Elementy Inteligencji Obliczeniowej - Zadanie 1
Implementacja drzew decyzyjnych

Dariusz Max Adamski 136674, Sławomir Gilewski 142192

{dariusz.adamski,slawomir.gilewski}@student.put.poznan.pl

Data oddania: 20 października 2020

Wstęp

W tym sprawozdaniu opiszemy naszą implementację algorytmu budowania drzew decyzyjnych ID3 wraz ze wszystkimi pomocniczymi funkcjami. Program i wizualizacje drzew znajdują się w załączonym Jupyter notebooku. W implementacji uwzględniliśmy obsługę ciągłych zmiennych. Pomimo tego, nasza implementacja jest bardzo prosta i z wyłączeniem kodu rysującego drzewa zajmuje tylko 48 linijek kodu.

1 Narzędzia

Do wykonania zadania użyliśmy języka Python i środowisko Jupyter notebook. Użyliśmy też biblioteki Pandas do wczytania danych i Numpy do wektorowych operacji matematycznych. Wizualizację drzewa wykonaliśmy przy pomocy biblioteki Graphviz.

2 Algorytm ID3

Algorytm ID3 zaimplementowaliśmy rekurencyjnie. Funkcja ID3 zwraca drzewo w formie krotki, której pierwszym elementem jest nazwa węzła, a drugim lista dzieci. Każde dziecko to krotka, której pierwszy element to opis wchodzącej krawędzi, a drugi to wartość rekurencyjnego wywołania funkcji ID3. Liść drzewa to krotka, której pierwszy element to etykieta danej (w przypadku Titanica, 0 lub 1), a drugi element to pusta lista.

Funkcja ID3 jako argumenty przyjmuje podzbiór danych, odpowiednio ograniczony na podstawie poprzednich podziałów drzewa na stosie wywołań. Jako argument przekazujemy też listę nazw zmiennych, które chcemy aby algorytm uznawał za ciągłe. Ostatnim argumentem jest flaga „drop_numerical".

Na początku sprawdzamy warunki stopu, a następnie wywołujemy funkcję „best_column", która zwraca nazwę najlepszej zmiennej do podziału drzewa, na podstawie podzbioru danych w aktualnym węźle drzewa.

Następnie w pętli dzielimy drzewo tworząc węzły dla każdej wartości zmiennej kategorycznej, lub dwa węzły dla wybranej wartości zmiennej ciągłej, gdzie pierwszy węzeł zawiera podzbiór danych, w których wartość zmiennej jest mniejsza od wybranej wartości, a drugi jest jego dopełnieniem.

Dla każdego węzła, ze zbioru danych jest tworzony podzbiór, w którym dla każdej danej, wartość wybranej zmiennej do podziału jest równa wartości lub warunkowi (dla zmiennych ciągłych) na krawędzi pomiędzy węzłami drzewa. Ze zbioru usuwana jest też zmienna na której dzieliliśmy

drzewo. Wyjątkiem od tego jest sytuacja, kiedy flaga „drop_numerical" przyjmuje wartość fałszywą i dzielimy drzewo na zmiennej ciągłej - wtedy nie usuwamy tej zmiennej ze zbioru, aby móc odtworzyć pełną informację ze zbioru danych w drzewie. Następnie podzbiór danych jest przekazywany do rekurenyjnego wywołania funkcji ID3, która zwraca drzewo którego korzeń dołączamy do aktualnego drzewa.

Budowę drzewa kończymy, zwracając liść z wartością etykiety pozostałych danych. Warunek zakończenia jest spełniony gdy dla każdej danej w podzbiorze danych etykieta jest ta sama, lub gdy wykorzystaliśmy już wszystkie zmienne przy podziałach w drzewie.

3 Wybór zmiennej do podziału

Funkcja „best_column" zwraca nazwę zmiennej, która ma największy „GainRatio". Dla zmiennych ciągłych wybierana jest wartość k, dla której po zamienieniu wartości zmiennej na wartości kategoryczne („$< k$" i „$\geq k$"), „GainRatio" jest największe. Jeśli taki podział na zmiennej ciągłej jest najlepszy ze wszyskich zmiennych w zbiorze, to zwracany jest też tak zdyskretyzowany zbiór.

Kandydatów na wartość k zwraca funkcja „splits", która sortuje dane rosnąco według wartości wybranej zmiennej a następnie zwraca listę wartości które są średnią poprzednika i następnika wartości zmiennej, gdzie następuje zmiana etykiet danych.

Implementacja „GainRatio", entropii warunkowej i entropii jest identyczna do podanych wzorów matematycznych. Jedyna modyfikacja, to zapewnienie, że w „GainRatio" nie dzielimy przez 0 (mimo, że w naszej implementacji algorytmu ID3 taka sytuacja nie wystąpi, przez warunki stopu).