Uczenie się Maszyn
Projekt "Zastosowanie drzew decyzyjnych do klasyfikacji miejsc rozcięcia w sekwencji DNA"

Dokumentacja końcowa

Igor Markiewicz

1 Spis treści

2 Założenia

- W trakcie procesu testowania programu okazało się że istnieje niewielka liczba przykładów w obu dostępnych zbiorach, które zawierają dodatkową literę N, oznaczającą najprawdopodobniej nieznaną lub dowolną zasadę azotową. Dlatego też do istniejących już czterech wartości atrybutów A, C, G i T została dołożona wartość N.
- Podział na zbiory uczący, walidacyjny i testowy przebiega w dwóch etapach. Najpierw procentowo określa się ile przykładów z pierwotnie dostępnego zbioru zostaje przeznaczonych na uczenie i walidację (reszta idzie na testowanie). Następnie z części przeznaczonej na uczenie i walidację, określa się procentowo ile przykładów jest przeznaczonych na uczenie (reszta idzie na walidację).
- Dla każdego pomiaru losowanie zbiorów, uczenie, (przycinanie) i testowania zostają przeprowadzone dziesięciokrotnie w celu poprawienia statystyki, a wyniki są wyznaczane średnią arytmetyczną (i ewentualnie zaokrąglane).
- Do przetestowania budowania drzewa został użyty przykład przedstawiony w dokumentacji wstępnej.
- Badania :
  - Eksperyment 1 - Podział zbioru pierwotnego na zbiór uczący o liczności 10 %, 20 %, ... , 90 % oraz zbiór testowy i badanie parametrów matrix confusion, błędu.
  - Eksperyment 2 - Podział zbioru pierwotnego na zbiór będący sumą zbioru uczącego i walidujacego (70 %) oraz na zbiór testowy (30 %), a następnie podział na zbiór uczący o liczności 10 %, 20 %, ... , 90 % i zbiór walidujący. Badamy parametry matrix confusion oraz błąd zarówno dla zbioru walidującego (czyli przed przycięciem drzewa) jak i dla zbioru testowego (po przycięciu drzewa, gdzie zbiorem do przycinania jest zbiór walidujący).

3 Przycinanie drzewa

3.1 Opis

Jako algorytm przycinania drzewa zostało wybrane Reduce Error Pruning, będące algorytmem zstępującym.

funkcja pruneDecisionTree(*root*, *data*)
argumenty wejściowe:

- *root* - korzeń drzewa decyzyjnego
- *data* - zbiór przycinający

zwraca: - (działa w miejscu względem oryginalnego drzewa);
1: **dopóki** korzeń nie jest sprawdzony
2: pruneEngine(*root*, *data*);
3: **koniec dopóki**

funkcja pruneEngine(*node*, *data*)
argumenty wejściowe:

