити те, що буде найбільш поширене значення (тут два дні); у цьому випадку це досягло б 46% класифікаційної точності. Використовуючи наївну модель Байєса, найкращий одиничний атрибут досяг 61%; найкращі два, три, і чотири атрибути, вибрані за 100 ітерацій, досягли точності класифікації 67%-70%. Серед усіх ітерацій, найбільш вдалими атрибутами, окремо або в комбінації, були ті, що стосуються кодів діагностики або процедури. Більш детальна інформація наведена в таблиці 2.1.

Таблиця 2.1 - Тривалість перебування в стаціонарі

Класифікація	Точність
Первинний діагноз	61%
Первинний діагноз, первинна процедура	67%
Первинний діагноз, первинна процедура, постановка діагнозу	69%
Первинний діагноз, первинна процедура, постановка діагнозу, DRG	70%

Використовуючи наївну модель Байєса, найкращі одиночні, два, три та чотири атрибути відібрано зі 100 ітерації змогли лише покращити це до 80-83% точності класифікації. Той самий набір атрибутів, пов'язаних з кодом діагностики та кодом процедури, були кращими провісниками. Детальніше у таблиці 2.2.

Таблиця 2.2 - Статус розряду.

Класифікація	Точність
Первинний діагноз	80%
Первинний діагноз плюс первинна процедура або діагноз при госпіталізації	82%
Первинний діагноз, первинна процедура та будь-який діагноз або джерело надходження	82%
Первинний діагноз, первинна процедура, постановка діагнозу, джерело прийому	83%

NHDS 2010 року (NCHS, 2010b) фіксує до п’ятнадцяти діагностичних кодів і до восьми процедур для кожного пацієнта. Два атрибути, які ми включили, були розрахунками кількості присутніх кодів. Можливі значення кількості діагностичних кодів становили 1-15.

Розподіл був досить рівним, з найбільш поширеним значенням у 17% пацієнтів.

Використовуючи наївну модель Байєса, найкращий одиничний атрибут підвищує точність класифікації до 40%. Найкращі два, три та чотири атрибути, вибрані за 100 ітерацій, покращили цей показник до 49%-56%. Більш детальна інформація наведена в таблиці 2.3.

Таблиця 2.3 - Кількість діагностичних кодів.

Класифікація	Точність
Первинний діагноз	40%
Первинний діагноз, постановка діагнозу	49%
Первинний діагноз, первинна процедура, постановка діагнозу	54%
Первинний діагноз, первинна процедура, постановка діагнозу, DRG	56%

Можливість того, що прогнозні відносини не підтверджуються даними або, з іншого боку, що прогнозні зв’язки, які випливають із даних, якимось чином є унікальними для цих даних і не мають застосовність за його межами, необхідно враховувати.

Фактором може бути проблема розмірності, тобто можливі комбінації всіх значень з усіх атрибутів може створити вхідний простір, достатньо великий, щоб доступних даних було недостатньо для впевненого оцінювання будь-яких передбачуваних відносин. Навіть набір даних із 150 000 пацієнтів охоплює невелика підмножина вхідного простору, створеного атрибутами цього набору даних.

Для більше точної роботи класифікатора потрібно, щоб не було даних рівномірно розподілених у вхідному просторі, але щоб дані були достатньо згруповані, щоб уможливити обчислення корисних класифікаторів.

3. РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ

3.1 Опис кроків алгоритму байєса.

Наївний алгоритм Байєса — це алгоритм навчання з наглядом, який базується на теоремі Байєса і використовується для розв’язування задач класифікації.

Ми можемо використовувати ймовірність, щоб робити прогнози в машинному навчанні. Наївний алгоритм Байєса не тільки легко зрозуміти, але він досягає дивовижно хороших результатів у широкому колі проблем.

Теорема Байеса надає спосіб обчислити ймовірність того, що частина даних належить до даного класу, враховуючи наші попередні знання. Теорема Байеса зформульована у формулі 3.1.

$$P(\text{дані}) = \frac{P(\text{дані}|\text{клас})P(\text{клас})}{P(\text{дані})} \tag{3.1}$$

Для ілюстрації роботи реалізуємо наївний алгоритм Байєса за допомогою Python. Тому для цього потрібно використовувати набір даних "user_data".

Для реалізації алгоритму, треба реалізувати такі кроки:

— Крок 1 попередньої обробки даних

— Крок 2: Розділіть за класами.

— Крок 3. Підсумуйте набір даних.

— Крок 4. Підсумуйте дані за класами.

— Крок 5: функція щільності гаусової ймовірності.

— Крок 6: Імовірності класів.

3.2 Крок попередньої обробки даних

Попередня обробка даних — це процес підготовки необроблених даних і придатності для моделі машинного навчання. Це перший і вирішальний крок під час створення моделі машинного навчання.

Дані реального світу зазвичай містять шуми, відсутні значення і, можливо, у непридатному для використання форматі, який не можна використовувати безпосередньо для моделей машинного навчання. Попередня обробка даних є обов'язковим завданням для очищення даних і придатності для моделі машинного навчання, що також підвищує точність та ефективність моделі машинного навчання.

Реалізація попередньої обробки даних включає в себе такі кроки:

— Отримання набору даних

— Імпорт бібліотек

— Імпорт наборів даних

— Пошук відсутніх даних

— Кодування категоріальних даних

— Розбиття набору даних на навчальний та тестовий набір

— Масштабування функцій

Щоб створити модель машинного навчання, перше, що для цього необхідно, це набір даних, оскільки модель машинного навчання повністю працює на даних. Зібрані дані для певної проблеми у належному форматі відомі як набір даних.

