Wstęp do sztucznej inteligencji
Sprawozdanie 4

Jan Kwiatkowski 299374

Kwiecień 2023

1 Treść zadania

Tematem czwartego zadania jest regresja i klasyfikacja. W ramach tego zadania będą musieli Państwo zaimplementować algorytm lasu losowego i przeprowadzić klasyfikację dla zadanego zbioru danych. W raporcie poza standardową analizą algorytmu i badanego zbioru danych należałoby:

1. Sprawdzić na wykresie (histogram) czy można na podstawie parametrów obserwacji stwierdzić istnienie poszukiwanych grup klas (może występują jakieś ciekawe zależności, jeżeli chodzi o częstotliwość występowania obserwacji w jakiejś kombinacji). Sugeruję zawrzeć w raporcie tylko te ciekawsze wykresy (2-3).

2. Sprawdzić jak podział danych na zbiór trenujący i testowy wpływa na sprawność modelu (poeksperymentować z proporcjami). Czy widoczne są oznaki niedouczenia / przeuczenia modelu? (jeżeli ktoś będzie chciał robić to iteracyjnie nie ma sensu sprawdzić co 0,5-1%, proszę potestować co 5-10% zmian, aby były widoczne jakieś różnice).

3. Porównać efektywność modelu, kiedy zbiór trenujący jest wstępnie posortowany z przypadkiem, gdy dane zostaną specjalnie pomieszane

4. Wyznaczyć macierz błędów (TP,FP,TN,FN) wraz z takimi miarami jak precyzja, czułość, dokładność modelu (sugerowałbym policzyć to ręcznie, ale pozwalam użyć do tego dedykowanych bibliotek) - proszę zamieścić tę macierz w raporcie.

5. Zamieścić wygląd przykładowego drzewa utworzonego przy pomocy Państwa algorytmu.

2 Problem

Zadanie klasyfikacji oceny zakupu samochodu na podstawie informacji o jego specyfikacji. Zbiór tworzy 1728 obserwacji – 4 klasy z czego 1 jest dominująca (1210 obserwacji).

$$https://archive.ics.uci.edu/ml/datasets/Car+Evaluation$$

3 Dane

3.1 Zbiór

Zbiór ten zawiera 1728 przykładów. Wstępnie nie jest on podzielony na zbiór testowy i treningowy.

Liczba przykładów	Atryb. dyskret.	Atryb. rzecz.	Brak. atryb.	Liczba klas
1728	6	0	0	4

Zbiór posiada nierówno reprezentowane klasy. Poniżej zamieszczono dystrybucję klas w zbiorze:

```
 class      N          N[%]
------------------------------
unacc     1210     (70.023 %)
acc        384     (22.222 %)
good        69     ( 3.993 %)
v-good      65     ( 3.762 %)
```

Klasy: unacc i acc (nieakceptowalny i akceptowalny) to aż ponad 92% przykładów, a klasy good i v-good (dobry i bardzo dobry) to prawie 8%.

3.2 Przygotowanie danych

Zaimplementowano możliwość losowego przetasowania danych aby zniwelować ich uporządkowanie i w ten sposób zapobiec wpływowi porządku danych na wynik uczenia. Wymaga to ustawienia parametru 'is_data_shuffle' na wartość True w momencie wywolywania funkcji 'preprocess_dataset'.

Kolumnę z klasami przepisano do oddzielnej ramki danych. Dla zmiennych kategorycznych ['buying', 'maint', 'lug_boot', 'safety'] zastosowano metodę 'one hot encoding' w celu zlikwidowania problemu naturalnego porządku dla tych atrybutów. Wartości klas zostały zamienione na wartości numeryczne.

4 Algorytm budowania drzewa Id3

Algorytm ID3 (Iteratywny Dychotomizer 3) to rekurencyjny algorytm, który tworzy drzewo decyzyjne od góry do dołu, poprzez podział zbioru danych na mniejsze podzbiory w każdym węźle.

