1 Treść zadania

Las losowy w zadaniu klasyfikacji. Podczas budowy każdego z drzew testy (z pełnego zbioru testów) wybieramy za pomocą selekcji progowej w połączeniu z ruletką, czyli odrzucamy część najgorszych, po czym każdy z pozostałych testów ma proporcjonalną do swojej jakości szansę na bycie wybranym.
Interpretacja: celem projektu jest stworzenie programu budującego decyzyjny las losowy. W tracie budowy drzew testy są wybierane przez selekcję progową z ruletką bazującą na funkcji jakości podziału przykładów przez dany test.
Założenia i wymagania: implementacja w R, Python, C++ lub C. Algorytm ma być niezależny od zbioru danych. Program musi działać pod kontrolą Ubuntu Linux 22.04.

2 Opis algorytmu

2.1 Drzewa decyzyjne

Zdecydowaliśmy się na oparcie naszego rozwiązania o istniejące już algorytmy takie jak ID3 i C4.5. Ich standardowe implementacje budowy drzewa decyzyjnego bazują na wybieraniu najlepszego (niewykorzystanego) atrybutu w każdej iteracji, na bazie którego w węzłach dokonywany jest podział na poszczególne gałęzie [1]. W naszym rozwiązaniu miarą jakości podziału będzie zysk informacyjny ozn. $\mathrm{IG}(x, S)$. Dla podziału zbioru S atrybutem $x \in X$ określa go wzór:

$$\mathrm{IG}(x, S) = H(S) - H(S|x)$$

przy czym:

- $H(A) = -\sum_{c \in C} p(c) \log_2 p(c)$ to entropia zbioru A, gdzie $p(c)$ to liczba przykładów klasy c w zbiorze A podzielona przez moc zbioru A,
- $H(S|x) = \sum_{v \in V_x} \frac{|S_{x=v}|}{|S|} H(S_{x=v})$ to entropia warunkowa, gdzie V_x to zbiór wartości atrybutu x, a $S_{x=v}$ to przykłady z S, dla których $x = v$.

W naszym rozwiązaniu nie będziemy wybierać jednak atrybutów o największym zysku informacji jak robi to algorytm ID3 lub C4.5. Po obliczeniu IG dla wszystkich niewykorzystanych wcześniej atrybutów najpierw odrzucimy te, których IG będzie poniżej określonego progu, a pozostałe atrybuty będą miały szansę bycia wybranym proporcjonalną do swojej jakości. Prawdopodobieństwo wyboru danego atrybutu $x \in X$ to:

$$p(x) = \frac{\mathrm{IG}(x, S)}{\sum_{y \in X} \mathrm{IG}(y, S)}$$

gdzie S to zbiór przykładów rozważany (i dzielony) w danym węźle, a X to zbiór niewykorzystanych atrybutów. Atrybut podziału A będzie zatem wynikiem losowania ze zbioru atrybutów $\{x_1, x_2, ..., x_n\}$, gdzie $P(A = x_i) = p(x_i)$ zgodnie z powyższą definicją.

Jesteśmy zatem w stanie sformułować algorytm budowy modeli bazowych w postaci pseudokodu [2][3]:

Algorytm 1 Pseudokod budowy modeli bazowych

```
1:  function DecisionTree(S, X) returns tree
2:      Let T be a new tree
3:      if all elements of S are of some class c then
4:          addLeaf(T, c); return T
5:      end if
6:      X' ← threshold(S, X)
7:      if X' is empty then
8:          addLeaf(T, bestClass(S)); return T
9:      end if
10:     A ← sample(X')
11:     addNode(T, A)
12:     for each value x of A do
13:         S_x ← elements of S with A = x
14:         if S_x is empty then
15:             Let T_x be a new tree
16:             addLeaf(T_x, bestClass(S))
17:         else
18:             T_x ← DecisionTree(S_x, X − {A})
19:         end if
20:         addBranch(T, T_x, x)
21:     end for
22:     return T
23: end function
```

gdzie:

- `addLeaf`(T,c): dodaje do drzewa T liść o klasie c,
- `addNode`(T,A): dodaje do drzewa T nowy węzeł z podziałem względem atrybutu A,
- `addBranch`(T, T_x, x): dodaje gałąź pomiędzy drzewem T i T_x opisaną przez wartość x atrybutu A,
- `bestClass`(S): zwraca klasę c, która pokrywa maksymalną liczbę przykładów w zbiorze S,
- `threshold`(S,X): zwraca zbiór par (atrybut-zysk) dla atrybutów z X, które przy podziale zbioru S dają zysk informacyjny wyższy od określonego progu (mechanizm selekcji progowej),
- `sample`(X'): losuje atrybut ze zbioru X' par (atrybut-zysk). Prawdopodobieństwo wylosowania jest proporcjonalne do zysku informacyjnego.