Щоб використовувати набір даних у нашому коді, зазвичай треба помістити їх у файл CSV . Однак іноді можна використовувати файл HTML або xlsx.

Щоб виконати попередню обробку даних за допомогою Python, потрібно імпортувати деякі попередньо визначені бібліотеки Python. Ці бібліотеки використовуються для виконання деяких конкретних завдань.

Тепер нам потрібно імпортувати набори даних, які було зібрано для нашого проекту машинного навчання.

Наступним кроком попередньої обробки даних є обробка відсутніх даних у наборах даних. Якщо набір даних містить деякі відсутні дані, це може створити величезну проблему для нашої моделі машинного навчання. Тому необхідно обробляти відсутні значення, присутні в наборі даних.

В основному існує два способи обробки відсутніх даних, а саме:

— Видаленням конкретного рядка: перший спосіб зазвичай використовується для роботи з нульовими значеннями. Таким чином, ми просто видаляємо певний рядок або стовпець, який складається з нульових значень. Але цей спосіб не настільки ефективний, і видалення даних може призвести до втрати інформації, яка не дасть точного результату.

— Обчислюючи середнє: таким чином ми обчислимо середнє значення для того стовпця або рядка, який містить пропущене значення, і помістимо його на місце пропущеного значення. Ця стратегія корисна для функцій, які мають числові дані, такі як вік, зарплата, рік тощо.

Оскільки модель машинного навчання повністю працює на основі математики та чисел, та якщо набір даних матиме категоріальну змінну, це може створити проблеми під час побудови моделі. Тому необхідно закодувати ці категоріальні змінні в числа.

Категоричні дані – це дані, які мають деякі категорії.

Під час попередньої обробки даних машинного навчання треба поділити набір даних на навчальний набір і тестовий набір. Це один із важливих кроків попередньої обробки даних, оскільки, роблячи це, можна підвищити продуктивність моделі машинного навчання.

Припустимо, що модель машинного навчання навчається за набором даних і тестується абсолютно іншим набором даних. Тоді це створить труднощі для моделі.

Якщо ми дуже добре натренуємо модель, і її точність навчання також дуже висока, та ми надамо їй новий набір даних, це знизить продуктивність. Тому треба завжди створювати модель машинного навчання, яка добре працює як з навчальним набором, так і з набором тестових даних.

Масштабування функцій — це останній крок попередньої обробки даних у машинному навчанні. Це метод стандартизації незалежних змінних набору даних у певному діапазоні. Під час масштабування ознак треба розмістити наші змінні в одному діапазоні та в одному масштабі, щоб жодна змінна не домінувала над іншою змінною.

Отже, код для попередньої обробки даних можна побачити у лістингу 3.1

```
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd

# Importing the dataset
dataset = pd.read_csv('user_data.csv')
x = dataset.iloc[:, [2, 3]].values
y = dataset.iloc[:, 4].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.25, random_state = 0)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

Лістинг 3.1 - Попередня обробка даних

3.3 Розподілення за класами

Для початку потрібно буде обчислити ймовірність даних за класом, до якого вони належать, так звану базову ставку.

Це означає, що спочатку потрібно буде розділити наші навчальні дані за класами.

Потрібно створити об’єкт словника, де кожен ключ є значенням класу, а потім додати список усіх записів як значення в словник.

У лістингу 3.2 наведена функція з назвою separat_by_class (), яка реалізує цей підхід. Він передбачає, що останній стовпець у кожному рядку є значенням класу.

