 МІНІСТЕРСТВО ОСВІТИ I НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ   
 ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ БІОМЕДИЧНОЇ ІНЖЕНЕРІЇ

КАФЕДРА БІОМЕДИЧНОЇ КІБЕРНЕТИКИ

## «Класифікатор на основі дерева розв’язків СART. Діагностика захворювань»

**Виконав:**

студент гр. БС-62

Кривошеєв Е.В.

**Керівник:**

доц.каф БМК

Добровськая Л.М.

Київ - 2018

**Зміст**

1. Вступ ......................................................................................... 3
2. Анотація .................................................................................... 4
3. Основна частина ....................................................................... 7
4. Висновки .................................................................................... 22
5. Список використаних джерел ................................................... 23
6. Додатки з допоміжним матеріалом ......................................... 24

**Вступ**

У деяких випадках при прийняття, рішень ми допускаємо помилки, які мають наслідки різної ступенені значущості. Тому одним, з безперечних умов успішної, діяльності незалежно від характеру або специфіки, праці, є прийняття правильних рішень. Що незавжди вдається навіть досвідченим працівникам через перевтому, особисті проблеми або інше. Також помилки можуть допускаються через неповноту інформації про об'єкт, обмеженості часових ресурсів, або якщо об'єкт досить складний і немає можливості прорахувати і вибрати найкраще рішення.

В даний час однією з основних тенденцій розвитку суспільства є її тотальна інформатизація. Стрімко розвиваються комп'ютерні технології, здатні накопичувати і обробляти великі обсяги цифрової інформації. Найчастіше в масивах даних містяться необхідні для прийняття успішних рішень не дуже помітні зв’язки, виявити які можна за допомогою сучасних методів аналізу даних. У більш складних випадках, доцільно побудувати математичну модель і приймати рішення відповідно до результатів проведених за моделлю розрахунків. Для вирішення цих завдань і створюються спеціалізовані інформаційні системи.

Дерева рішень - це статистичний метод, що дозволяє передбачати приналежність спостережень або об'єктів до того чи іншого класу категоріально залежної змінної або значення кількісної змінної в залежності від відповідних значень однієї або декількох незалежних змінних.

Переваги CART методу :

* 1. Даний метод є непараметричним, це означає, що для його застосування немає необхідності розраховувати різні параметри імовірнісного розподілу.
  2. Для застосування алгоритму CART немає необхідності заздалегідь вибирати змінні, який братимуть участь в аналізі: змінні відбираються безпосередньо під час проведення аналізу на підставі значення індексу Gini.
  3. CART легко бореться з викидами: механізм «розбиття» (від англ. Splitting), закладений в алгоритмі просто поміщає «викиди» в окремий вузол, що дозволяє очистити наявні дані від шумів.
  4. Для застосування цього алгоритму не треба брати до уваги ніяких припущень або припущень перед проведенням аналізу.
  5. Великою перевагою є швидкість роботи алгоритму.

В медицині дерева рішень застосовуються для діагностики різних захворювань, що полегшує працю лікарів, не допускає помилки через людський фактор, та дає можливість зосередитися на нетривіальних хворобах пацієнтів.

**Анотація**

В даній роботі представлено реалізацію класифікатора на основі дерева розв’язків CART для встановлення медичного діагнозу.

Виконана робота складається з трьох частин. Перша: створення класифікатора на основі дерева розв’язків CART. Друга: її навчання та тестування на основі медичних даних . Третя: висновки що до реалізованої системи на основі отриманих даних.

Мета : вирішення поставленої практичної навчальної задачі, а саме створення класифікатора на основі дерева розв’язків CART з використанням вивченого на лекційних та практичних заняттях, а також самостійно вивченого теоретичного матеріалу з обов’язковим застосуванням навчального матеріалу.

Робота складається з 23сторінок, у тому числі в основній частині — 15сторінок, враховуючи додаток — 27. Присутня з 1 ілюстрація та 2 таблиці*.* Використано 6джерел.

**Ключові слова**

1. CART

### Дерева рішень

### Коефіцієнт Gini

# Навчальна вибірка

### Тестова вибірка

1. Класифікатор

**Aннотация**

В данной работе представлены реализацию классификатора на основе дерева решений CART для установления медицинского диагноза.

Выполненная работа состоит из трех частей. Первая: создание классификатора на основе дерева решений CART. Вторая: его обучения и тестирования на основе медицинских данных. Третья: выводы к реализованной системы на основе полученных данных.

Цель: решение поставленной практической учебной задачи, а именно создание классификатора на основе дерева решений CART с использованием изученного на лекционных и практических занятиях, а также самостоятельно изученного теоретического материала с обязательным применением учебного материала.

Работа состоит из 23 страниц, в том числе в основной части - 15 страниц, учитывая приложение - 27. Присутствует 1 илюстрация и 2 таблицы . Использовано 6 источников.

Ключевые слова

1. CART

2. Деревья решений

3. Коэффициент Gini

4. Учебная выборка

5. Тестовая выборка

6. Классификатор

**Abstract**

In this work, the implementation of a classifier based on the tree of solutions to the CART for the establishment of medical diagnosis.

The completed work consists of three parts. First: creating a classifier based on the tree of solutions CART. Second: her training and testing on the basis of medical data. Third: the findings of the implementation of the system based on the data.

Aim: to solve a given practical task, namely the creation of a classifier based on the tree of solutions CART using learned in lectures and practical classes, as well as independently studied theoretical material with the obligatory application of educational material.

The work consists of 23 pages, including the main part - 15 pages, taking into account the application - 27. There are 1 illustration and 2 tables. 6 sources were used.

