Uczenie maszynowe:
Projekt - dokumentacja końcowa.

Jan Kwiatkowski, Jarosław Nachyła

February 8, 2023

1 Temat projektu

W ramach projektu zaimplementowano las losowy z naiwnym klasyfikatorem bayesowskim (NBC) w zadaniu klasyfikacji. Postępujemy tak jak przy tworzeniu lasu losowego, tylko co któryś (wartość zadana przy inicjalizacji) klasyfikator w lesie to NBC. Klasyfikator NBC pochodzi z istniejącej implementacji z biblioteki scikit-learn.

W ramach projektu została również stworzona dokumentacja wstępna, w której opisano dokładne działanie algorytmów oraz wykorzystywanych do ich działania narzędzi matematycznych. W tamtym dokumencie przy użyciu pseudokodu w uproszczony sposób opisano działanie stworzonego przez nas drzewa decyzyjnego oraz wykorzystywanego klasyfikatora Bayesowskiego NBC.

2 Przygotowanie danych

Przed przystąpieniem do implementacji lasu losowego należało odpowiednio przygotować dane wykorzystywane do trenowania oraz testowania modelu. Względem wstępnych założeń postanowiono nie wykorzystywać do eksperymentów zbioru klasyfikacji grzybów ze względu na jego prostotę. W trakcie testów wynikło, że do predykcji grzybów wystarczy jedno drzewo. Wybrane zbiory podzielono stosując walidację krzyżową z podziałem równym 10 w celu osiągnięcia lepszej estymacji dokładności modeli.

2.1 Zbiór Cars

Pierwszym zbiorem danych którego użyliśmy jest zbiór Car Evaluation. Jest to mały zbiór zawierający 1728 przykładów. Wstępnie nie jest podzielony na zbiory testowy i treningowy, zastosowano więc dla niego walidację krzyżową. Dostępny jest pod linkiem:

https://archive.ics.uci.edu/ml/datasets/car+evaluation

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

Podsumowanie zbioru: Car Evaluation.

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%. Zostanie to uwzględnione przy analizie wyników (obliczaniu metryk per klasa), a także przy podziałach zbiorów na treningowy i testowy(zachowanie proporcji po podziale).

2.2 Zbiór do przewidywania poglądów politycznych

Drugiem zbiorem jest zbiór o nazwie Congressional Voting Records.
Dostępny jest pod linkiem:
https://archive.ics.uci.edu/ml/datasets/Congressional+Voting+Records

Liczba przykładów	Atryb. dyskret.	Atryb. rzecz.	Brak. atryb.	Liczba klas
435	16	0	0	2

Podsumowanie zbioru: Congressional Voting Records.

W porównaniu do pierwszego zbioru "Car Evaluation" powyższy zbiór cechuje się dużą liczbą atrybutów i dwoma klasami o zbliżonej ilości wystąpień.

class	N	N[%]
democrat	267	(61.4 %)
republican	168	(38.6 %)

3 Drzewo decyzyjne Id3

3.1 Implementacja algorytmu

Algorytm tworzący drzewo jest algorytmem rekurencyjnym, który buduje drzewo w kolejności top down od korzenia do liścia metodą „dziel i zwyciężaj".

Jego implementacja znajduje się w pliku o nazwie "decision_tree_id3.py". Zawarty tam algorytm jest rozwinięciem pseudokodu z rysunku "Algorithm 1". Atrybuty tworzące podział w węzłach są wyznaczane za pomocą funkcji zaimplementowanych w pliku "information_gain.py", które na podstawie entropii zbioru obliczają zysk informacji dla każdego atrybutu. Dokładny opis działania tego narzędzia został opisany w dokumentacji wstępnej w rozdziale 2 o nazwie "Algorytm budowania drzewa Id3". Warto nadmienić, że zaimplementowany przez nas algorytm zawiera modyfikację względem orginalnego algorytmu Id3 dla lasu losowego w postaci losowania podzbioru atrybutów biorących udział w podziale drzewa. Losowanie atrybutów do podziału jest sparametryzowane liczbą losowanych atrybutów - *split_features* z możliwymi watościami: None - wszystkie atrybuty, sqrt - pierwiatek l. atrybutów i log2 - logarytm l. atrybutów.