```
1  # Split the dataset by class values, returns a dictionary
2  def separate_by_class(dataset):
3          separated = dict()
4          for i in range(len(dataset)):
5                  vector = dataset[i]
6                  class_value = vector[-1]
7                  if (class_value not in separated):
8                          separated[class_value] = list()
9                  separated[class_value].append(vector)
10         return separated
```

Лістинг 3.2 - Словник

Треба створити невеликий набір даних для тестування(Лістинг 3.3). Також побудуємо цей набір даних і використовувати окремі кольори для кожного класу. Діаграма розсіювання невеликого надуманого набору даних для тестування наївного алгоритму Байєса проілюстровано на рисунку 3.1

Зібравши все це разом, протестуємо нашу функцію separate_by_class() на надуманому наборі даних.

	X1	X2	Y
1			
2	3.393533211	2.331273381	0
3	3.110073483	1.781539638	0
4	1.343808831	3.368360954	0
5	3.582294042	4.67917911	0
6	2.280362439	2.866990263	0
7	7.423436942	4.696522875	1
8	5.745051997	3.533989803	1
9	9.172168622	2.511101045	1
10	7.792783481	3.424088941	1
11	7.939820817	0.791637231	1

Лістинг 3.3 - Тестовий набор даних

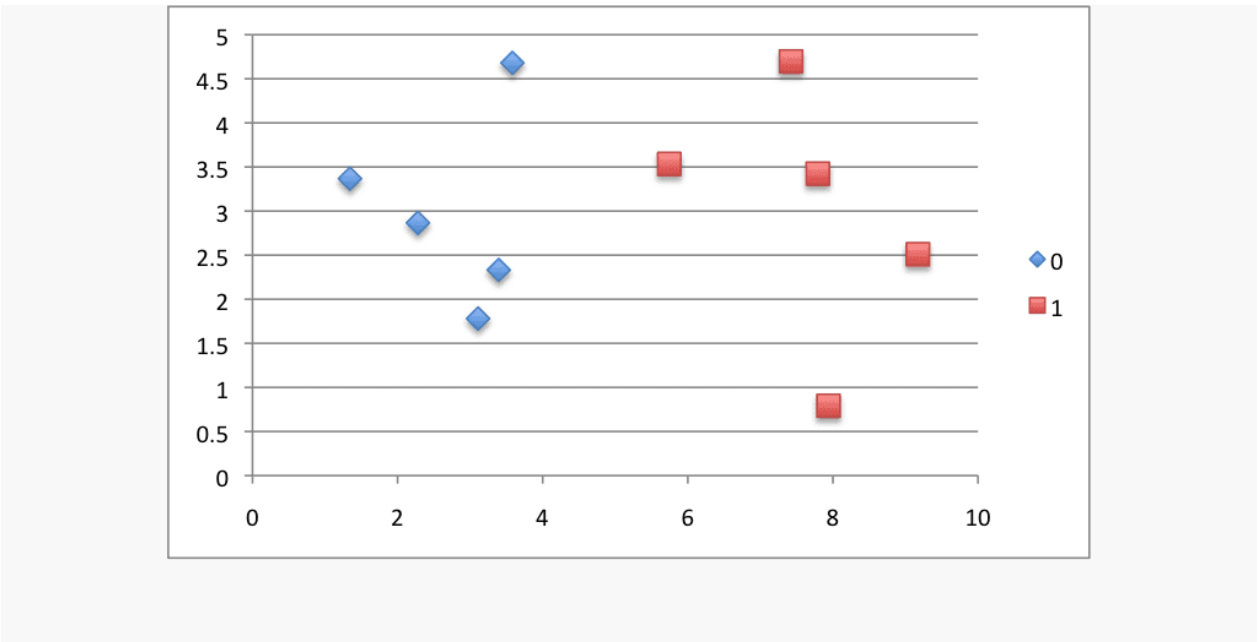

Рисунок 3.1 - Діаграма розсіювання невеликого надуманого набору даних

Запуск прикладу (Лістинг 3.4) сортує спостереження в наборі даних за значенням їх класу, а потім друкує значення класу, а потім усі ідентифіковані записи(Лістинг 3.5).

```
# Example of separating data by class value

# Split the dataset by class values, returns a dictionary
def separate_by_class(dataset):
	separated = dict()
	for i in range(len(dataset)):
		vector = dataset[i]
		class_value = vector[-1]
		if (class_value not in separated):
			separated[class_value] = list()
		separated[class_value].append(vector)
	return separated

# Test separating data by class
dataset = [[3.393533211,2.331273381,0],
	[3.110073483,1.781539638,0],
	[1.343808831,3.368360954,0],
	[3.582294042,4.67917911,0],
	[2.280362439,2.866990263,0],
	[7.423436942,4.696522875,1],
	[5.745051997,3.533989803,1],
	[9.172168622,2.511101045,1],
	[7.792783481,3.424088941,1],
	[7.939820817,0.791637231,1]]
separated = separate_by_class(dataset)
for label in separated:
	print(label)
	for row in separated[label]:
		print(row)
```

Лістинг 3.4 - Код для тестування separate_by_class()

```
1  0
2  [3.393533211, 2.331273381, 0]
3  [3.110073483, 1.781539638, 0]
4  [1.343808831, 3.368360954, 0]
5  [3.582294042, 4.67917911, 0]
6  [2.280362439, 2.866990263, 0]
7  1
8  [7.423436942, 4.696522875, 1]
9  [5.745051997, 3.533989803, 1]
10 [9.172168622, 2.511101045, 1]
11 [7.792783481, 3.424088941, 1]
12 [7.939820817, 0.791637231, 1]
```

Лістинг 3.5 - Результати виконання separate_by_class()

3.4 Підсумовування набору даних

Для цього кроку потрібні два статистичних параметру з заданного набору даних. Визначимо, як ці параметри використовуються при обчисленні ймовірностей за кілька кроків. Параметри, які потрібні для даного набору даних, це середнє значення і стандартне відхилення (середнє відхилення від середнього).

Середнє є середнім значенням і може бути розраховано як:

$$M(x) = \frac{\sum x}{n} \tag{3.2}$$

де x - це список значень або стовпець, який шукається;
n – кількість значень або стовпців;

Нижче наведена невелика функція з назвою mean(), яка обчислює середнє значення списку чисел(Лістинг 3.6).

```
1 # Calculate the mean of a list of numbers
2 def mean(numbers):
3         return sum(numbers)/float(len(numbers))
```

Лістинг 3.6 - Обчислення середнього значення

Вибіркове стандартне відхилення розраховується як середня відмінність від середнього значення. Обчислення надано у формулі 3.3

$$\sigma(x) = \sqrt{\frac{\sum_i^N (x_i - M(x))^2}{N-1}} \tag{3.3}$$

З цього видно, що возводиться різниця між середнім і заданим значенням, обчислюється середня квадратична різниця від середнього, а потім береться квадратний корінь, щоб повернути одиниці до початкового значення.

У лістингу 3.7 наведена невелика функція з назвою standard_deviation(), яка обчислює стандартне відхилення списку чисел. Цей метод обчислює середнє. Можливо, було б ефективніше обчислити середнє значення списку чисел один раз і передати його у функцію standard_deviation() як параметр.