Keywords

* 1. CART

2. Tree making

3. Gini coefficient

4. Educational sample

5. Test sample

6. Classifier

**Основна частина**

**Класифікатор на основі дерева розв’язків СART**

Дерево прийняття рішень або деревом класифікації – засіб що використовується для прийняття рішень, та має застосування в машинному навчанні, аналізі даних і статистиці. Структура дерева являє собою «листя» в яких записані значення цільової функції. і «гілки» на яких записані атрибути, від яких залежить цільова функція. Щоб класифікувати нові дані, треба спуститися по дереву до листа і видати відповідне значення. Подібні ДР широко використовуються в медицині наприклад для встановлення діагнозу. Мета полягає в тому, щоб створити модель, яка передбачає значення цільової змінної на основі змінних на вході.

Алгоритм CART призначений для побудови бінарного дерева рішень. Бінарні дерева також називають двійковими, значить, що кожен вузол дерева при розбитті має тільки двох нащадків. Для алгоритму CART «поведінка» об'єктів виділеної групи означає частку модального значення вихідного ознаки. Виділені групи - ті, для яких ця частка досить висока. На кожному кроці побудови дерева правило, формується в вузлі, ділить заданий безліч прикладів на дві частини - частина, в якій виконується правило (нащадок - right) і частина, в якій правила не виконується (нащадок - left). Перевагою алгоритму CART є певна гарантія того, що, якщо шукані детермінації існують в досліджуваній сукупності, то вони будуть виявлені. Крім того, CART дозволяє не «Замикатися» на єдиному значенні вихідного ознаки, а шукати все такі його значення, для яких можна знайти відповідне пояснює вираз.

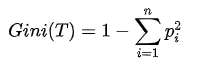
Метод CART застосовується для номінальних (зазвичай дворівневих) і порядкових предикторних змінних. У цьому методі перебираються всі можливі варіанти розгалуження для кожного вузла, і вибирається та предикторна змінна, при якій оцінна функція дає найкращий показник.

Правила розбиття

Вектор предикторних змінних, що подається на вхід дерева може містити як числові (порядкові) так і категоріальні змінні. У будь-якому випадку в кожному вузлі розбиття йде тільки по одній змінній. Якщо змінна числового типу, то в вузлі формується правило виду Хi <= c. Де с - деякий поріг, який найчастіше вибирається як середнє арифметичне двох сусідніх упорядкованих значень змінної xi навчальної вибірки. Якщо змінна категоріального типу, то в вузлі формується правило Xi V (Xi), де V (Xi) - деякий непорожнє підмножина безлічі значень змінної Xi в навчальній вибірці. Отже, для n значень числового атрибута алгоритм порівнює n-1 розбиття, а для категоріального (2n-1 - 1). На кожному кроці побудови дерева алгоритм послідовно порівнює всі можливі розбиття для всіх атрибутів і вибирає найкращий атрибут і найкращу розбивку для нього.

Оцінка якості моделі

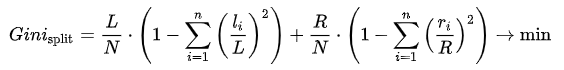
Оціночна функція, яка використовується алгоритмом CART, базується на інтуїтивної ідеї зменшення невизначеності (неоднорідності) в вузлі. Як приклад розглянемо задачу з двома класами і вузлом, що має по 50 прикладів кожного класу. Вузол має максимальну невизначеність. Якщо буде знайдено розбиття, яке розбиває дані на дві підгрупи 40: 5 прикладів в одній і 10:45 у інший, то інтуїтивно неоднорідність зменшиться. Вона повністю зникне, коли буде знайдено розбиття, яке створить підгрупи 50: 0 і 0:50. В алгоритмі CART ідея невизначеності формалізована в індексі Gini. Якщо набір даних Т містить дані n класів, тоді індекс Gini визначається наступним чином .



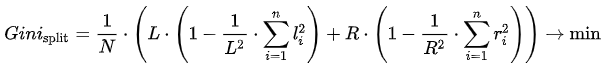
Де pi - ймовірність (відносна частота) класу i в T. Якщо набір Т розбивається на дві частини Т1 і Т2 з числом прикладів в кожному N1 і N2 відповідно, тоді показник якості розбиття буде дорівнює:



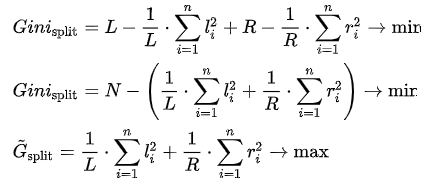
Найкращим вважається те розбиття, для якого Ginisplit (T) мінімально. Позначимо N - число прикладів у вузлі - предка, L, R - число прикладів відповідно в лівому і правому нащадку, li і ri - число примірників i-го класу в лівому / правому нащадку. Тоді якість розбиття оцінюється за такою формулою:



Щоб зменшити обсяг обчислень формулу можна перетворити:



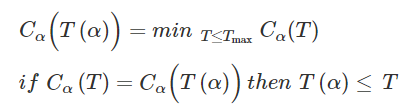
Так як множення на константу не грає ролі при мінімізації:



Механізм відсікання

Механізм відсікання дерева, оригінальна назва minimal cost-complexity tree pruning - найбільш серйозна відмінність алгоритму CART від інших алгоритмів побудови дерева. CART розглядає відсікання як отримання компромісу між двома проблемами: отримання дерева оптимального розміру і отримання точної оцінки ймовірності помилкової класифікації.

