Dokumentacja wstepna projektu z UMA

Piotr Jabłoński (325163) i Paweł Wysocki (325248)

Grudzień 2024

Contents

1 Temat projektu

Celem naszego projektu jest implementacja algorytmu konstruującego drzewo klasyfikujące z wyborem testu przy pomocy ruletki.

2 Opis problemu

2.1 Drzewo klasyfikacyjne

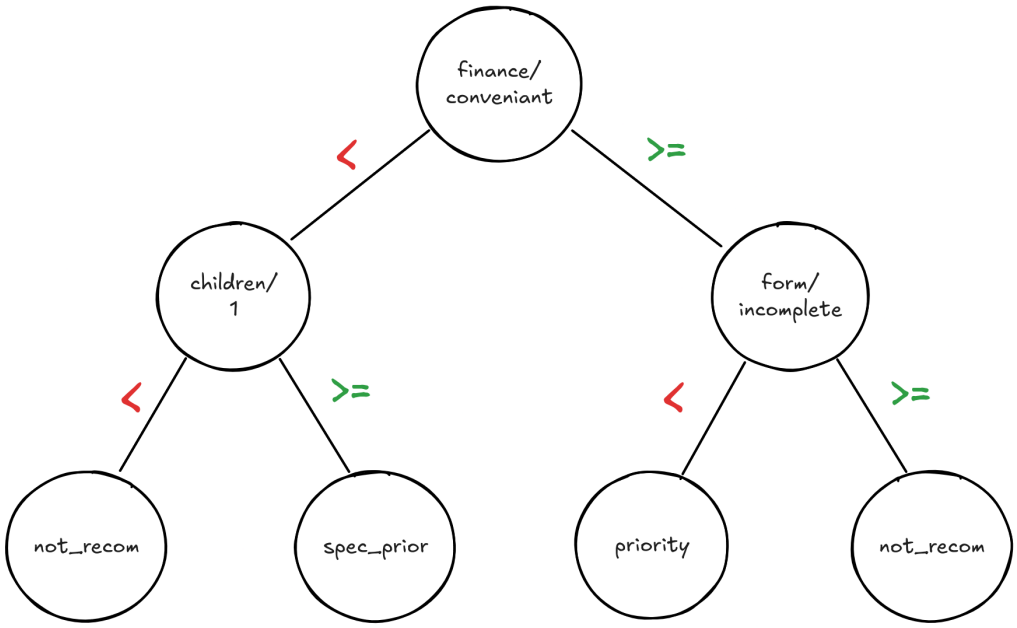
Drzewo klasyfikacyjne to relatywnie prosty model uczenia maszynowego. Polega on na konstrukcji drzewa binarnego gdzie:

- węzeł - atrybut, na podstawie którego dzielimy klasy na podzbiory. Tak zwany "test"
- liść - klasa lub predykcja klasy

Jest to w pewnym sensie jedna wielka "if-else" instrukcja, z tą różnicą że wybór testu dla dodawanego węzłu odbywa się automatycznie. Najlepszy atrybut wybieramy na podstawie największego **zysku informacji** (Information Gain) dla tego atrybutu.

W tym zadaniu izolujemy dane do momentu, gdzie wszystkie dane należą do tej samej klasy (dane jednorodne). Im wybierzemy lepszy test do podziału danych, tym precyzyjniej i ogólniej nasze drzewo się będzie zachowywało. Naszym celem jest więc znalezienie takiego testu, aby maksymalnie zmiejszyć entropię całego zbioru.

Przykładowe drzewo dla zbioru danych "Nursery"



2.2 Ruletka

Zwykle drzewa decyzyjne są zachłanne, tzn. wybierają ten test, który ma największą jakość. Takie podejście jest proste w realizacji oraz bardzo efektywne, natomiast takie drzewo jest bardzo łatwo przeuczyć. W naszym przypadku selekcja testu odbędzie się ruletkowo:

$$P(a_i) = \frac{IG(a_i)}{\sum_i^n IG(a_j)}$$

gdzie

- $P(a_i)$ - prawdopodobieństwo wybrania atrybutu a_i
- $IG(a_i)$ - zysk informacji dla atrybutu a_i
- n - ilość atrybutów

Takie podejście sprawia, że prawdopodobieństwo wybrania testu dla atrybutu a_i jest wprost-proporcjonalne do zysku informacji, dzięki czemu drzewa powinny mieć miejszą podatność na przeuczenie.