```
1 from math import sqrt
2
3 # Calculate the standard deviation of a list of numbers
4 def stdev(numbers):
5         avg = mean(numbers)
6         variance = sum([(x-avg)**2 for x in numbers]) / float(len(numbers)-1)
7         return sqrt(variance)
```

Лістинг 3.7 - Код обчислення стандартного відхилення

Потрібно, щоб середнє значення та статистика стандартного відхилення були розраховані для кожного вхідного атрибута або кожного стовпця набору даних.

Цього можна досягти, зібравши всі значення для кожного стовпця в список і обчисливши середнє значення і стандартне відхилення в цьому списку. Після обчислення ми можемо зібрати статистичні дані у список або кортеж статистичних даних. Потім повторимо цю операцію для кожного стовпця в наборі даних і повернемо список кортежів статистики.

У лістингу 3.8 наведена функція з назвою *summarize_dataset(),* яка реалізує цей підхід. Він використовує деякі хитрощі Python, щоб скоротити кількість необхідних рядків.

```
  # Calculate the mean, stdev and count for each column in a dataset
1
  def summarize_dataset(dataset):
2
          summaries = [(mean(column), stdev(column), len(column)) for column in
3
  zip(*dataset)]
4
          del(summaries[-1])
5
          return summaries
```

Лістинг 3.8 - Код для окремої обробки стовпців

При запуску програми, вона виводить список кортежів статистичних даних для кожної з двох вхідних змінних.

Інтерпретуючи результати виконання програми, середнє значення X1 становить 5,178333386499999, а стандартне відхилення X1 становить 2,7665845055177263.

Тепер ми готові використовувати ці функції для кожної групи рядків у нашому наборі даних.

3.5 Підсумування даних за класами

На цьому кроці потрібна статистика з навчального набору даних, організованого за класами.

Вище було розроблено функцію separat_by_class() для розділення набору даних на рядки за класами. Та функцію summarize_dataset() для обчислення підсумкової статистики для кожного стовпця.

У лістингу 3.9 наведена функція під назвою summarize_by_class(), яка реалізує цю операцію. Набір даних спочатку розбивається за класами, потім для кожної підмножини розраховується статистика. Результати у вигляді списку кортежів статистики потім зберігаються в словнику за значенням їх класу.

```
1 # Split dataset by class then calculate statistics for each row
2 def summarize_by_class(dataset):
3         separated = separate_by_class(dataset)
4         summaries = dict()
5         for class_value, rows in separated.items():
6                 summaries[class_value] = summarize_dataset(rows)
7         return summaries
```

Лістинг 3.9 - Сбор статистики

Запуск програми обчислює статистичні дані для кожної вхідної змінної та друкує їх, упорядковані за значенням класу. Інтерпретуючи результати, які зображені у лістингу 3.10, видно, що значення X1 для рядків для класу 0 мають середнє значення 2,7420144012.

```
1 0
2 (2.7420144012, 0.9265683289298018, 5)
3 (3.0054686692, 1.1073295894898725, 5)
4 1
5 (7.6146523718, 1.2344321550313704, 5)
6 (2.9914679790000003, 1.4541931384601618, 5)
```

Лістинг 3.10 - Результати підсумовування даних за класами

3.6 Функція щільності гаусової ймовірності

Розрахувати ймовірність або ймовірність спостереження даного реального значення, такого як X1, важко.

Один із способів зробити це — припустити, що значення X1 отримані з розподілу, такого як крива дзвіночка або гаусівський розподіл.

Гаусове розподіл можна підсумувати, використовуючи тільки два числа: середнє значення і стандартне відхилення. Тому за допомогою функції розподілу ймовірностей Гаусса можемо оцінити ймовірність заданого значення. Ця функція розраховується за формулою 3.4:

$$f(x) = \frac{1}{\sqrt{2\pi}}\sigma e^{-\frac{x-M^2}{2\sigma^2}} \quad (3.4)$$

де σ - це стандартне відхилення для x ;

M – це середнє значення для x;

π – значення pi.

У лістингу 3.11 наведена функція, яка реалізує це.

```
1 # Calculate the Gaussian probability distribution function for x
2 def calculate_probability(x, mean, stdev):
3         exponent = exp(-((x-mean)**2 / (2 * stdev**2 )))
4         return (1 / (sqrt(2 * pi) * stdev)) * exponent
```

Лістинг 3.11 - Код функції щільності

Протестуємо, як це працює. У лістингу 3.12 наведено кілька робочих прикладів

```
1  # Example of Gaussian PDF
2  from math import sqrt
3  from math import pi
4  from math import exp
5
6  # Calculate the Gaussian probability distribution function for x
7  def calculate_probability(x, mean, stdev):
8          exponent = exp(-((x-mean)**2 / (2 * stdev**2 )))
9          return (1 / (sqrt(2 * pi) * stdev)) * exponent
10
11 # Test Gaussian PDF
12 print(calculate_probability(1.0, 1.0, 1.0))
13 print(calculate_probability(2.0, 1.0, 1.0))
14 print(calculate_probability(0.0, 1.0, 1.0))
```

Лістинг 3.12 - Тестування функції щільності

Запуск коду друкує ймовірність деяких введених значень. У лістингу 3.13 видно, що коли значення дорівнює 1, а середнє значення та стандартне відхилення дорівнює 1, вхід є найбільш вірогідним і має ймовірність 0,39.

Коли ми зберігаємо статистику незмінною і змінюємо значення x на 1, ймовірності цих вхідних значень однакові і дорівнюють 0,24.