- *node* - węzeł drzewa decyzyjnego
- *data* - zbiór przycinający

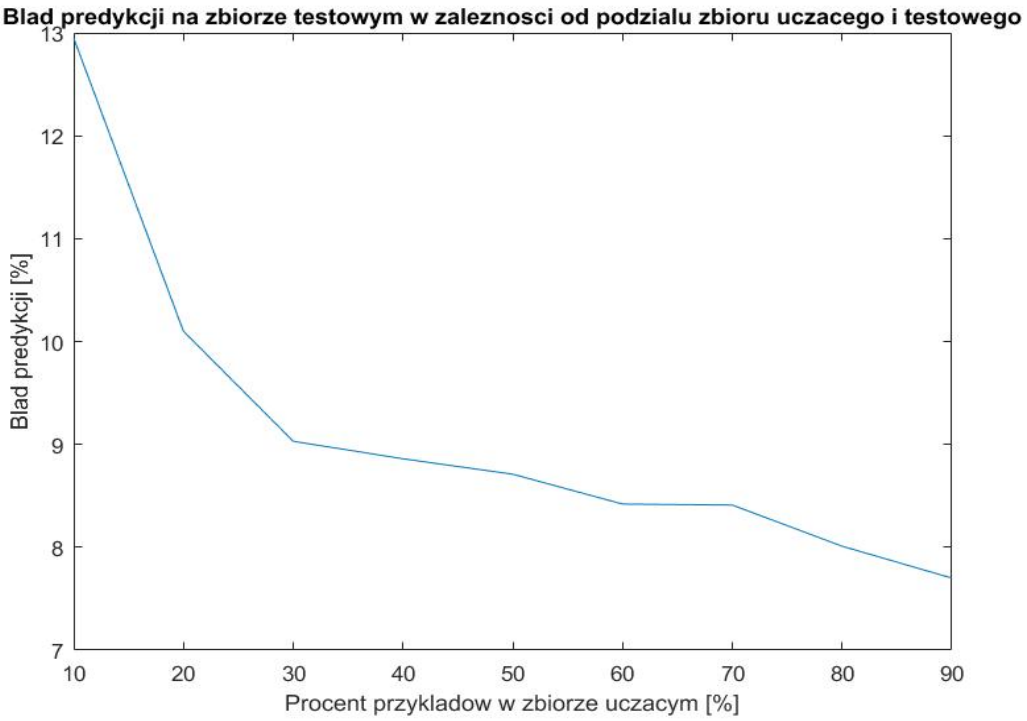
zwraca: - (działa w miejscu względem oryginalnego drzewa);
1: **jeśli** wszyscy potomkowie węzła nie są liśćmi lub nie są sprawdzeni lub nie ma superpozycji tych stanów **to**
2: **dla każdego** potomka węzła
3: **jeśli** potomek nie jest liściem i nie jest sprawdzony **to**
4: pruneEngine(*child*, *data*);
5: **koniec jeśli**
6: **koniec dla każdego**
7: **w przeciwnym przypadku**
8: ustaw węzeł jako sprawdzony;
9: sprawdź klasyfikację c_1 w podanym węźle drzewa i podanym zbiorze;
10: zbuduj przewidywaną klasyfikację c_2 na podstawie kategorii większościowej w klasyfikacji c_1;
11: oblicz błąd dla c_1 i c_2;
12: **jeśli** błąd dla klasyfikacji c_2 jest mniejszy lub równy błędowi dla klasyfikacji c_1 **to**
13: ustaw węzeł jako liść;
14: przypisz do utworzonego liścia kategorię większościową klasyfikacji c_1;
15: Usuń rekurencyjnie potomków i ustaw wskaźniki na dzieci na wartość domyślną;
16: **koniec jeśli**
17: **koniec w przeciwnym przypadku**
18: **koniec jeśli**

4 Analiza i wnioski

4.1 Eksperyment 1.

4.1.1 Rozcięcia typu donorowego

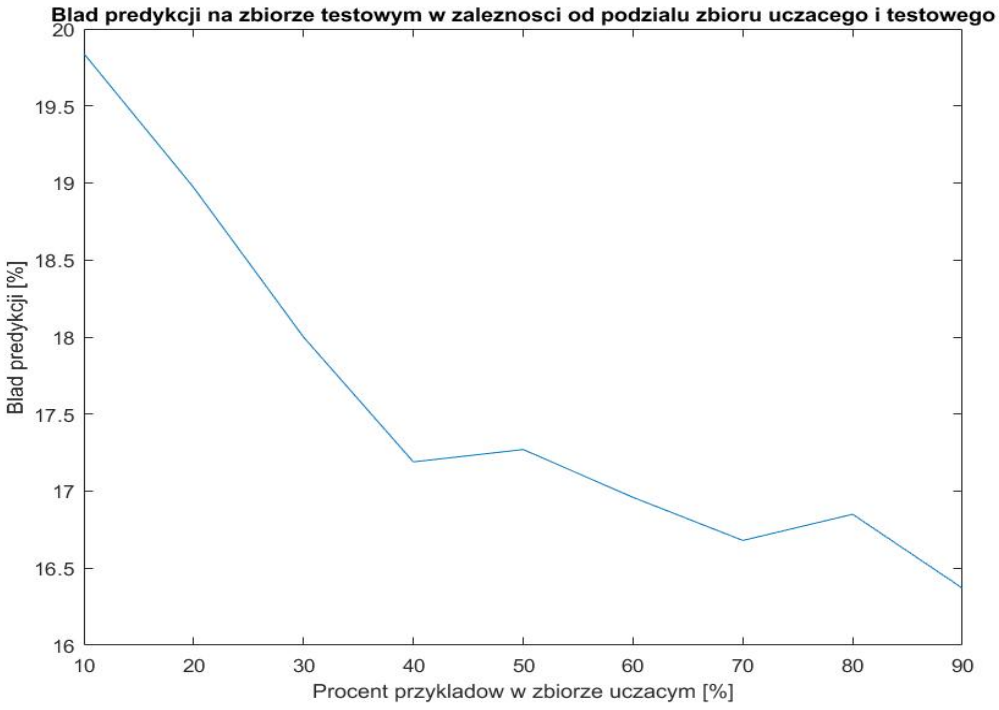
Nr	error[%]
1	12.95
2	10.10
3	9.03
4	8.86
5	8.71
6	8.42
7	8.41
8	8.01
9	7.70