Celem algorytmu ID3 jest wybór atrybutów, które najlepiej dzielą zbiór danych na podzbiory, mierząc jakość podziału przez wzrost informacji (Information Gain). W każdym kroku algorytm wybiera atrybut, który maksymalizuje wartość informacyjną, czyli różnicę entropii przed i po podziale. Zbiór danych jest dzielony na dwa podzbiory na podstawie wartości wybranego atrybutu, a następnie algorytm rekurencyjnie wywołuje się na każdym z podzbiorów, aż do momentu, gdy w każdym liściu drzewa znajdzie się tylko jedna klasa lub zostaną spełnione inne kryteria zakończenia budowy drzewa.

4.1 Podział zbioru

Atrybut względem, którego dokonywany jest podział wybierany jest w taki sposób aby podzielić zbiór S na takie podzbiory, że w każdym z nich występują tylko obiekty jednej klasy. Zwykle jednak zmierza się do sytuacji aby w podzbiorach przypisanych do poddrzew częstości występowania obiektów poszczególnych klas były jak najbardziej zróżnicowane. Miarą tego zróżnicowania jest entropia.

4.1.1 Entropia zbioru

Entropia danego zbioru danych jest miarą nieuporządkowania docelowej cechy zbioru danych. Im jest większa tym nieuporządkowanie jest większe, z kolei im mniejsza tym w zbiorze panuje większy porządek (istnieje wiecej jednoznacznych/podobnych wartości w zbiorze). Wartość entropii zdefiniowana jest następująco:

$$E(S) = -\sum_{c \in C} P_c(S) * log_2 P_c(S) \tag{1}$$

gdzie:

$P_c(S)$ - częstość występowania klasy c w zbiorze S (entropia maksymalna gdy P_c są sobie równe dla różnych klas c, entropia minimalna (=0), gdy częstość występowania jednej z klas wynosi 1, tzn. brak obiektów innych klas)

S - zbiór dla którego liczona jest entropia

4.1.2 Entropia zbioru podzielonego na podzbiory przez atrybut d

Entropia zbioru podzielonego na podzbiory przez atrybut d to miara nieporządku w tym podziale. Bierze pod uwagę dwie zmienne losowe i daje odpowiedź jak jedna zmienna losowa pomaga w porządkowaniu wartości drugiej zmiennej losowej. Jest to średnia ważona entropii podzbiorów i wyraża się wzorem:

$$H(D, S) = \sum_{j} \frac{|S_j|}{|S|} * E(S_j) \tag{2}$$

gdzie:

$|S|$ - liczba elementów zbioru S

$|S_j|$ - dla j = 1,2,.. to zbiory powstałe przez podział zbioru S ze względu na wartość atrybutu D

4.1.3 Information Gain (Zysk informacji)

Zysk informacji jest różnicą entropii całego zbioru przed podziałem i entropii po podziale. Jest miarą na podstawie której wybierany jest atrybut dla drzewa w ID3 (atrybut dla którego IG jest największe). Zysk informacji dla danego zbioru S i atrybutu D możemy zdefiniować następująco:

$$IG(D, S) = E(S) - H(D, S) \tag{3}$$

gdzie:

$E(S)$ - funkcja entropii dla rozpatrywanego zbioru/ podzbioru

$H(D, S)$ - funkcja entropii dla podzbioru

4.2 Formalny zapis algorytmu ID3 dla lasu losowego.

Poniżej znajduje się pseudokod algorytmu ID3, który został zaimplementowany w programie:

Algorithm 1 Algorytm ID3

Input: A - zbiór atrybutów, S - zbiór danych, ytarget - klasa
Output: *Node* (główny węzęł zbudowanego drzewa)

function ID3(S, A)
    if S jest pusty **then**
        return stwórz węzeł
    end if
    if S zawiera te same wartości ytarget= v **then**
        return stwórz liść(label=v)
    end if
    if A jest pusty **then**
        return stwórz liść(label = najczęstsza wartość ytarget w S)
    end if
    $Amax \leftarrow argmax(IG(S, A\_to\_split))$
    $Node \leftarrow$ stwórz węzeł($Amax$)
    for v_i in Amax.values **do**
        Dodaj poddrzewo (Node, v_i)
        $Sv_i \leftarrow$ S $\mid value = v_i$
        if Sv_i jest pusty **then**
            stwórz liść(label = najczęstsza wartość ytarget w S)
        else
            return ID3 (Sv_i, $A - \{Amax\}$)
        end if
    end for
    return $Node$
end function