```
1 0.3989422804014327
2 0.24197072451914337
3 0.24197072451914337
```

Лістинг 3.13 - Результати роботи функції щільності.

3.7 Ймовірності класів

Тепер потрібно використати статистичні дані, розраховані на основі наших навчальних даних, щоб обчислити ймовірності для нових даних.

Імовірності розраховуються окремо для кожного класу. Це означає, що спочатку обчислюємо ймовірність того, що новий фрагмент даних належить до першого класу, потім обчислюємо ймовірність того, що він належить до другого класу, і так далі для всіх класів.

Імовірність того, що частина даних належить до класу, обчислюється за формулою 3.5.

$$P(\text{дані}) = P(X|\text{клас})P(\text{клас}) \quad (3.5)$$

Формула відрізняється від описаної вище теореми Байєса. Для спрощення обчислень поділ вилучено.

Це означає, що результат більше не є суворою ймовірністю належності даних до класу. Значення все ще максимізується, тобто обчислення для класу, що призводить до найбільшого значення, приймається як передбачення. Це звичайне спрощення реалізації, оскільки нас часто більше цікавить прогноз класу, а не ймовірність.

Вхідні змінні розглядаються окремо, тому методика отримала назву «наївна».Для наведеного вище прикладу, де є 2 вхідні змінні, обчислення ймовірності того, що рядок належить до першого класу 0 можна обчислити за формулою 3.6.

$$P(x_{1,}x_2) = P(\text{клас} = 0)P(\text{клас} = 0)P(\text{клас} = 0) \qquad (3.6)$$

З формули вище видно, чому потрібно розділяти дані за значенням класу. Функція щільності гаусової ймовірності на попередньому кроці – це спосіб обчислення ймовірності реального значення, такого як X1, і підготовлені статистичні дані використовуються в цьому розрахунку.

У лістингу 3.14 наведена функція з назвою Calculate_class_probabilities(), яка пов’язує все це разом.

Вона приймає набір підготовлених підсумків і новий рядок як вхідні аргументи.

Спочатку загальна кількість записів про навчання розраховується з підрахунків, що зберігаються в підсумковій статистиці. Це використовується при розрахунку ймовірності даного класу або P(class) як відношення рядків із заданим класом усіх рядків у навчальних даних.

Далі обчислюються ймовірності для кожного вхідного значення в рядку з використанням функції щільності ймовірності Гаусса та статистичних даних для цього стовпця та цього класу. Імовірності помножуються разом у міру їх накопичення.

Цей процес повторюється для кожного класу в наборі даних.

Нарешті повертається словник ймовірностей з одним записом для кожного класу.

```
# Calculate the probabilities of predicting each class for a given row
def calculate_class_probabilities(summaries, row):
    total_rows = sum([summaries[label][0][2] for label in summaries])
    probabilities = dict()
    for class_value, class_summaries in summaries.items():
        probabilities[class_value] = summaries[class_value][0][2]/float(total_rows)
        for i in range(len(class_summaries)):
            mean, stdev, count = class_summaries[i]
            probabilities[class_value] *= calculate_probability(row[i], mean, stdev)
```

Лістинг 3.14 - Код результуючої функції

Запуск програми виводить ймовірності, обчислені для кожного класу.

У лістингу 3.15 наведений вихід программ для заданого набору даних. Ймовірність того, що перший рядок належить до класу 0 (0,0503) вища, ніж ймовірність того, що він належить до класу 1 (0,0001).