W powyższym pseudokodzie w pętli `for each` istotne staje się rozważenie przypadku atrybutów ciągłych. Można je rozpatrzeć na dwa sposoby:

- podział na przedziały i dyskretyzację.
- wyznaczenie wartości atrybutu, dla której uzyskamy największy zysk informacyjny przy podziale. Może być zrealizowane przez binarny warunek $A \leqslant t$. Wartość progową t można szukać jako średnią arytmetyczną dwóch wartości x_i, x_{i+1}, będących wartościami atrybutu A dla kolejnych przykładów posortowanego względem A zbioru S. Metoda wykorzystywana w algorytmie C4.5 [4].

2.2 Las losowy

Budowę lasu losowego rozpoczyna się od budowy N drzew decyzyjnych (zazwyczaj co najmniej kilkuset) . Ich tworzenie można podzielić na dwa etapy:

- dla każdego drzewa niezależnie losuje się pewną listę atrybutów z pełnego zbioru m atrybutów. Zazwyczaj losuje się ich $\lfloor\sqrt{m}\rfloor$.
- dla każdego drzewa niezależnie losuje się ze zwracaniem zbiory (z dopuszczonymi duplikatami) ze zbioru przykładów treningowych. Na ich podstawie trenuje się drzewa, rozważając podział jedynie względem wylosowanych atrybutów.

W wyniku otrzymujemy N klasyfikatorów postaci $h(x)$, gdzie x jest wektorem (przykładem) wejściowym. Klasyfikacja przykładu x polega na wprowadzeniu go do każdego z wygenerowanych klasyfikatorów, a następnie na zagłosowaniu na najczęściej występującą klasę [5]. Możemy zatem zapisać algorytm budowy decyzyjnego lasu losowego w postaci pseudokodu:

Algorytm 2 Pseudokod decyzyjnego lasu losowego

```
1:  function RandomForest(TS, N, S) returns classifications
2:      Let Trees be an empty array
3:      A ← attributes(TS)
4:      for i=1,...,N do
5:          T ← sampleR(TS, |TS|)
6:          X ← sampleNR(A, ⌊√|A|⌋)
7:          Add DecisionTree(T, X) to Trees
8:      end for
9:
10:     Let P be an empty list of classes
11:     for each value s of S do
12:         Let C be an array of counters for all classes
13:         Set all C's counters to 0
14:         for each tree of Trees do
15:             class ← tree's classification of s
16:             Increment counter of class in C
17:         end for
18:         Add class with the biggest C counter to P
19:         or select one at random if there are > 1
20:     end for
21:     return P
22: end function
```

gdzie:

- *TS*: zbiór treningowy.
- *N*: liczba drzew decyzyjnych.
- *S*: przykłady do sklasyfikowania.
- `attributes`(*TS*): zwraca listę atrybutów przykładów ze zbioru *TS*.
- `sampleR`(*Y*,*n*): losuje ze zwracaniem *n* przykładów ze zbioru *Y* zgodnie z rozkładem jednostajnym.
- `sampleNR`(*Y*,*n*): losuje bez zwracania *n* przykładów ze zbioru *Y* zgodnie z rozkładem jednostajnym. Nie modyfikuje *Y*.

3 Plan eksperymentów

Planujemy wykonać szereg eksperymentów, aby zbadać wpływ określonych parametrów na jakość uzyskiwanego modelu. Takimi eksperymentami będą:

- Manipulowanie wielkością listy losowanych atrybutów podczas budowy drzew decyzyjnych. Porównanie wpływu tego parametru dla zbiorów danych o różnej złożoności.
- Manipulowanie ilością modeli bazowych, jaką wykorzystujemy do budowy lasu losowego.

Planujemy również porównać jakość predykcji lasu losowego uzyskanego w wyniku naszej implementacji z rozwiązaniem, w którym podczas budowy modeli bazowych wybierane są atrybuty o największym zysku informacji (ID3, C4.5). Szczególną uwagę zwrócimy na zbiory z wieloma atrybutami wielowartościowymi. Przypuszczamy, że nasze rozwiązanie może sprawdzić się dla nich najlepiej.

4 Sposób oceny jakości

Dla każdego typu modelu lasu (tj. konkretnych parametrów) przeprowadzimy wielokrotne próby w celu oceny jego jakości. W każdej iteracji zbadamy odpowiednie metryki, a na końcu je uśrednimy. Szczególną uwagę zwrócimy na dokładność ACC.

Do porównania różnych typów modeli na wykresie w przestrzeni ROC naniesiemy odpowiadające im punkty, będące wynikiem uśrednienia prób.

Aby w sposób poprawny ocenić i porównać wpływ konkretnych parametrów na jakość modelu, zawsze będziemy modyfikować tylko jeden parametr.