Основна проблема відсікання - велика кількість всіх можливих відсічених піддерев для одного дерева. Більш точно, якщо бінарне дерево має | T | - листів, тоді існує ~ [1.5028369 | T |] відсічені піддерев. І якщо дерево має хоча б 1000 лисів, тоді число відсічені піддерев стає просто величезним. Базова ідея методу - не розглядати всі можливі піддерева, обмежившись лише "кращими представниками" згідно наведеної нижче оцінки. Позначимо | T | - число листів дерева, R (T) - помилка класифікації дерева, що дорівнює відношенню числа неправильно класифікованих прикладів до числа прикладів в навчальній вибірці. Визначимо Cα (T) - повну вартість (оцінку / показник витрати-складність) дерева Т як: Cα (T) = R (T) + α \* | T |, де | T | - число листів (термінальних вузлів) дерева - деякий параметр, що змінюється від 0 до + ∞. Повна вартість дерева складається з двох компонент - помилки класифікації дерева і штрафу за його складність. Якщо помилка класифікації дерева незмінна, тоді зі збільшенням повна вартість дерева буде збільшуватися. Тоді менш гіллясте дерево, що дає більшу помилку класифікації може коштувати менше, ніж те що дає меншу помилку, але більш гіллясте. Визначимо Tmax - максимальне за розміром дерево, яке варто обрізати. Якщо ми зафіксуємо значення α, тоді існує найменше мінімізоване піддерево α, яке виконує наступні умови:



Перша умова говорить, що не існує такого піддерева дерева Tmax, яке мало б меншу вартість, ніж T (α) при цьому значенні α. Друга умова говорить, що якщо існує більше одного піддерева, що має дану повну вартість, тоді ми вибираємо найменше дерево. Можна показати, що для будь-якого значення існує таке найменше мінімізоване дерево. Але це завдання не тривіальне. Не може бути такого, коли два дерева досягають мінімуму повної вартості і жодне з них не є піддерево іншого. Можна побудувати послідовність зменшуються піддерев дерева Tmax: T1> T2> T3> ...> {t1}, (Де t1 - кореневий вузол дерева) таку, що Tk - найменше, мінімізуєме піддерево для [αk, αk + 1). Це важливий результат, тому що це означає, що ми можемо отримати наступне дерево в послідовності, застосувавши відсікання до поточного дерева. Це дозволяє розробити ефективний алгоритм пошуку найменшого мінімізованого піддерева при різних значеннях α.

Перше дерево в цій послідовності - найменше піддерево дерева Tmax має таку ж помилку класифікації, як і Tmax, тобто T1 = T (α = 0). Внесок Tt в повну вартість дерева Т складе Cα (Tt) = R (Tt) + α | Tt |, де R (Tt) = Σt'∈TtR (t '). Дерево T - Tt буде краще, ніж Т, коли Cα {(t}) = Cα (Tt), тому що при цій величині вони мають однакову вартість, але T - Tt найменше з двох. Коли Cα {(t}) = Cα (Tt) ми отримуємо: R (Tt) + α | Tt | = R (t) + α вирішуючи для α, отримуємо: α = R (t) -R (Tt) | Tt | -1. Так для будь-якого вузла t в Т1, якщо ми збільшуємо α, тоді коли α = R (t) -R (T1, t) | T1, t | -1 дерево, отримане відсіканням в вузлі t, буде краще, ніж Т1. Основна ідея полягає в наступному: обчислимо це значення для кожного вузла в дереві Т1, і потім виберемо "слабкі зв'язки" тобто вузли для яких величина g (t) = R (t) -R (T1, t) | T1, t | -1 є найменшою. Ми відкидаємо Т1 в цих вузлах, щоб отримати Т2 - наступне дерево в послідовності. Потім ми продовжуємо цей процес для отриманого дерева і так поки ми не отримаємо кореневий вузол (дерево в якому тільки один вузол).

Вибір фінального дерева

Отже, ми маємо послідовність дерев, нам необхідно вибрати краще дерево з неї. Те, що ми і будемо використовувати надалі. Найбільш очевидним є вибір фінального дерева через тестування на тестовій вибірці. Дерево, що дало мінімальну помилку класифікації, і буде найкращим.

Перехресна перевірка

Перехресна перевірка є найбільш складною і одночасно оригінальною частиною алгоритму CART. Вона являє собою шлях вибору остаточного дерева, за умови, що набір даних має невеликий об'єм або ж записи набору даних настільки специфічні, що розділити набір на навчальну і тестову вибірку не представляється можливим .

**Особливості реалізації**

Нижче приводяться основні функції реалізовані в програмі та їх опис.

Функції :

main – запрошує шлях до вибірки, та створює 10 різних варіантів вхідних даних із заданої вибірки, що дає змогу оцінити ефективність системи.

recording – отримує шлях до даних, зчитує їх та записує в змінну.

Функції класу cart\_tree :

create\_tree – отримує дані вибірки, та створює на їх основі дерево.

calc\_best\_split – знаходить найкраще розбиття з отриманих даних.

calc\_uncertainty – розраховує коефіцієнта Gini для даного набору даних.

calc\_unique\_values – знаходить кількість унікальних значень в отриманих даних.

split\_data – отримує дані та об’єкт умови, записує результат перевірки умови.

calc\_info\_gain - вираховує зміну невизначеності.

test\_accuracy – підраховує кількість вірних прогнозів системи на тестовій вибірці.

classify– класифікує тестові данні на основі попереднього навчання.

save – зберігає результат класифікації в файл.

print\_tree – будує візуалізацію дерева.

Функції класу condition:

answer – отримує на вхід стоку з введених даних та перевіряє умову для конкретного атрибуту.

Функції класу node:

classify – класифікує текстові данні на основі попереднього навчання.

print\_tree – виводить візуалізацію вузлів.

Функції класу leaf:

class\_frequency – визначає частоту появи класу.

classify - класифікує тестові данні .

print\_tree – виводить візуалізацію листа.

**Початкові данні та результат**

Нижче наведений фрагмент початкової вибірки за атрибутами: температура, пульс, головна біль, кашель, висип на тілі, клас стану пацієнта. Всього 120 екземплярів серед яких : 40 відносяться до класу “грип”, 40 відносяться до класу “сонячний удар”, 40 відносяться до класу “кір”, 40 відносяться до класу “здоровий”.