5 Las losowy

Algorytm lasu losowego, jako model bazowy wykorzystuje model drzewa. Ponieważ pojedyńcze drzewo jest bardzo wrażliwe na trenujący zbiór danych (istnieje prawdopodobieństwo, że będzie ono bardzo szczegółowe/przeuczone), tworzy się z nich las to znaczy zbiór drzew decyzyjnych. Zastosowanie lasu losowego ma na celu zwiększenie odporności algorytmu na przeuczenie (zredukowanie wariancji) oraz zwiększanie uogólnienia modelu (nie szukamy jednej szczegółowej hipotezy a tworzymy ich wiele).

Do nauki drzew decyzyjnych można wykorzystać jedną z dwóch metod:

- Bagging
- Boosting

W tworzonym algorytmie wykorzystana została uproszczona metoda bagging.

5.1 Tworzeniu lasu losowego metodą bagging

Podstawą algorytmu jest drzewo decyzyjne, które z założenia powinno być jak najprostsze. Ilość takich prostych klasyfikatorów w systemie może być bardzo duża. Każdy klasyfikator do nauki wykorzystuje pewną próbkę danych (część/część atrybutów) z oryginalnego zbioru.

W metodzie bagging, każdy element (atrybut) ma takie samo prawdopodobieństwo pojawienia się w nowej próbce danych treningowych, w stworzonym algorytmie za każdym razem brane są wszystkie atrybuty do nauki drzewa. Standardowe podejście zmniejsza wariancję poprzez zwiększenie prawdopodobieństwa wylosowania próby, która została niepoprawnie zaklasyfikowana przez poprzedni klasyfikator (uśrednia model i pomaga uniknąć nadmiernego dopasowania).

W stworzonym algorytmie najpierw losowany jest podzbiór o takiej samej wielkości jak zbiór treningowy ale złożony z losowo wybranych krotek zbioru (losowanie ze zwracaniem). Podzbiór ten służy następnie do uczenia modelu. Dzięki tej operacji każde drzewo ma inny, uproszczony zbiór treningowy stworzony z podstawowego zbioru treningowego przekazanego jako parametr do budowy lasu.

5.2 Predykcja lasu losowego

Predykcja lasu losowego polega na wskazaniu przez każde z drzew klasy do której należy obserwowany przykład. Następnie na podstawie wyników wskazywanych przez każdy z klasyfikatorów podawany jest ostateczny wynik, będący wynikiem przewidzianym przez większość drzew.

6 Metryki klasyfikacji

6.1 Macierz pomyłek (ang. confusion matrix)

Macierz pomyłek pokazuje wartości błędnie sklasyfikowane na rzecz innych klas w wygodnej do analizy formie tabelarycznej. Jej rozmiar jest definiowany przez liczbę klas, które przewidujemy. Macierz ta dla badanego zbioru z czterema możliwymi wartościami klasy, na przekątnej obrazuje liczbę poprawnych predykcji, natomiast poza nią błędne klasyfikacje. Rozważana macierz pomyłek prezentuje tabela 1

Tabela 1: Macierz pomyłek

	Przew Klasa 0	Przew Klasa 1	Przew Klasa 2	Przew Klasa 3
Praw Klasa 0	T	F	F	F
Praw Klasa 1	F	T	F	F
Praw Klasa 2	F	F	T	F
Praw Klasa 3	F	F	F	T

Do oceny modeli wykorzystując niżej wymienione metryki należy jeszcze zdefiniować takie pojęcia jak (na przykładzie czy ktoś jest chory):

- TP (true positive) - jest chory i test wykrył, że jest chory,
- TN (true negative) - nie jest chory i test wykrył, że nie jest chory,
- FP (false positive) - jest chory, ale test wykrył, że nie jest chory,
- FN (false negative) - nie jest chory, ale test wykrył, że jest chory.