Rysunek 1:

4.1.2 Rozcięcia typu akceptorowego

Nr	error[%]
1	19.84
2	18.97
3	18.00
4	17.19
5	17.27
6	16.96
7	16.68
8	16.85
9	16.37



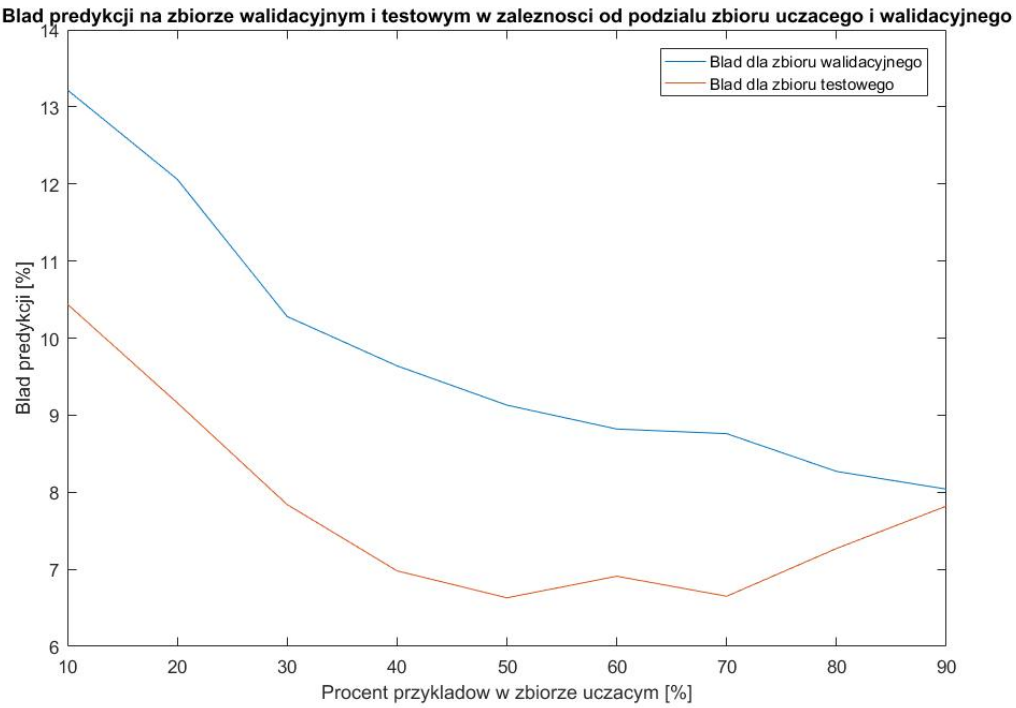
Rysunek 2:

W obu przypadkach zauważamy że następuje wraz ze wzrostem ilości przykładów w zbiorze uczącym, spadek błędu predykcji. Jest to zgodne z oczekiwaniami mówiącymi że im większa liczba przykładów w zbiorze uczącym, tym lepiej algorytm może nauczyć się rozdzielać klasy. Porównując algorytm dla rozcięć typu donorowego i akceptorowego zauważamy, że ze względu na podobną liczbę przykładów (odpowiednio 5256 i 5788), różnica w zakresie błędów wynika najprawdopodbniej z długości ciągów znakowych (odpowiednio 15 i 90). Jest to również zgodne z oczekiwaniem, że dla bardziej skomplikowanych przykładów prawdopodobieństwo popełnienia pomyłki przy klasyfikacji jest większe.