*Таблиця 1. Вхідні данні*

|  |  |
| --- | --- |
| 38.77 , 88 , 1 , 1 , 0 , flu | 38.65 , 84 , 0 , 0 , 1 , smallpox |
| 39.99 , 98 , 1 , 1 , 0 , flu | 39.52 , 59 , 0 , 0 , 1 , smallpox |
| 37.7 , 75 , 1 , 1 , 0 , flu | 38.06 , 82 , 1 , 0 , 1 , smallpox |
| 40.65 , 62 , 0 , 1 , 0 , flu | 39.16 , 83 , 0 , 1 , 1 , smallpox |
| 36.85 , 93 , 0 , 1 , 0 , flu | 38.21 , 66 , 1 , 0 , 1 , smallpox |
| 36.96 , 74 , 1 , 1 , 0 , flu | 40.37 , 71 , 0 , 0 , 1 , smallpox |
| 42.25 , 61 , 1 , 1 , 0 , flu | 38.53 , 89 , 0 , 0 , 1 , smallpox |
| 37.89 , 81 , 1 , 1 , 0 , flu | 37.63 , 82 , 0 , 0 , 1 , smallpox |
| 41.26 , 75 , 1 , 1 , 0 , flu | 37.12 , 70 , 1 , 1 , 1 , smallpox |
| 38.97 , 90 , 1 , 1 , 0 , flu | 37.31 , 62 , 0 , 0 , 1 , smallpox |
| 41.41 , 96 , 0 , 1 , 0 , flu | 39.87 , 57 , 0 , 0 , 1 , smallpox |
| 38.32 , 96 , 1 , 1 , 0 , flu | 38.19 , 87 , 1 , 0 , 1 , smallpox |
| 39.2 , 62 , 1 , 1 , 0 , flu | 38.14 , 92 , 0 , 0 , 1 , smallpox |
| 42.01 , 102 , 1 , 1 , 0 , flu | 37.45 , 70 , 1, 0 , 1 , smallpox |
| 39.6 , 108 , 1 , 1 , 0 , flu | 37.16 , 79 , 0 , 0 , 1 , smallpox |
| 38.12 , 73 , 0 , 1 , 0 , flu | 38.86 , 77 , 0 , 0 , 1 , smallpox |
| 41.35 , 83 , 1 , 1 , 0 , flu | 38.09 , 65 , 0 , 0 , 1 , smallpox |
| 37.18 , 73 , 1 , 1 , 0 , flu | 37.81 , 56 , 0 , 0 , 1 , smallpox |
| 40.89 , 58 , 0 , 1 , 0 , flu | 38.9 , 56 , 0 , 0 , 1 , smallpox |
| 40.35 , 88 , 0 , 1 , 0 , flu | 39.06 , 107 , 0 , 0 , 1 , smallpox |
| 39.2 , 93 , 1 , 1 , 0 , flu | 37.99 , 95 , 0 , 0 , 1 , smallpox |
| 37.74 , 73 , 1 , 1 , 0 , flu | 39.18 , 71 , 0 , 0 , 1 , smallpox |
| 39.28 , 82 , 1 , 1 , 0 , flu | 37.88 , 86 , 0 , 0 , 1 , smallpox |
| 36.41 , 68 , 1 , 1 , 0 , flu | 37.21 , 83 , 0 , 1 , 1 , smallpox |
| 37.58 , 57 , 1 , 1 , 0 , flu | 38.23 , 71 , 0 , 0 , 1 , smallpox |

Повна таблиця з вхідними даними знаходиться в додатку.

Порядок виконання програми :

Введіть шлях до вибірки : pat.csv

Атрибут 5 >= 1.0:

\\_\_\_\_Так: Атрибут 1 >= 37.12:

\\_\_\_\_Так: smallpox - 100.0

\\_\_\_\_Ні: sunstroke - 100.0

\\_\_\_\_Ні: Атрибут 4 >= 1.0:

\\_\_\_\_Так: Атрибут 1 >= 36.41:

\\_\_\_\_Так: flu - 100.0

\\_\_\_\_Ні: sunstroke - 100.0

\\_\_\_\_Ні: Атрибут 2 >= 107.0:

\\_\_\_\_Так: sunstroke - 100.0

\\_\_\_\_Ні: Атрибут 3 >= 1.0:

\\_\_\_\_Так: sunstroke - 100.0

\\_\_\_\_Ні: healthy - 100.0

Ітерація - 1, Точність - 0.9500

Ітерація - 2, Точність - 0.9750

Ітерація - 3, Точність - 0.8750

Ітерація - 4, Точність - 1.0000

Ітерація - 5, Точність - 1.0000

Ітерація - 6, Точність - 0.9000

Ітерація - 7, Точність - 0.9250

Ітерація - 8, Точність - 0.9250

Ітерація - 9, Точність - 0.9500

Ітерація - 10, Точність - 0.9750

Класифікатор CART має точність 94.75%, +/- 8.32%

Скріншоти тестування знаходяться у додатку.

Знизу приведено результат останньої ітерації для наглядної демонстрації роботи класифікатору. Атрибути: температура, пульс, головна біль, кашель, висип на тілі, клас стану пацієнта, клас визначений класифікатором.