2.3 Algorytmy

Żeby skonstruować drzewo klasyfikacyjne potrzebujemy 3 algorytmów:

- Algorytm budowania drzewa
- Algorytm wyboru testu z ruletką
- Algorytm obliczania zysku informacji (**IG**)

Opracowaliśmy pseudokod w języku Python-podobnym, żeby lepiej zwizualizować nasz tok myślenia:

2.3.1 Algorytm budowania drzewa

```
def build_tree(attrs, data, classes, max_depth) -> DecisionTree:
    if max_depth == 0 or len(attrs) == 0:
        return most_common(attrs)

    tree = DecisionTree()

    # przeprowadzamy test z ruletką
    tree.attr, tree.threshold = test(attrs, data, classes)
    new_attr = attrs - tree.attr

    # dzielimy dane na postawie testu
    left_data, right_data = [...]
    tree.left = build_tree(new_attr, left_data, classes, max_depth - 1)
    tree.right = build_tree(new_attr, right_data, classes, max_depth - 1)

    return tree
```

2.3.2 Algorytm wyboru testu z ruletką

```
def test(attrs, data, classes):
    IGs = []
    for a in attrs:
        for c in classes:
            IQs.append(IQ(a, data, classes, threshold=c))

    # ruletkowy wybór zysku informacji
    total = sum(IQs); running_total = 0; p = randint(0, total)
    for iq in IQs:
        running_total += iq
        if running_total >= p:
            return iq
```

2.3.3 Algorytm obliczania zysku informacji (IG)

Informacja to tak naprawdę różnica entropi węzła nadrzędnego i średniej ważonej entropii węzła potomnego. Im większy zysk informacji, tym bardziej zmiejszyliśmy entropię w danych - tym dane stają się czystrze.

$$IQ(S, a) = H(S) - \sum_{v \in vals(a)} \frac{|Sv|}{|S|} \cdot H(Sv)$$

gdzie

- S - podzbiór danych
- Sv - podzbiór danych po dzieleniu przez atrybut a
- entropia $H(S) = \sum_{i=1} c - p_i \cdot \log_2 p_i$

3 Plan eksperymentów

Aby przeprować odpowienie testy statystyczne postanowiliśmy przeprowadzić eksperymenty na wielu różnych zbiorach danych oraz porównać uzyskane wyniki do klasyfikatora DecisionTreeClassifer z pakietu naukowego scikit-learn.

Macież błędów (tablica pomyłek) posłuży nam do zwizualizowania i zweryfikowania skuteczności klasyfikacji. Będziemy skupiać się na miarach: **PPV**, **Recall** i **F1**.

4 Zbiory danych

Przygotowaliśmy 4 zbiory danych, na których będziemy prowadzić eksperymenty.

4.1 Red Wine Quality

Zawiera 11 fizykochemicznych atrybutów win:

1. Kwasowość stała
2. Kwasowość wulkaniczna
3. Kwas cytrynowy
4. Cukier pozostały
5. Chloridy
6. Dwutlenek siarki wolny
7. Dwutlenek siarki całkowity
8. Gęstość
9. pH
10. Siarczanaty
11. Procent alkoholu

Zadanie klasyfikacji:

- Jakość wina w skali całkowitoliczbowej (1-10)

4.2 Loan Approval Classification

Zawiera 9 atrybutów o osobie składającej wniosek o pożyczkę oraz 4 atrybuty o samej pożyczce - łącznie 13 atrybutów, na podstawie których należy zklasyfikować stan wniosku (zaakceptowany bądź odrzucony). Atrybuty:

1. Wiek
2. Płeć
3. Edukacja
4. Dochód roczny
5. Ilość lat doświadczenia zawodowego
6. Stan posiadania domu (wynajem, na własność, hipoteka)
7. Kwota pożyczki
8. Cel pożyczki
9. Oprocentowanie pożyczki

10. Wysokość wypożyczenia w relacji do dochodu rocznego (%)
11. Zdolność kredytowa
12. Długość historii kredytowej w latach
13. Indikator wcześniejszych niespłaconych wypożyczeń

Zadanie klasyfikacji:

- Akceptacja wniosku o pożyczkę (prawda/fałsz)

4.3 Nursery

Zawiera 8 atrybutów dotyczących rodziny:

1. Zawód rodziców
2. Przedszkole dziecka
3. Struktura rodziny
4. Ilość dzieci
5. Warunki zamieszkania
6. Finansowa sytuacja
7. Społeczna sytuacja
8. Zdrowotna sytuacja

Zadanie klasyfikacji:

- Ocena aplikacji do przedszkola (ocena stanu zdrowia rodziny)

4.4 Mobile Device Usage and User Behavior

1. Id użytkownika
2. Model urządzenia
3. System operacyjny
4. Czas używania aplikacji
5. SOT (Screen On Time)
6. Codzienne zużycie baterii (mAh)
7. Liczba zainstalowanych aplikacji
8. Codzienne zużycie danych
9. Wiek
10. Płeć (M/K)

Zadanie klasyfikacji:

- Ocena zachowania użytkownika (od lekkiego do ekstremalnego użycia w skali całkowitoliczbowej 1-5)