4.2 Eksperyment 2.

4.2.1 Rozcięcia typu donorowego

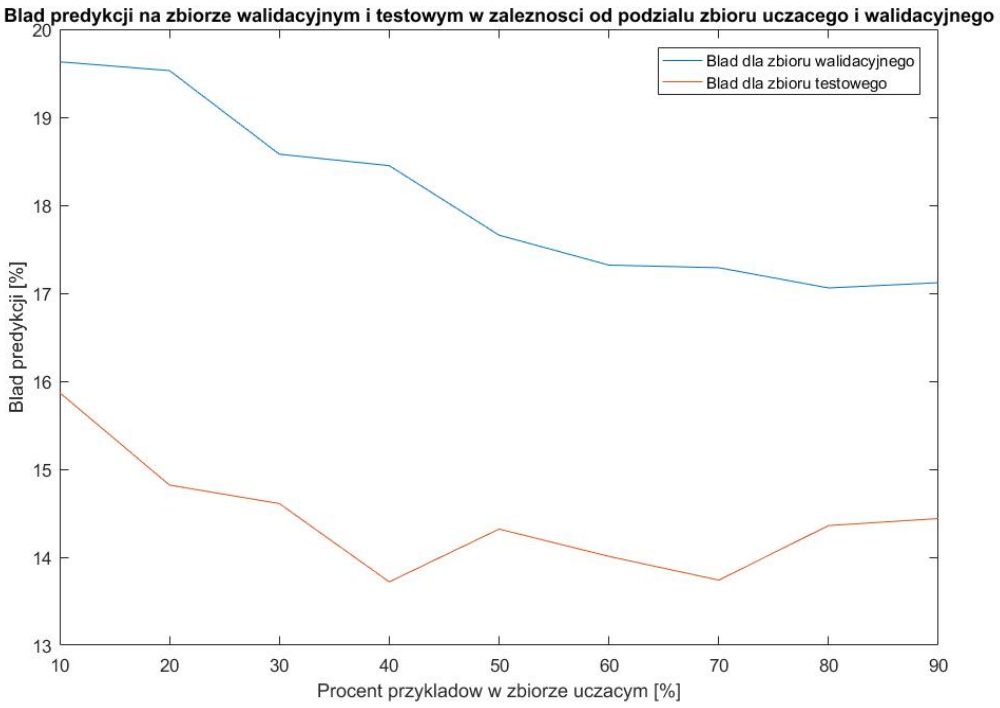
Nr	validError[%]	testError[%]
1	13.22	10.44
2	12.06	9.16
3	10.28	7.84
4	9.64	6.98
5	9.13	6.63
6	8.82	6.91
7	8.76	6.65
8	8.27	7.27
9	8.04	7.82



Rysunek 3:

4.2.2 Rozcięcia typu akceptorowego

Nr	validError[%]	testError[%]
1	19.63	15.87
2	19.53	14.82
3	18.58	14.61
4	18.45	13.72
5	17.66	14.32
6	17.32	14.01
7	17.29	13.74
8	17.06	14.36
9	17.12	14.44



Rysunek 4:

Zauważamy że dla rozcięć typu donorowego wraz ze wzrostem ilości przykładów trenujących, spada błąd predykcji zarówno na zbiorze walidacyjnym (przed przycięciem) jak i na zbiorze testowym (po przycięciu) aż do ok 70 % rozmiaru zbioru uczenie + walidacja, kiedy zaczyna rosnąć błąd testowy. Dla rozcięć typu akceptorowego przycięcie powoduje spadek błędu predykcji do ok 40 % rozmiaru zbioru uczenie + walidacja, a następnie oscylacje błędu. Oba powyższe efekty są najprawdopodobniej spowodowane tym, że dla dużej liczby przykładów uczących, przycinanie następuje dla coraz lepszych hipotez i w efekcie niewiele wnosi czy wręcz może pogarszać predykcję. Porównując algorytm dla

rozcięć donorowych jaki i akceptorowych, podobnie jak w poprzednim eksperymencie zauważamy że dla bardziej skomplikowanego zbioru danych, błędy predykcji są większe.

5 Bibliografia

[1] Wykłady do przedmiotu Uczenie się Maszyn

[2] P.Cichosz *Systemy Uczące się*, Wydanie Drugie, Wydawnictwo Naukowo-Techniczne, Warszawa 2000, 2007

[3] https://en.wikipedia.org/wiki/Confusion_matrix