*Таблиця 2. Вихідні данні*

|  |  |
| --- | --- |
| 37.45,70.0,1.0,0.0,1.0, smallpox, smallpox | 38.19,87.0,1.0,0.0,1.0, smallpox, smallpox |
| 36.11,66.0,0.0,0.0,0.0, healthy, healthy | 40.89,58.0,0.0,1.0,0.0, flu, flu |
| 37.12,70.0,1.0,1.0,1.0, smallpox, sunstroke | 36.85,84.0,0.0,0.0,0.0, healthy, healthy |
| 36.32,125.0,1.0,0.0,0.0,sunstroke,sunstroke | 37.18,73.0,1.0,1.0,0.0, flu, flu |
| 38.04,106.0,1.0,1.0,0.0, flu, flu | 36.62,95.0,0.0,0.0,0.0, healthy, healthy |
| 39.2,101.0,1.0,1.0,0.0, flu, flu | 36.6,62.0,0.0,0.0,0.0, healthy, healthy |
| 36.25,102.0,1.0,0.0,0.0,sunstroke,sunstroke | 36.69,130.0,1.0,0.0,0.0,sunstroke,sunstroe |
| 37.99,95.0,0.0,0.0,1.0, smallpox, smallpox | 37.48,129.0,1.0,0.0,0.0,sunstroke,sunstrokе |
| 38.1,91.0,0.0,0.0,1.0, smallpox, smallpox | 38.65,84.0,0.0,0.0,1.0, smallpox, smallpox |
| 37.76,60.0,0.0,0.0,1.0, smallpox, smallpox | 36.2,131.0,1.0,0.0,0.0, sunstroke, sunstroke |
| 41.35,83.0,1.0,1.0,0.0, flu, flu | 36.71,113.0,1.0,0.0,0.0,sunstroke,sunstroke |
| 41.41,96.0,0.0,1.0,0.0, flu, flu | 36.62,85.0,0.0,0.0,0.0, healthy, healthy |
| 38.65,81.0,0.0,0.0,1.0, smallpox, smallpox | 39.2,93.0,1.0,1.0,0.0, flu, flu |
| 37.89,81.0,1.0,1.0,0.0, flu, flu | 38.14,92.0,0.0,0.0,1.0, smallpox, smallpox |
| 37.63,82.0,0.0,0.0,1.0, smallpox, smallpox | 36.53,129.0,0.0,0.0,0.0, sunstroke, sunstroke |
| 41.26,75.0,1.0,1.0,0.0, flu, flu | 36.54,59.0,0.0,0.0,0.0, healthy, healthy |
| 38.12,73.0,0.0,1.0,0.0, flu, flu | 39.99,98.0,1.0,1.0,0.0, flu, flu |
| 36.59,86.0,0.0,0.0,0.0, healthy, healthy | 37.31,62.0,0.0,0.0,1.0, smallpox, smallpox |
| 36.64,143.0,0.0,0.0,0.0,sunstroke,sunstroke | 36.47,122.0,1.0,0.0,0.0, sunstroke, sunstroke |
| 42.25,61.0,1.0,1.0,0.0, flu, flu | 36.4,139.0,1.0,0.0,0.0, sunstroke, sunstroke |

Висновки : побудований класифікатор на основі дерева розв’язків СART на підставі 10-и тестів має точність 94.75% з можливою похибкою 8.32 % . Це досить гарний результат, але він може значною мірою змінитися на практиці. Від демонстрував велику швидкість, та виявився зручним в використанні так як може працювати на будь-яких вхідних даних та є непараметричним. Також він є ефективним за рахунок стійкості до викидів, що звільняє від пошуку і видалення їх власноруч, чи за допомогою стороннього програмного податку.

**Програмна реалізація**

**cart.py**

import random

from collections import Counter

class condition(object):

def \_\_init\_\_(self, attr\_id, value):

self.attr\_id = attr\_id

self.value = value

self.num\_mode = isinstance(self.value, float) or isinstance(self.value, int)

def \_\_str\_\_(self):

comparator = ">=" if self.num\_mode else "=="

return "Атрибут {} {} {}:".format(self.attr\_id + 1, comparator, self.value)

def answer(self, instance):

if self.num\_mode:

return instance[self.attr\_id] >= self.value

else:

return instance[self.attr\_id] == self.value

class node(object):

def \_\_init\_\_(self, cond, true\_subtree, false\_subtree):

self.cond = cond

self.true\_sub = true\_subtree

self.false\_sub = false\_subtree

def print\_tree(self, offset = 0):

padding = "\n" + offset \* "\t\t" + "\\_\_\_\_"

q = str(self.cond)

true = padding + "Так: " + self.true\_sub.print\_tree( offset=offset + 1)

false = padding + "Ні: " + self.false\_sub.print\_tree( offset=offset + 1)

return q + true + false

def classify(self, instance):

if self.cond.answer(instance):

return self.true\_sub.classify(instance)

else:

return self.false\_sub.classify(instance)

class leaf(object):

def \_\_init\_\_(self, data):

self.probabil = self.class\_frequency(data)

@staticmethod

def class\_frequency(data):

classes = [row[-1] for row in data]

return Counter(classes)

def print\_tree(self, offset = 0):

total\_count = sum(self.probabil.values())

results = {}

for class\_name, count in self.probabil.items():

results[class\_name] = count \* 100 / total\_count

return ",".join(["{} - {}".format(r[0], r[1]) for r in results.items()])

def classify(self, \_):

return self.probabil

class cart\_tree(object):

def \_\_init\_\_(self, data):

self.tree = self.create\_tree(data)

@staticmethod

def calc\_uncertainty(data):

uncer = 1

count\_of\_labels = Counter([row[-1] for row in data])

num\_instances = len(data)

for label, count in count\_of\_labels.items():

prob\_of\_matching = (count/num\_instances)\*\*2

uncer -= prob\_of\_matching

return uncer

def calc\_info\_gain(self, lhs\_data, rhs\_data, cur\_uncert):

num\_instances = len(lhs\_data) + len(rhs\_data)

uncert\_lhs = self.calc\_uncertainty(lhs\_data)

uncert\_rhs = self.calc\_uncertainty(rhs\_data)

weight\_lhs = len(lhs\_data)/num\_instances

weight\_rhs = len(rhs\_data)/num\_instances

new\_uncert = weight\_lhs\*uncert\_lhs + weight\_rhs\*uncert\_rhs

return cur\_uncert - new\_uncert

@staticmethod

def calc\_unique\_values(data, attr\_id):

values = [instance[attr\_id] for instance in data]

unique\_values = set(values)

return unique\_values

def calc\_best\_split(self, data):

best\_info\_gain = 0

best\_cond = None

for attr\_id in range(len(data[0]) - 1):

for val in self.calc\_unique\_values(data, attr\_id):

cond = condition(attr\_id, val)

true\_instances, false\_instances = self.split\_data(data, cond)

if max(len(true\_instances), len(false\_instances)) >= len(data):

continue

info\_gain = self.calc\_info\_gain(false\_instances, true\_instances,self.calc\_uncertainty(data))