6.2 Czułość (ang. recall, TP rate, sensitivity)

Czułość jest zdefiniowana jako stosunek poprawnie sklasyfikowanych przykładów (TP) do wszystkich przykładów, które powinny zostać sklasyfikowane jako Positive(TP + FN). Innymi słowy ile ze wszystkich naprawdę pozytywnych przypadków udało nam się znaleźć (tzn. zaklasyfikować jako pozytywne):

Dobre jeśli celem jest wykrywalność a nie interesują nas fałszywe alarmy. Wadą jest to, że ignoruje połowę przypadków (fałszywe), więc model który zawsze będzie zwracał true będzie miał TPr=1, co jest bezwartościowe.

$$Recall = TPrate = \frac{TruePositives}{TruePositives + FalseNegatives} = \frac{TP}{TP + FN} \tag{4}$$

6.3 Precyzja (ang. precision)

Jest zdefiniowana jako stosunek liczby poprawnie sklasyfikowanych przykładów (True Positive - TP) do liczby wszystkich przewidzianych przykładów Positive (ile z przewidzianych pozytywnych przypadków naprawdę jest pozytywnych).

Wadą tej miary będzie przypadek, gdy dla zadania klasyfikacji "czy ktoś jest nieletni"model zaklasyfikuje wszystkich emerytów jako pełnoletnich a resztę obserwacji zignoruje, wtedy Recall będzie mały, ale Precision wysokie.:

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives} = \frac{TP}{TP + FP} \tag{5}$$

6.4 Dokładność (ang. accuracy)

Accuracy czyli dokładność jest to ilość poprawnie przewidzianych w stosunku do wszystkich predykcji (stosunek liczby poprawnych przewidywań do całkowitej liczby przypadków).

Nie sprawdza na przykład dla lotów samolotowych: w USA oznaczamy każdego pasażera jako nie-terrorysta. Przyjmując, że w latach 2000-2017 było średnio 800 mln pasażerów i tylko 17 wykrytych terrorystów to nasz model potrafi ich identyfikować z ACC = 99,99999%.

$$Accuracy = \frac{\text{liczba poprawnych predykcji}}{\text{liczba wszystkich predykcji}} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

6.4.1 Miara F1

Metryka F1 jest średnią harmoniczną precyzji i czułości:

$$F1 = \frac{1}{\frac{\frac{1}{recall} + \frac{1}{precision}}{2}} = \frac{2 \cdot recall \cdot precision}{recall + precision} \tag{7}$$

7 Testy i eksperymenty

Projekt został podzielony na odpowiednie fragmenty, umieszczone w określonych folderach:

- model: zawiera implementacje algorytmu przygotowania danych, drzewa ID3, lasu losowego oraz entropii i 'Information Gain'
- test: zawiera testy poszczególnych algorytmów z folderu model
- experiments: zawiera implementacje przeprowadzonych eksperymentów
- database: tu należy umieść plik ze zbiorem danych o nazwie 'car.data'

7.1 Użyte przy implementacji biblioteki

Do implementacji algorytmu oraz przeprowadzenia odpowiednich testów wykorzystano poniższe biblioteki:

- graphviz: do wizualizacji stworzonego drzewa
- sklearn: do wyznaczania metryk
- numpy
- pandas
- matplotlib: macierz błędów

7.2 Testy

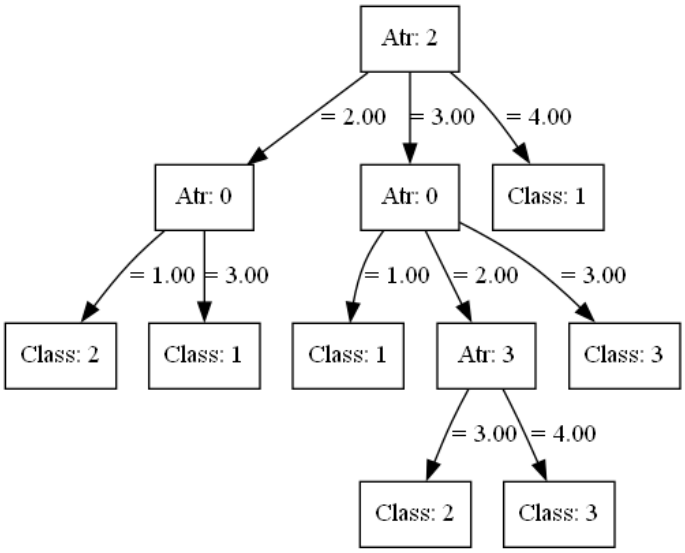
W ramach testów przeprowadzono między innymi:
- sprawdzenie poprawności wyznaczania entropii oraz wartości InformationGain, umieszczone one są w pliku 'tests_entropy_infGain.py',
- sprawdzenie poprawności przygotowania zbioru danych do nauki modelu, umieszczono w pliku 'tests_dataset.py',
- sprawdzono poprawność budowania pojedyńczego drzewa, testy umieszczono w plikach 'tree_draw.py', 'tree_tests.py'.