Algorithm 1 Algorytm ID3

Input: A - zbiór atrybutów, S - zbiór danych, ytarget - atrybut przewidywany, split_features- parametr określający losowanie podzbioru atrybutów dla drzewa (las losowy)

Output: $Node$ (główny węzeł zbudowanego drzewa)

1: **function** ID3TREE.FIT(S, A)
2: **if** S jest pusty **then**
3: **return** stwórz węzeł
4: **end if**
5: **if** S zawiera te same wartości ytarget= v **then**
6: **return** stwórz liść(label=v)
7: **end if**
8: **if** A jest pusty **then**
9: **return** stwórz liść(label = najczęstsza wartość ytarget w S)
10: **end if**
11: $A\_to\_split \leftarrow$ losuj_podzbiór(A, split_features)
12: $Amax \leftarrow argmax(IG(S, A\_to\_split))$
13: $Node \leftarrow$ stwórz węzeł($Amax$)
14: **for** v_i in Amax.values **do**
15: Dodaj poddrzewo (Node, v_i)
16: $Sv_i \leftarrow$ S | $value = v_i$
17: **if** Sv_i jest pusty **then**
18: stwórz liść(label = najczęstsza wartość ytarget w S)
19: **else**
20: **return** ID3 (Sv_i, $A - \{Amax\}$)
21: **end if**
22: **end for**
23: **return** $Node$
24: **end function**

3.2 Testy i wyniki dla pojedyńczego drzewa

W trakcie tworzenia algorytmu drzewa decyzyjnego przeprowadzono odpowiednie testy w celu sprawdzenia poprawności implementowanego kodu. Każdy z poszczególnych etapów tego fragmentu projektu to znaczy:

* wyznaczanie entropii,
* wyznaczanie parametru information gain,
* tworzenie drzewa,
* predykcja

został dokładnie przetestowany w celu uniknięcia późniejszych problemów przy pracy na większych zbiorach danych. Pierwszym testem było wyznaczenie przez zaimplementowane funkcje zawarte w pliku "information_gain.py" entropii oraz później parametru information gain. Wartości te obliczono dla identycznego jak opisany w dokumentacji wstępnej zbioru danych, które następnie porównano.

W kolejnym kroku po zaimplementowaniu zmodyfikowanego algorytmu id3 przystąpiono do sprawdzania poprawności budowania drzewa dla tego samego zbioru danych oraz następnie dla popularnego zbioru zwiazanego z pogodą.

Ostatnim etapem tworzenia algorytmu drzewa decyzyjnego było sprawdzenie skuteczności działania predykcji. Dla różnych zbiorów danych porównywano działanie naszego algorytmu z algorytmem referencyjnym w postaci gotowej klasy drzewa decyzyjnego z biblioteki scikit-learn.

4 Las losowy

Algorytm lasu losowego, jest zbiorem klasyfikatorów (drzew decyzyjnych), który został tak zaimplementowany, aby zawierał dodatkowo odpowiednią ilość klasyfikatorów NBC. Oznacza to, że klasyfikator NBC może stanowić, co któryś w kolei klasyfikator w lesie. Wartość tą wraz z pozostałymi parametrami określa się w trakcie inicjalizacji obiektu lasu losowego.

Implemetnacja algorytmu lasu losowego o dokładnie opisanych w dokumentacji wstępnej założeniach znajduje się w pliku "rf_id3_nb.py", a jego uproszczona zasada działania przedstawiona jest na rysunku "Algorithm 2".

Algorithm 2 Algorytm trenowania lasu losowego

Input: N - liczba modeli, S - zbiór danych

function RandomForest_NaivyBayes.fit(N, S)
    for n : od 1 do N: **do**
        Wylosuj ze zwracaniem k-elementową próbę bootstrapową Sn ze zbioru S
        if num_built % nb_at_every **then**
            Mn = Stwórz model naiwnego klasyfikatora bayesowskiego dla próby Sn)
        else
            Mn = stwórz model przy użyciu drzewa używając algorytmu Id3 dla próby Sn
        end if
        Dodaj model Mn do listy klasyfikatorów.
    end for
end function

function RandomForest_NaivyBayes.predict($S_{testowy}$)
    for n : od 1 do Liczba modeli: **do**
        Wyznacz predykcje dla modelu n.
        Dodaj predykcje do listy Sp.
    end for
    Wyznacz predykcje p z listy Sp.
    return Predykcja p
end function

Przy tworzeniu obiektu typu "RandomForest_NaivyBayes" można ustawić następujące parametry:

- num_trees = liczba klasyfikatorów w lesie
- max_depth = maksymalna głębokość drzewa
- nb_at_every = co który klasyfikator w lesie będzie NBC

5 Eksperymenty i uzyskane wyniki

Poniższe eksperymenty zostały przeprowadzone dla dwóch opisanych wcześniej zbiorów, na których przetrenowane zostały wszystkie z rozważanych w eksperymencie modeli. Modele oceniano metrykami dokładnie opisanymi w dokumentacji wstępnej:

* accuracy,
* precision,
* recall,
* F1.