if info\_gain > best\_info\_gain:

best\_info\_gain = info\_gain

best\_cond = cond

return best\_info\_gain, best\_cond

@staticmethod

def split\_data(data, cond):

true\_data = [row for row in data if cond.answer(row)]

false\_data = [row for row in data if not cond.answer(row)]

return true\_data, false\_data

def create\_tree(self, data):

info\_gain, cond = self.calc\_best\_split(data)

if info\_gain == 0:

return leaf(data)

true\_data, false\_data = self.split\_data(data, cond)

true\_subtree = self.create\_tree(true\_data)

false\_subtree = self.create\_tree(false\_data)

return node(cond, true\_subtree, false\_subtree)

def print\_tree(self):

print("")

print(self.tree.print\_tree())

print("")

def classify(self, instance):

result\_counts = self.tree.classify(instance)

results = []

for class\_name, count in result\_counts.items():

results += [class\_name] \* count

return random.choice(results)

def test\_accuracy(self, test\_data):

num\_correct = 0

for instance in test\_data:

answer = self.classify(instance[0:-1])

if answer == instance[-1]:

num\_correct += 1

return num\_correct/len(test\_data)

def save(self, test\_data):

with open("results.csv", mode="w") as file:

for instance in test\_data:

answer = self.classify(instance[0:-1])

row = ",".join([str(i) for i in instance] + [answer])

file.write(row + "\n")

**driver.py**

import random

from statistics import mean, stdev

from cart import cart\_tree

def recording(path):

# read in and preprocess data

with open(path) as file:

input = file.readlines()

data = []

for entry in input:

readings = entry.rstrip("\n").split(",")

attributes = [float(r) for r in readings[0:-1]]

attributes.append(readings[-1])

# add back in the classes

data.append(attributes)

return data

def main():

path\_to = input('Введіть шлях до вибірки : ')

data = recording(path\_to)

# build and print CART decision tree

cart\_tree(data).print\_tree()

# test accuracy 10 times and display the results

scores = []

for i in range(0,10):

random.shuffle(data)

train\_n = len(data) \* 2//3

training\_data = data[0:train\_n]

test\_data = data[train\_n:]

cart = cart\_tree(training\_data)

accuracy = cart.test\_accuracy(test\_data)

print("Ітерація - {}, Точність - {:.4f}".format(i+1, accuracy))

scores.append(accuracy)

cart.save(test\_data)

print("")

print("Класифікатор CART має точність {:.2f}%, +/- {:.2f}%"

.format(mean(scores) \* 100, stdev(scores) \* 2 \* 100))

if \_\_name\_\_ == '\_\_main\_\_': main()

**Висновки**

1. У ході роботи було досліджено актуальність даної теми та переваги обраного методу.

2. Була проаналізована та застосована інформація що до реалізації алгоритму CART.

3. Було реалізовано класифікатор на основі дерева розв’язків CART.

4. Проведено навчання та тестування системи на медичних даних про стан пацієнтів та їх захворювання.

5. На основі тестів були зроблені висновки що до ефективності системи та її переваг.

6. Було вирішено поставлене практичне завдання.

**Список використаниї джерел**

* + 1. Чубукова И. А. Data Mining. М.: Бином, 2008
  1. Breiman L., Friedman J. H., Olshen R. A., & Stone C. J. Classification and regression trees. Monterey, CA: Wadsworth & Brooks/Cole Advanced Books & Software, 1984
  2. Толстова Ю.Н. Анализ социологических данных. М.: Научный мир, 2000
  3. Деревья решений – CART математический аппарат. Часть №1 // h ttp://www.basegroup.ru/trees/math\_cart\_part1.htm
  4. Электронный учебник «Statistica» // http://www.statsoft.ru/home/textbook.htm (недоступная ссылка)
  5. Деревья решений - CART математический аппарат. https://docplayer.ru/