7.2.1 Zbudowane drzewo - rysunek

Rysunek 1 przedstawia zbudowane drzewo stworzone przez algorytm dla zbioru 2 (plik 'tree_draw.py' z folderu tests):

Tabela 2: Zbiór wykorzystany do tworzenia przykładowego drzewa

atr0	atr1	atr2	atr3	class
1	2	3	4	1
1	1	2	4	2
2	3	4	4	1
1	2	2	4	2
2	2	3	4	3
3	2	2	4	1
1	2	2	3	2
2	2	3	3	2
3	3	3	3	3
1	1	3	3	1



Rys. 1: Drzewo decyzyjne stworzone na podstawie zbioru z tabeli 2

7.3 Eksperymenty

Eksperymenty przeprowadzono dla następujących ustawień:

- liczba drzew w lesie = 25
- maksymalna głębokość drzewa = 100 (czyli dla naszego przypadku brak ograniczenia)
- minimalna liczba atrybutów do podziału = 2

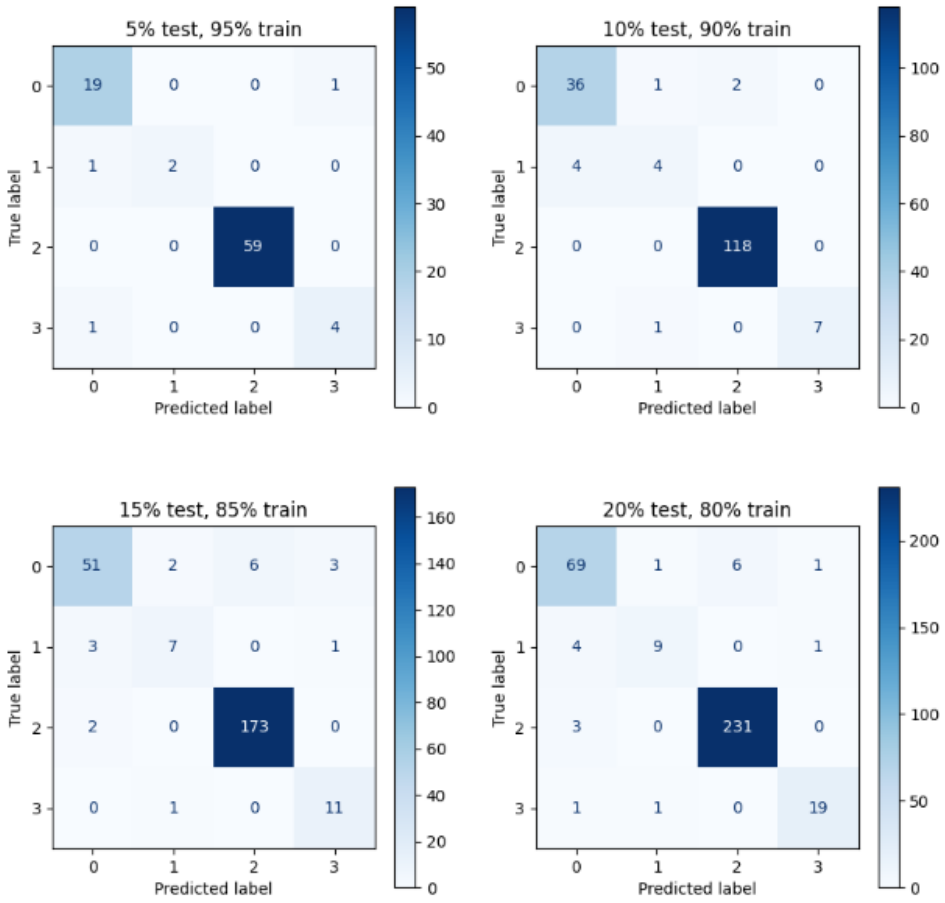
Dla każdego przypadku wyznaczono macierz pomyłek oraz oceniano opisanymi wcześniej metrykami:

- precyzja (precision),
- czułość (recall),
- dokładność (accuracy),
- f1-score.

