Міністерство освіти і науки України

Національний технічний університет України

“Київський політехнічний інститут ім. Ігоря Сікорського”

Кафедра автоматизації проектування енергетичних процесів і систем

Звіт

з лабораторної роботи №4

«Дерево рішень. Банківська справа. Оцінка кредитоспроможності клієнтів банку при видачі кредитів.»

Виконав:

студент 5-го курсу

ТЕФ

гр. ТМ-01мп  
Каліка Б.М.

Перевірив:

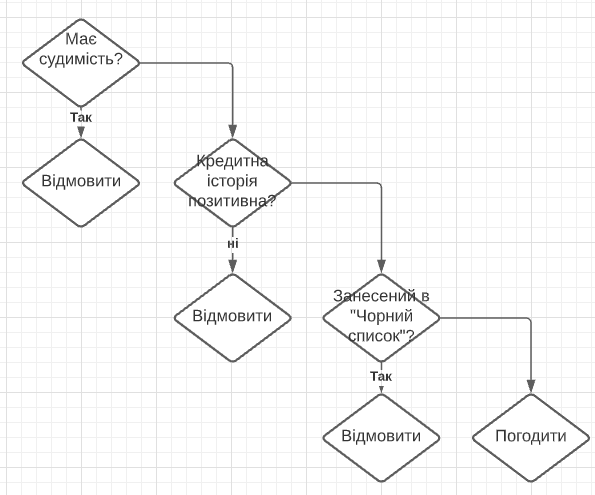
Проф. Федорова Н.В.

Київ – 2020

Для кожного атрибута набору даних алгоритм дерева рішень формує вузол, де найважливіший атрибут розміщується в кореневому вузлі. Для оцінки ми починаємо з кореневого вузла і рухаємось вниз по дереву, слідуючи відповідному вузлу, який відповідає нашій умові або "рішенню". Цей процес триває до досягнення листового вузла, який містить прогноз або результат дерева рішень.

Розглянемо сценарій, коли людина просить кредит, і ми повинні прийняти рішення, погодити їм чи ні. Існує кілька факторів, які допомагають визначити наше рішення, деякі з них перераховані нижче:

Дерево рішень для вищезазначеного сценарію виглядає так:

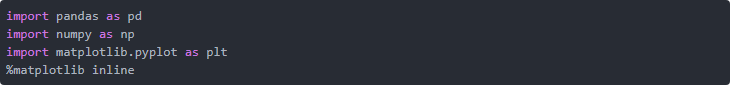


**Застосування дерев рішень за допомогою Python**

Ми реалізуємо алгоритм дерева рішень, використовуючи бібліотеку Scikit-Learn Python.

**Імпорт бібліотек**

Наступний сценарій імпорту необхідних бібліотек:



**Імпортування набору даних**

Оскільки наш файл у форматі CSV, ми будемо використовувати метод read\_csv panda для читання нашого файлу даних CSV. Для цього виконайте такий сценарій:



**Аналіз даних**

Використаємо наступну команду, щоб побачити кількість рядків і стовпців у нашому наборі даних:

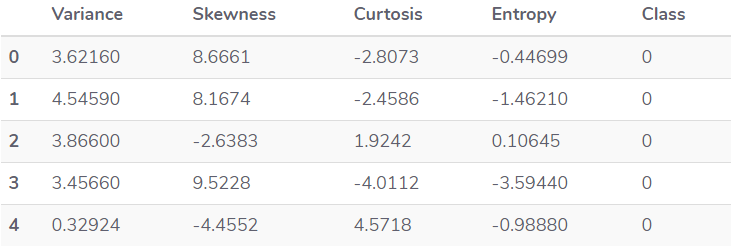


На виході показано "(1372,5)", що означає, що наш набір даних має 1372 записи та 5 атрибутів.

Виконаємо команду, щоб перевірити перші п’ять записів набору даних:



Результат буде виглядати так:



**Підготовка даних**

Ми розділимо наші дані на атрибути та мітки, а потім поділимо отримані дані на навчальні та тестові набори. Роблячи це, ми можемо навчити наш алгоритм на одному наборі даних, а потім перевірити його на зовсім іншому наборі даних, яких алгоритм ще не бачив. Це надає вам більш точне уявлення про те, як насправді буде працювати ваш навчений алгоритм.

Щоб розділити дані на атрибути та мітки, виконаємо такий код:



Тут змінна X містить усі стовпці з набору даних, крім стовпця "Клас", який є міткою. Змінна y містить значення зі стовпця "Клас". Змінна X - це наш набір атрибутів, а змінна y містить відповідні мітки.

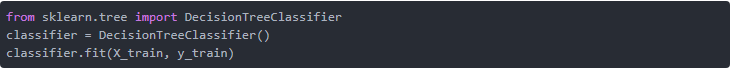
Останнім кроком попередньої обробки є розподіл наших даних на навчальні та тестові набори. Бібліотека model\_selection Scikit-Learn містить метод train\_test\_split, який ми використовуватимемо для випадкового розподілу даних на навчальні та тестові набори. Для цього викона наступний код:



У наведеному вище коді параметр test\_size визначає співвідношення тестового набору, який ми використовуємо для розподілу 20% даних до тестового набору та 80% для навчання.

**Навчання та прогнозування**

Після того, як дані були розділені на навчальний і тестувальний набори, завершальним етапом є тренування алгоритму дерева рішень на цих даних та складання прогнозів. Scikit-Learn містить бібліотеку дерев, яка містить вбудовані класи / методи для різних алгоритмів дерева рішень. Оскільки ми збираємось виконати тут завдання класифікації, для цього прикладу ми використаємо клас DecisionTreeClassifier. Метод придатності цього класу викликається для навчання алгоритму на навчальних даних, який передається як параметр методу придатності. Для навчання алгоритму виконаємо наступний сценарій:

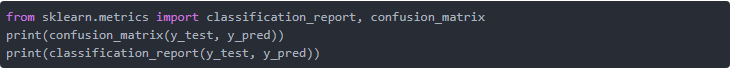


Тепер, коли наш класифікатор пройшов підготовку, зробимо прогнози щодо даних тесту. Для прогнозування використовується метод передбачення класу DecisionTreeClassifier. Напишемо наступний код для використання:

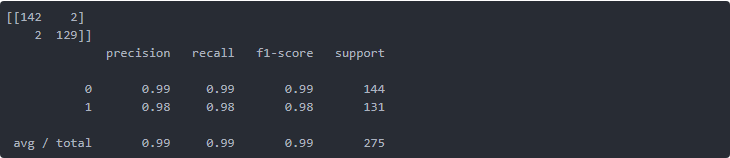


**Оцінка алгоритму**

На даний момент ми навчили наш алгоритм і зробили кілька прогнозів. Зараз ми побачимо, наскільки точним є наш алгоритм. Для класифікаційних завдань деякими часто використовуваними метриками є матриця змішання, точність, відкликання та оцінка F1. На щастя для нас Scikit=-Learn's metrics містить методи classification\_report та confusion\_matrix, які можуть бути використані для обчислення цих метрик для нас:



Це дасть таку оцінку:



З confusion\_matrix видно, що з 275 тестових екземплярів наш алгоритм неправильно класифікував лише 4. Це 98,5% точності. Не погано!