**Додаток**

*Таблиця 1. Вхідні данні*

|  |  |
| --- | --- |
| 38.77 , 88 , 1 , 1 , 0 , flu | 38.65 , 84 , 0 , 0 , 1 , smallpox |
| 39.99 , 98 , 1 , 1 , 0 , flu | 39.52 , 59 , 0 , 0 , 1 , smallpox |
| 37.7 , 75 , 1 , 1 , 0 , flu | 38.06 , 82 , 1 , 0 , 1 , smallpox |
| 40.65 , 62 , 0 , 1 , 0 , flu | 39.16 , 83 , 0 , 1 , 1 , smallpox |
| 36.85 , 93 , 0 , 1 , 0 , flu | 38.21 , 66 , 1 , 0 , 1 , smallpox |
| 36.96 , 74 , 1 , 1 , 0 , flu | 40.37 , 71 , 0 , 0 , 1 , smallpox |
| 42.25 , 61 , 1 , 1 , 0 , flu | 38.53 , 89 , 0 , 0 , 1 , smallpox |
| 37.89 , 81 , 1 , 1 , 0 , flu | 37.63 , 82 , 0 , 0 , 1 , smallpox |
| 41.26 , 75 , 1 , 1 , 0 , flu | 37.12 , 70 , 1 , 1 , 1 , smallpox |
| 38.97 , 90 , 1 , 1 , 0 , flu | 37.31 , 62 , 0 , 0 , 1 , smallpox |
| 41.41 , 96 , 0 , 1 , 0 , flu | 39.87 , 57 , 0 , 0 , 1 , smallpox |
| 38.32 , 96 , 1 , 1 , 0 , flu | 38.19 , 87 , 1 , 0 , 1 , smallpox |
| 39.2 , 62 , 1 , 1 , 0 , flu | 38.14 , 92 , 0 , 0 , 1 , smallpox |
| 42.01 , 102 , 1 , 1 , 0 , flu | 37.45 , 70 , 1, 0 , 1 , smallpox |
| 39.6 , 108 , 1 , 1 , 0 , flu | 37.16 , 79 , 0 , 0 , 1 , smallpox |
| 38.12 , 73 , 0 , 1 , 0 , flu | 38.86 , 77 , 0 , 0 , 1 , smallpox |
| 41.35 , 83 , 1 , 1 , 0 , flu | 38.09 , 65 , 0 , 0 , 1 , smallpox |
| 37.18 , 73 , 1 , 1 , 0 , flu | 37.81 , 56 , 0 , 0 , 1 , smallpox |
| 40.89 , 58 , 0 , 1 , 0 , flu | 38.9 , 56 , 0 , 0 , 1 , smallpox |
| 40.35 , 88 , 0 , 1 , 0 , flu | 39.06 , 107 , 0 , 0 , 1 , smallpox |
| 39.2 , 93 , 1 , 1 , 0 , flu | 37.99 , 95 , 0 , 0 , 1 , smallpox |
| 37.74 , 73 , 1 , 1 , 0 , flu | 39.18 , 71 , 0 , 0 , 1 , smallpox |
| 39.28 , 82 , 1 , 1 , 0 , flu | 37.88 , 86 , 0 , 0 , 1 , smallpox |
| 36.41 , 68 , 1 , 1 , 0 , flu | 37.21 , 83 , 0 , 1 , 1 , smallpox |
| 37.58 , 57 , 1 , 1 , 0 , flu | 38.23 , 71 , 0 , 0 , 1 , smallpox |
| 39.79 , 75 , 0 , 1 , 0 , flu | 38.17 , 108 , 1 , 0 , 1 , smallpox |
| 39.2 , 101 , 1 , 1 , 0 , flu | 38.1 , 91 , 0 , 0 , 1 , smallpox |
| 38.04 , 106 , 1 , 1 , 0 , flu | 38.65 , 81 , 0 , 0 , 1 , smallpox |
| 39.54 , 77 , 1 , 1 , 0 , flu | 38.68 , 98 , 0 , 0 , 1 , smallpox |
| 39.7 , 73 , 1 , 1 , 0 , flu | 37.76 , 60 , 0 , 0 , 1 , smallpox |
| 36.69 , 130 , 1 , 0 , 0 , sunstroke | 36.36 , 70 , 0 , 0 , 0 , healthy |
| 36.64 , 143 , 0 , 0 , 0 , sunstroke | 36.73 , 59 , 0 , 0 , 0 , healthy |
| 36.98 , 144 , 1 , 0 , 0 , sunstroke | 36.6 , 62 , 0 , 0 , 0 , healthy |
| 36.47 , 122 , 1 , 0 , 0 , sunstroke | 36.68 , 75 , 0 , 0 , 0 , healthy |
| 36.2 , 131 , 1 , 0 , 0 , sunstroke | 36.65 , 81 , 0 , 0 , 0 , healthy |
| 36.4 , 139 , 1 , 0 , 0 , sunstroke | 36.77 , 103 , 0 , 0 , 0 , healthy |
| 36.42 , 134 , 0 , 0 , 0 , sunstroke | 36.7 , 68 , 0 , 0 , 0 , healthy |
| 36.37 , 152 , 1 , 0 , 0 , sunstroke | 36.71 , 101 , 0 , 0 , 0 , healthy |
| 36.66 , 136 , 1 , 0 , 0 , sunstroke | 36.11 , 66 , 0 , 0 , 0 , healthy |
| 37.0 , 128 , 1 , 0 , 0 , sunstroke | 36.36 , 86 , 0 , 0 , 0 , healthy |
| 36.77 , 159 , 1 , 0 , 0 , sunstroke | 36.56 , 72 , 0 , 0 , 0 , healthy |
| 36.43 , 112 , 1 , 0 , 0 , sunstroke | 36.68 , 65 , 0 , 0 , 0 , healthy |
| 37.48 , 129 , 1 , 0 , 0 , sunstroke | 36.32 , 96 , 0 , 0 , 0 , healthy |
| 36.34 , 140 , 1 , 1, 0 , sunstroke | 36.49 , 71 , 0 , 0 , 0 , healthy |
| 36.94 , 114 , 1 , 0 , 0 , sunstroke | 36.53 , 102 , 0 , 0 , 0 , healthy |
| 36.88 , 113 , 1 , 0 , 0 , sunstroke | 36.67 , 94 , 0 , 0 , 0 , healthy |
| 36.87 , 128 , 1 , 0 , 0 , sunstroke | 36.49 , 71 , 0 , 0 , 0 , healthy |
| 36.86 , 126 , 0 , 0 , 0 , sunstroke | 36.67 , 67 , 0 , 0 , 0 , healthy |
| 36.25 , 102 , 1 , 0 , 0 , sunstroke | 36.36 , 106 , 0 , 0 , 0 , healthy |
| 36.71 , 113 , 1 , 0 , 0 , sunstroke | 36.74 , 77 , 0 , 0 , 0 , healthy |
| 36.65 , 127 , 0, 1 , 1 , sunstroke | 36.54 , 59 , 0 , 0 , 0 , healthy |
| 36.86 , 130 , 1 , 0 , 0 , sunstroke | 36.62 , 95 , 0 , 0 , 0 , healthy |
| 36.54 , 123 , 1 , 0 , 0 , sunstroke | 36.34 , 63 , 0 , 0 , 0 , healthy |
| 36.53 , 129 , 0 , 0 , 0 , sunstroke | 36.5 , 66 , 0 , 0 , 0 , healthy |
| 36.27 , 143 , 1 , 0 , 0 , sunstroke | 36.62 , 85 , 0 , 0 , 0 , healthy |
| 36.32 , 125 , 1 , 0 , 0 , sunstroke | 36.76 , 81 , 0 , 0 , 0 , healthy |
| 36.68 , 121 , 1 , 0 , 0 , sunstroke | 36.46 , 74 , 0 , 0 , 0 , healthy |
| 36.31 , 142 , 1 , 0 , 0 , sunstroke | 36.85 , 84 , 0 , 0 , 0 , healthy |
| 36.7 , 100 , 1 , 0 , 0 , sunstroke | 36.66 , 98 , 0 , 0 , 0 , healthy |
| 36.66 , 107 , 1 , 0 , 0 , sunstroke | 36.59 , 86 , 0 , 0 , 0 , healthy |