Ze względu na różną liczność klas zastosowano mikro uśrednianie. Metoda ta sumuje wyniki z poszczególnych klas i wylicza na ich podstawie miarę dla całego zbioru danych.

7.3.1 Wyniki dla różnych wartości proporcji podziału zbioru treningowego i testowego

Macierze pomyłek dla każdego z rozważanych w tym przypadku eksperymentów zostały przedstawiona na rysunku 2.



Rys. 2: Macierze pomyłek dla różnych proporcji podziału zbioru

Widać, że najliczniejszą klasą jest klasa 2 i dla proporcji podziału z przedziału [5%,20%], model z podobną skutecznością dobrze przewiduje klasy.

- Podział 5% testowy, 95% treningowy

Tabela 3: Wyniki:

Accuracy	Micro Precision	Micro Recall	Micro F1
0,98	0,98	0,98	0,98

- Podział 10% testowy, 90% treningowy

Tabela 4: Wyniki:

Accuracy	Micro Precision	Micro Recall	Micro F1
0,96	0,96	0,96	0,96

- Podział 15% testowy, 85% treningowy

Tabela 5: Wyniki:

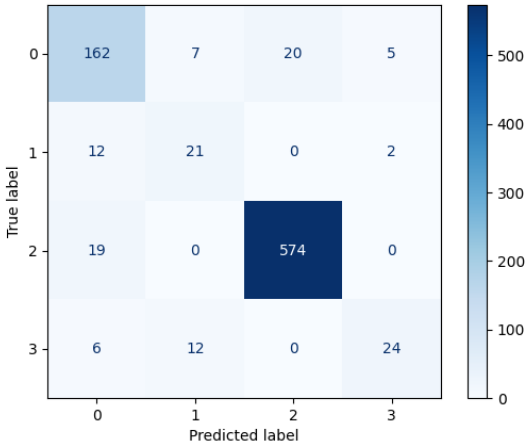
Accuracy	Micro Precision	Micro Recall	Micro F1
0,96	0,96	0,96	0,96

- Podział 20% testowy, 80% treningowy

Tabela 6: Wyniki:

Accuracy	Micro Precision	Micro Recall	Micro F1
0,95	0,95	0,95	0,95

- Podział 50% testowy, 50% treningowy
  Dla takiego podziału widać już znacząco większa liczbę błędnych klasyfikacji (rys 3), co potwierdzają także wartości metryk. Lecz ze względu na przewagę jednej klasy, nie widać spadku skutecznej klasyfikacji dla klasy 1 oraz 3, które są rzadkimi klasami (dla klasy 1 praktycznie połowa przykładów źle klasyfikowana).



Rys. 3: Macierz pomyłek dla podziału zbioru treningowego i testowego w proporcji 50% na 50%

Tabela 7: Wyniki:

Accuracy	Micro Precision	Micro Recall	Micro F1
0,92	0,92	0,92	0,92

7.3.2 Wyniki dla wstępnie posortowanego lub pomieszanego zbioru danych

Eksperyment przeprowadzono dla zbioru testowego 15%, a treningowego 85%.

- Posortowany zbiór:

Tabela 8: Wyniki:

Accuracy	Micro Precision	Micro Recall	Micro F1
0,94	0,94	0,94	0,94

- Pomieszany zbiór:

Tabela 9: Wyniki:

Accuracy	Micro Precision	Micro Recall	Micro F1
0,96	0,96	0,96	0,96

- Wnioski:
  Posortowanie zbioru zwiększyło poprawność klasyfikacji z tego względu, że dane treningowe są bardziej zróżnicowane.