```
{0: 0.05032427673372075, 1: 0.00011557718379945765}
```

Лістинг 3.15 - Ймовірності для кожного класу

3.8 Обговорення результатів дослідження

Медична інтелектуальна система (MIC) це програмний продукт, головним призначенням якого є агентна автоматизація всіх головних процесів, пов'язаних з наданням медичних послуг. За кордоном прийнято застосовувати термін HIS (Hospital Information System) – госпітальна інформаційна система для комплексного керування всіма процесами медобслуговування, включаючи юридичний аспект. Доповненнями до неї можуть бути специфічні модулі - агенти, наприклад, RIS (Radiology Information System) - радіологічна інформаційна система агент або PACS (Picture Archiving and Communication System) - система збереження медичних зображень. Окремий вид MIC - лабораторні інформаційні агентні системи (Laboratory Information Management Systems) і аптечні інформаційні системи.

Тут можливо використати простий варіант статистичного аналізу, до якого входять еталонний вектор хвороби , частота виявлення й ваговий коефіцієнт кожного симптому без врахування статистичного аналізу впливу симптомів на діагноз. У медичній практиці діагностування використовується таблиця з відображенням зв’язку «симптом-хвороба». Таким чином, будь-якому діагнозу відповідає певний набір «симптомів – ознак», які можна представити у вигляді вектора відповідності , де ставиться у відповідність «1» і «0»(Формула 3.7):

$$B_{D_j^o} = B_{D_j^o}\left(\delta_{D_1^o}, \delta_{D_2^o}, \dots, \delta_{D_{k+1}^o}\right), \text{ де } \delta_{i,j}\begin{cases}1, -\text{якщо це захворювання має симптом } Si_k\\ 0, -\text{інакше}\end{cases} \quad (3.7)$$

Цей простий, відомий засіб статистичного аналізу може бути надалі використовуватися у подальшому розвитку засобів діагностування.

Але при реальному діагностуванні з’являється проблема невизначеності. Це велика проблема на шляху до постановки точного діагнозу, так як заважає вибрати краще рішення, і, отже, може стати причиною неякісної відповіді

щодо остаточного діагнозу. У даній роботі під невизначеністю розуміється недостовірність наявності симптому, тобто суб'єктивність оцінювання специфічності симптому у симптомокомплексу.

Вирішення проблеми невизначеності у зазначеній методиці вирішується за рахунок використання байєсовського виводу. Варіативність початкових даних видно у таблиці 3.1.

Таблиця 3.1 – Перераховані значення апріорних і умовних гіпотез

p(…)/i	1	2	3
$p(H_i)$	0,59	0,39	0,02
$p(E_1\|H_i)$	0,49	0,89	0,39
$p(E_2\|H_i)$	0,09	0,79	0,99

При цьому початкові гіпотези характеризують подію, пов’язану з визначенням надійності деякої хвороби:

- H_1 – "пневмонія";
- H_2 – " Covid";
- H_3 – "туберкульоз".

Подіями, умовно незалежними свідоцтвами, що є, підтримують початкові гіпотези є:

- E_1 – "висока температура";
- E_2 – "кашель".

В процесі збирання фактів ймовірності гіпотез підвищуватимуться, якщо факти підтримують їх або зменшуватись, якщо спростовують їх. Припустимо, що ми маємо тільки одне свідоцтво E_1 – "висока температура" (тобто з вірогідністю одиниця наступив факт E_1 – "висока температура"). Спостерігаючи E_1 – "висока температура" ми обчислюємо апостеріорну ймовірність для гіпотез згідно формули Байєса для одного свідоцтва(Формула 3.8).

$$p(H_i\,|E_1) = \frac{p(E_1|H_i)*p(H_i)}{\sum_{k=1}^{3} p(E_1|H_k)*p(H_k)}\,, i = 1,2,3. \quad (3.8)$$

Таким чином, обчислемо вірогідності для пневмонії, ковіду, та туберкульозу. Обчислення представленні у формулах 3.9, 3.10 та 3.11 відповідно.

$$p(H_1\,|E_1) = \frac{0{,}49*0{,}59}{0{,}49*0{,}59+0{,}89*0{,}39+0{,}39*0{,}02} = 0{,}4489; \quad (3.9)$$

$$p(H_2\,|E_1) = \frac{0{,}89*0{,}39}{0{,}49*0{,}59+0{,}89*0{,}39+0{,}39*0{,}02} = 0{,}5390; \quad (3.10)$$

$$p(H_3\,|E_1) = \frac{0{,}39*0{,}02}{0{,}49*0{,}59+0{,}89*0{,}39+0{,}39*0{,}02} = 0{,}0121\,. \quad (3.11)$$

Робимо перевірку, сума апостеріорних ймовірностей повинна дати одиницю. Формула та обчислення наведененні у формулах 3.12 та 3.13:

$$p(H_1\,|E_1) + p(H_2\,|E_1) + \; p(H_3\,|E_1) = 1 \quad (3.12)$$

$$0{,}4489 + 0{,}5390 + 0{,}0121 = 1. \quad (3.13)$$

Після того, як E_1 – "висока температура" відбулась довіра до гіпотез H_1 – "пневмонія" і H_3 – "туберкульоз" знизилася, тоді як довіра до H_2 – " Covid" зросла. У тих випадках, коли є факти, підтверджуючі як подію E_1 – "висока температура" так і подію E_2 – "кашель", то апостеріорна ймовірність початкових гіпотез також може бути обчислена за правилом Байєса (Формула 3.14).

$$p(H_i\,|E_1E_2) = \frac{p(E_1E_2|H_i) * p(H_i)}{\sum_{k=1}^{3} p(E_1E_2|H_k) * p(H_k)}\,, i = 1,2,3\,. \quad (3.14)$$

Оскільки висока температура і кашель умовно незалежні при даних гіпотезах H_i, то формулу Байєса можна переписати(Формула 1.15):

$$p(H_i\,|E_1E_2) = \frac{p(E_1|H_i) * p(E_2|H_i) * p(H_i)}{\sum_{k=1}^{3} p(E_1|H_k) * p(H_i) * p(H_k)}\,, i = 1{,}2{,}3\,. \tag{3.15}$$