5.1 Porównanie modeli default parametrów.

Table 1: Porównanie modeli default dla zbioru Cars

	RF Baseline	RF Custom
Accuracy	0.96	0.86
F1	0.91	0.73
Precision	0.92	0.89
Recall	0.91	0.67

Table 2: Porównanie modeli default dla zbioru Votes

	RF Baseline	RF Custom
Accuracy	0.97	0.94
F1	0.97	0.90
Precision	0.97	0.95
Recall	0.97	0.93

Zaimplementowany model spisuje sie podobnie na defaultowych parametrach jak Random Forest z scikit-learn.

5.2 Porównanie modeli podstawowych ID3 vs Naiwny Bayess.

Table 3: Porównanie modeli default dla zbioru Cars

	ID3	Naiwny Bayess
Accuracy	0.95	0.75
F1	0.87	0.45
Precision	0.88	0.61
Recall	0.88	0.42

Table 4: Porównanie modeli default dla zbioru Votes

	ID3	Naiwny Bayess
Accuracy	0.94	0.90
Precision	0.90	0.84
F1	0.93	0.92
Recall	0.92	0.87

5.3 Porównanie hiperametrów modelu RF Custom.

5.3.1 Liczba estymatorów

Table 5: Wyniki dla liczby estymatorów dla Cars

Num Trees	10	20	50	100	150	200
Accuracy	0.89	0.90	0.87	0.87	0.86	0.86
Macro F1	0.77	0.78	0.76	0.77	0.75	0.76
Macro Precision	0.89	0.90	0.88	0.90	0.90	0.89
Macro Recall	0.71	0.72	0.70	0.70	0.70	0.69

Table 6: Wyniki dla liczby estymatorów dla Votes

Num Trees	10	20	50	100	150	200
Accuracy	0.94	0.94	0.94	0.94	0.93	0.94
Macro Precision	0.92	0.91	0.89	0.89	0.88	0.89
Macro F1	0.94	0.94	0.95	0.95	0.96	0.96
Macro Recall	0.93	0.92	0.92	0.92	0.92	0.92

5.3.2 Głębokość drzewa

Table 7: Wyniki dla głębokości drzewa dla Cars

Max depth	5	10	20	30	50	100	150	200
Accuracy	0.82	0.86	0.86	0.87	0.87	0.87	0.87	0.87
Macro F1	0.54	0.75	0.75	0.75	0.76	0.76	0.76	0.76
Macro Precision	0.73	0.89	0.89	0.89	0.90	0.90	0.90	0.90
Macro Recall	0.50	0.69	0.69					

Table 8: Wyniki dla głębokości drzewa dla Votes

Max depth	5	10	20	30	50	100	150	200
Accuracy	0.94	0.93	0.94	0.93	0.94	0.94	0.94	0.94
Macro Precision	0.89	0.89	0.89	0.89	0.89	0.89	0.90	0.89
Macro F1	0.96	0.95	0.96	0.95	0.96	0.95	0.95	0.96
Macro Recall	0.92	0.92	0.92	0.92	0.92	0.92	0.92	

5.3.3 Wyniki dla parametru nb_at_every(co który naiwny bayess).

Table 9: Wyniki nb_at_every dla Cars

NB position	2	3	4	5	10
Accuracy	0.87	0.78	0.77	0.76	0.76
Macro F1	0.76	0.65	0.55	0.51	0.47

Do porównania użyjemy następujących modeli na 2 opisanych wcześniej zbiorach danych.

- zaimlementowany model: las losowy(drzewa ID3 i naiwny bayess)
- lasu losowego z biblioteki scikit-learn:
  sklearn.ensemble.RandomForestClassifier
- naiwny bayess z biblioteki scikit-learn: sklearn.naive_bayes.CategoricalNB

6 Wnioski

Do eksperymentów wykorzystano dwa różne zbiory różniące się między sobą ilością atrybutów, klas i przykładów, spodziewano się więc widocznych w wynikach różnic. Różnorodność wybranych zbiorów pozwoliła dokładnie sprawdzić skuteczność stworzonego algorytmu klasyfikacji i ocenić jego zdolność do pracy na różnych zbiorach.

Dzięki temu projektowi pogłębiliśmy swoją wiedzę z python'a oraz zrozumieliśmy koncepcję działania lasu losowego wraz z klasyfikatorem bayesowskim.