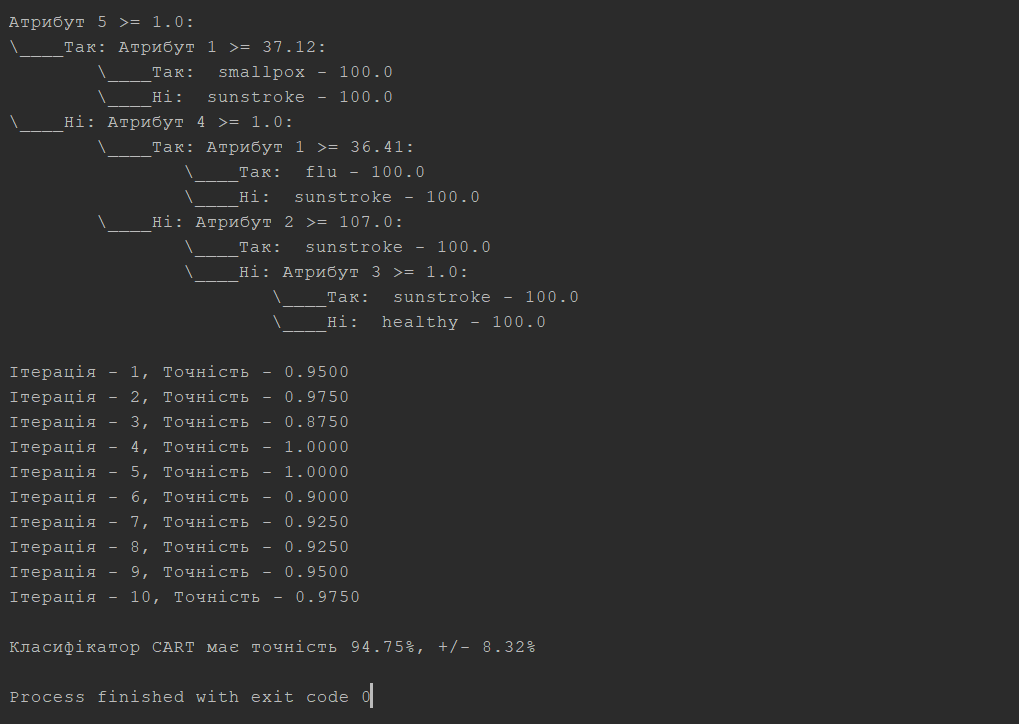


Рис. 1. Скріншот виконання програми.

*Таблиця 2. Вихідні данні*

|  |  |
| --- | --- |
| 37.45,70.0,1.0,0.0,1.0, smallpox, smallpox | 38.19,87.0,1.0,0.0,1.0, smallpox, smallpox |
| 36.11,66.0,0.0,0.0,0.0, healthy, healthy | 40.89,58.0,0.0,1.0,0.0, flu, flu |
| 37.12,70.0,1.0,1.0,1.0, smallpox, sunstroke | 36.85,84.0,0.0,0.0,0.0, healthy, healthy |
| 36.32,125.0,1.0,0.0,0.0,sunstroke,sunstroke | 37.18,73.0,1.0,1.0,0.0, flu, flu |
| 38.04,106.0,1.0,1.0,0.0, flu, flu | 36.62,95.0,0.0,0.0,0.0, healthy, healthy |
| 39.2,101.0,1.0,1.0,0.0, flu, flu | 36.6,62.0,0.0,0.0,0.0, healthy, healthy |
| 36.25,102.0,1.0,0.0,0.0,sunstroke,sunstroke | 36.69,130.0,1.0,0.0,0.0,sunstroke,sunstroe |
| 37.99,95.0,0.0,0.0,1.0, smallpox, smallpox | 37.48,129.0,1.0,0.0,0.0,sunstroke,sunstrokе |
| 38.1,91.0,0.0,0.0,1.0, smallpox, smallpox | 38.65,84.0,0.0,0.0,1.0, smallpox, smallpox |
| 37.76,60.0,0.0,0.0,1.0, smallpox, smallpox | 36.2,131.0,1.0,0.0,0.0, sunstroke, sunstroke |
| 41.35,83.0,1.0,1.0,0.0, flu, flu | 36.71,113.0,1.0,0.0,0.0,sunstroke,sunstroke |
| 41.41,96.0,0.0,1.0,0.0, flu, flu | 36.62,85.0,0.0,0.0,0.0, healthy, healthy |
| 38.65,81.0,0.0,0.0,1.0, smallpox, smallpox | 39.2,93.0,1.0,1.0,0.0, flu, flu |
| 37.89,81.0,1.0,1.0,0.0, flu, flu | 38.14,92.0,0.0,0.0,1.0, smallpox, smallpox |
| 37.63,82.0,0.0,0.0,1.0, smallpox, smallpox | 36.53,129.0,0.0,0.0,0.0, sunstroke, sunstroke |
| 41.26,75.0,1.0,1.0,0.0, flu, flu | 36.54,59.0,0.0,0.0,0.0, healthy, healthy |
| 38.12,73.0,0.0,1.0,0.0, flu, flu | 39.99,98.0,1.0,1.0,0.0, flu, flu |
| 36.59,86.0,0.0,0.0,0.0, healthy, healthy | 37.31,62.0,0.0,0.0,1.0, smallpox, smallpox |
| 36.64,143.0,0.0,0.0,0.0,sunstroke,sunstroke | 36.47,122.0,1.0,0.0,0.0, sunstroke, sunstroke |
| 42.25,61.0,1.0,1.0,0.0, flu, flu | 36.4,139.0,1.0,0.0,0.0, sunstroke, sunstroke |