Звідки, обчислемо за цією формулою вірогідності для кожної хвороби. Обчислення представленні для пневмонії, ковіду, та туберкульозу у формулах 3.16, 3.17 та 3.18 відповідно.

$$p(H_1\,|E_1E_2) = \frac{0{,}49*0{,}09*0{,}59}{0{,}49*0{,}09*0{,}59+0{,}89*0{,}79*0{,}39+0{,}39*0{,}99*0{,}02} = 0{,}0845; \tag{3.16}$$

$$p(H_2\,|E_1E_2) = \frac{0{,}89*0{,}79*0{,}39}{0{,}49*0{,}09*0{,}59+0{,}89*0{,}79*0{,}39+0{,}39*0{,}99*0{,}02} = 0{,}8904; \tag{3.17}$$

$$p(H_3\,|E_1E_2) = \frac{0{,}39*0{,}99*0{,}02}{0{,}49*0{,}09*0{,}59+0{,}89*0{,}79*0{,}39+0{,}39*0{,}99*0{,}02} = 0{,}0251. \tag{3.18}$$

Перевірка за формулою рівності вірогідностей. Загальна формула та її обчислення представленно у формулі 3.19 та формулі 3.20 відповідно.

$$p(H_1\,|E_1E_2) + \; p(H_2\,|E_1E_2) + p(H_3\,|E_1E_2) = 1\,. \tag{3.19}$$

$$0{,}0845 + 0{,}8904 + 0{,}0251 = 1. \tag{3.20}$$

Початковим ранжуванням було H_1 – "пневмонія", H_2 – " Covid" та H_3 – "туберкульоз", і всі три залишилися після отримання ознак E_1 – "висока температура" і E_2 – "кашель". При цьому бронхіт ймовірніше, ніж пневмонія та туберкульоз. Це свідчить про те, що маючи кашель та високу температуру ймовірність захворювання бронхітом набагато більша, ніж ймовірність захворювання пневмонією чи туберкульозом.

Однак реально, поширення ймовірності відбувається поетапно з підсумовуванням окремих свідоцтв і їх впливу на умовну вірогідність у міру надходження окремих E_i. Це можна зробити, використовуючи апріорну і апостеріорну ймовірність, таким чином:

1. Задаємо $p(H_i)$ – апріорну ймовірність подій H_i.
2. Для одержаних ознак E_j записуємо $p(E_j|H_i)$.
3. З врахуванням теореми Байєса підраховуємо $p(H_i|E_j)$ залежно від результату E_j, тобто обчислюємо апостеріорну ймовірність події H_i.
4. Тепер можна не звертати уваги всі настали E_j і перезначити поточну апостеріорну ймовірність події H_i, як нову апріорну ймовірність H_i. Отже, $p(H_i)$ рівна $p(H_i|E_j)$ залежно від значення E_j.
5. Потім виберемо нове свідоцтво для розгляду і перейдемо до п.2.

Проілюструємо цю послідовність по формулам 3.21, 3.22, 3.23 на приведеному вище прикладі в припущенні, що спочатку поступило свідоцтво E_2 – "кашель". Тоді:

$$p(H_1 |E_2) = \frac{0{,}09*0{,}59}{0{,}09*0{,}59+0{,}79*0{,}39+0{,}99*0{,}02} = 0{,}1394; \quad (3.21)$$

$$p(H_2 |E_2) = \frac{0{,}79*0{,}39}{0{,}09*0{,}59+0{,}79*0{,}39+0{,}99*0{,}02} = 0{,}8087; \quad (3.22)$$

$$p(H_3 |E_2) = \frac{0{,}99*0{,}02}{0{,}09*0{,}59+0{,}79*0{,}39+0{,}99*0{,}02} = 0{,}0519. \quad (3.23)$$

Перевіремо за фомулою 3.24 сумарну вірогідність. Обчислення наведено у формулі 3.25:

$$p(H_1 |E_2) + p(H_2 |E_2) + p(H_3 |E_2) = 1 \quad (3.24)$$

.

$$0{,}1394 + 0{,}8087 + 0{,}0519 = 1 \quad (3.25)$$

Одержану ймовірність можна прийняти за нову апостеріорну вірогідність гіпотез H1, H2 та H3, тобто:

- $\mathrm{p}(\tilde{\mathrm{H}}_1) = 0{,}1394$;
- $\mathrm{p}(\tilde{\mathrm{H}}_2) = 0{,}8087$;
- $\mathrm{p}(\tilde{\mathrm{H}}_3) = 0{,}0519$.

І, якщо тепер додатково поступить свідоцтво E_2 – "кашель", то нова апостеріорна ймовірність гіпотез може бути обчислена тільки на основі свідоцтва, що знов поступило. Обчислення наведені у формулах 3.26, 3.27 та 3.28 для пневмонії, ковіду та туберкульозу відповідно:

$$p(H_1 \,|E_1E_2) = \mathrm{p}(E_1) = \frac{0{,}49*0{,}1394}{0{,}49*0{,}1394+0{,}89*0{,}8087+0{,}39*0{,}0519} = 0{,}0845\,; \qquad (3.26)$$

$$p(H_2 \,|E_1E_2) = \mathrm{p}(E_1) = \frac{0{,}89 * 0{,}8087}{0{,}49 * 0{,}1394 + 0{,}89 * 0{,}8087 + 0{,}39 * 0{,}0519} = 0{,}8905 \qquad (3.27)$$

$$p(H_3 \,|E_1E_2) = \mathrm{p}(E_1) = \frac{0{,}39 * 0{,}0519}{0{,}49 * 0{,}1394 + 0{,}89 * 0{,}8087 + 0{,}39 * 0{,}0519} = 0{,}0250 \qquad (3.28)$$

З приведеного прикладу видно, що ітераційна процедура послідовного розподілу ймовірності у міру надходження свідоцтв дозволяє отримати результати аналогічні безпосередньому застосування правила Байєса для випадку одночасного двох свідоцтв, що поступили. Значення гіпотези H_2 – "Covid" є найбільш вірогідним, аніж H_1 – "пневмонія" та H_3 – "туберкульоз".

МІС, побудовані на об'єктивних і суб'єктивних поглядах на поняття ймовірності події. У базі знань накопичують людські знання. Тому інтерпретації на основі суб'єктивної довіри найкраще підходять для представлення знань експертів у світлі ймовірностей. Як наслідок, більшість сучасних МІС, які використовують теорію ймовірностей, є «байєсовськими».

Використання байєсівської стратегії в МІС реалізує з використанням байєсовської формули "зворотних ймовірностей", тобто оцінки умовних

ймовірностей гіпотез. При наявності декількох ознак (симптомів) такі розрахунки просто реалізуються в припущенні статистичної незалежності характеристик, що далеко не завжди відповідає дійсності. Однак практика показує, що такий підхід, незважаючи на його очевидну математичну некоректність, цілком застосовний, оскільки він зазвичай призводить до правильних висновків.

МІС, що використовують теорію суб'єктивних ймовірностей, широко використовуються як у медицині, так і в інших застосуваннях, де необхідно чітко і значимо визначати ймовірність виникнення певної хвороби. При розрахунках розподілу ймовірностей в МІС з заданими гіпотезами, - «пневмонія», - «Covid», - «туберокульоз», що характеризує хворобу, пов'язану з визначенням деяких захворювань. Отриманий результат вказує на ймовірність появи Covid у діагнозі пацієнта, відносно зазначених двох інших захворювань.

ВИСНОВКИ

Це дослідження мало дві цілі: допомогти зацікавленим сторонам у нинішній системі охорони здоров’я та допомогти дослідникам, які застосовують методи аналізу даних у сфері охорони здоров'я.

Комбінації атрибутів, обрані як найкращі предиктори обраних цільових атрибутів, які самі по собі є показником результату лікування або показником корисності для зацікавлених сторін у нинішній системі охорони здоров’я. Зацікавлені сторони можуть прийняти рішення на основі такого типу інформації.

Крім того, було виявлено вплив вибору ознак на точність класифікації у всій літературі. Найкращі комбінації атрибутів, визначені тут, можуть виявитися корисними дослідникам, які шукають зменшення розмірності для власних досліджень. Ці комбінації з атрибутів можуть бути вхідними для більш витончених моделей, пристосованих до конкретних атрибутів.

Комбінація наївного байесового класифікатора з жадібним вибором ознак послідовно визначило найпередбачуваніші атрибути для даного цільового атрибута. Для кожного вибору цілі найбільш передбачуваними атрибутами, окремо або в поєднанні, були ті, що стосуються діагностики або процедурні коди. Цей результат вказує на деякі уточнення або розширення, які, на нашу думку, варто розглянути.

Одним з аспектів NHDS 2010 року, який ми до кінця не вивчили, є конкретний діагноз і процедурні коди. Оскільки вони зазвичай не ранжуються після першого з кожного, їх позиції у записах кожного пацієнта не є інформативними. Вони можуть бути представлені у вигляді двійкового масиву атрибутів, кожен із яких представляє індивідуальний код.

Може бути цікаво розглянути, чи є певні коди сильнішими провісниками, ніж первинний код або набір кодів, однак цим зусиллям може перешкоджати вхідний простір.

NHDS (НЦСЗ, 2010b) проводиться щорічно з 1965 року і охоплює досить узгоджений набір елементів даних; зміни обсягу та методології були добре задокументовані.

Розширення цих зусиль, щоб включити дані за більше років, може надати більшу впевненість у вхідних даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. How does Machine Learning work [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/machine-learning.(дата звернення 14.09.2021)
2. Logistic Regression in Machine Learning [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/logistic-regression-in-machine-learning. (дата звернення 14.09.2021)
3. Support Vector Machine Algorithm [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm. (дата звернення 14.09.2021)
4. Naïve Bayes Classifier Algorithm [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/machine-learning-naive-bayes-classifier. (дата звернення 20.10.2021)
5. Decision Tree Classification Algorithm [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm. (дата звернення 15.09.2021)
6. Random Forest Algorithm [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/machine-learning-random-forest-algorithm. (дата звернення 16.09.2021)
7. Беллацці Р. Прогнозний аналіз даних у клінічній медицині: Актуальні проблеми та настанови. Міжнародний журнал медичної інформатики / Р. Беллацці, Б. Зупан., 2008. – 81 с.
8. Cios K. J. Унікальність аналізу медичних даних. Штучний інтелект в медицині / K. J. Cios, W. W. Moore., 2002. – 24 с.
9. Оцінка методів машинного навчання для прогнозування смертності від пневмонії. Штучний інтелект в медицині / Купер Г.Ф., Аліферіс К.Ф., Амбросіно Р. та ін.]., 1997. – 138 с.

10. Хассан С.З. Гібридний підхід аналізу даних для отримання знань та класифікація в медичних базах даних. / Хассан С.З., Верма Б., 2007.
11. Попеску М. Покращення прогнозування захворювань за допомогою особливостей онтологічного МКБ-9 / Попеску М., Халілія М.., 2011.
12. Вибір і класифікація ознак. Побудова моделі на основі даних пацієнтів з цукровим діабетом 2 типу. Штучний інтелект у медицині / Хуанг Ю., МакКаллах П., Блек Н., Харпер Р., 2007.
13. Сьома міжнародна конференція з проектування інтелектуальних систем / L. M.Mourelle, N. Nedjah, J. Kacprzyk, A. Abraham.
14. Домінгос П. Кілька корисних речей про машинне навчання. / Домінгос П., 2012. – 55 с.
15. Національне обстеження виписки з лікарні: суспільне використання документація файлу даних. [Електронний ресурс] // Національний центр статистики здоров'я. – 2010. – Режим доступу до ресурсу: http://ftp.cdc.gov/pub/Health_Statistics/NCHS/(дата звернення 20.10.2021)
16. Ріш І. Аналіз характеристик даних, які впливають з наївного Байєса / Ріш І., Хеллерштейн Дж., Татхачар Дж. – Готорн, Нью-Йорк: IBM TJ Watson, 2001.