5 Zbiory danych

Zdecydowaliśmy się na dwa zbiory danych, aby wykorzystać algorytm zarówno do klasyfikacji binarnej jak i do wieloklasowej. Ważna jest dla nas również dziedzina atrybutów. Szukaliśmy zbiorów, które zawierają zarówno dane ciągłe jak i dyskretne. Wybranymi przez nas zbiorami są:

- **Adult data set** (źródło) - atrybuty o wartościach dyskretnych i ciągłych, klasa binarna, niezbalansowany.
- **Red Wine Quality** (źródło) - atrybuty o wartościach ciągłych, wiele klas, niezbalansowany.

5.1 Adult data set

Zbiór danych składa się z opisów osób fizycznych, którym przypisywana jest klasa "$<= 50$K" lub "> 50K" w zależności od tego, czy ich roczny przychód przekracza 50 tysięcy dolarów. Sześć atrybutów jest ciągłych, osiem dyskretnych (większość wielowartościowych).

Zbiór składa się z 32561 instancji, zawiera puste wartości i jest niezbalansowany.

Ilość wystąpień instancji poszczególnych klas:

Klasa	N	N[%]
>=50K	24720	75,92%
<50K	7841	24,08%
total	32561	

5.2 Red Wine Quality

Zbiór danych, który zawiera analizę chemiczną różnych wariantów tego samego portugalskiego wina. Wszystkie atrybuty zawierają wartości ciągłe, a wynikowy atrybut to ocena jakości w skali od 0 do 10. W zbiorze danych występują jedynie wina o jakości od 3 do 8, więc tylko one będą istotne w zadaniu klasyfikacji. Zbiór składa się z 1599 przykładów i jest niezbalansowany - najwięcej jest win o jakości 5 i 6.
Ilość wystąpień instancji poszczególnych klas:

Klasa	N	N[%]
3	10	0,6%
4	53	3,3%
5	681	42,6%
6	638	40%
7	199	12,4%
8	18	1,1%
total	1599	

6 Zmiany względem dokumentacji wstępnej

Zgodnie z informacją otrzymaną od Prowadzącego zdecydowaliśmy się zmienić zbiór testowy Mushroom Data Set na zbiór Adult. Wcześniej wybrany zbiór był zbyt prosty. Zrezygnowaliśmy również z obrazowania punktów w przestrzeni ROC dla kolejnych iteracji agregowanych modeli ze względu na nieczytelność wykresów. Pozostaliśmy przy obrazowaniu jedynie zagregowanych wyników dla rozważanej wersji modelu.

Do naszego rozwiązania dodaliśmy tryby obsługi atrybutów ciągłych: "mean" (podział względem średniej), "median" (względem mediany), "buckets" (szukanie najlepszego podziału przez kubełkowanie), "search" (liniowe poszukiwanie wśród średnich); oraz tryby selekcji progowej: "none" (odrzucanie podziału, gdy IG=0), "half" (odrzucenie połowy najgorszych podziałów), "best" (wybór najlepszego podziału).

7 Struktura rozwiązania

Tworząc nasze rozwiązanie zadbaliśmy o rozdzielenie implementacji poszczególnych części algorytmu i funkcji/modeli pomocniczych. Ostatecznie nasze rozwiązanie przybrało strukturę opisaną poniżej.

Folder `data` zawiera pliki z wykorzystywanymi zbiorami danych:

- `adult.csv`: związany ze zbiorem Adult,
- `wine.csv`: związany ze zbiorem Red Wine Quality.

Folder `src` zawiera implementacje algorytmów i funkcji pomocniczych. Znajdują się w nim:

- folder `forest` – zawierający implementację lasu losowego w pliku `random_forest.py`
- folder `decision_tree` – zawierający pliki związane z drzewem decyzyjnym:
  - ◇ plik `node.py`: definiujący obiekt węzła drzewa decyzyjnego,
  - ◇ plik `branch.py`: definiujący obiekt gałęzi łączącej węzły drzewa decyzyjnego,
  - ◇ plik `decision_tree.py`: definiujący obiekt drzewa decyzyjnego korzystającego z mechanizmu selekcji progowej z ruletką,
  - ◇ plik `entropy_handler.py`: definiujący funkcje związane z działaniem na entropii.
- folder `utilities` – zawierający pliki z funkcjami pomocniczymi:
  - ◇ plik `dataframes_handler.py`: definiujący funkcje związane z obsługą zbiorów danych,
  - ◇ plik `quality.py`: definiujący funkcje związane z miarami jakości.

Folder `tests` zawiera plik `test_forest.py`, w którym znajdują się wywołania symulacji wykorzystanych w trakcie zbierania wyników eksperymentów.

Folder `output`, do którego trafiają wszystkie pliki wynikowe takie jak np. obraz `confusion_matrix.png` przedstawiający tablicę pomyłek.

W pliku `main.py` zdefiniowaliśmy "demo" działania algorytmu – jego użycie opisane jest wewnątrz pliku.

Uwaga: ze względu na strukturę rozwiązania poszczególne programy należy wykonywać z najwyższego folderu (`kod`, w którym znajdują się `data`, `output`, `src`, `tests` i plik `main.py`) korzystając z polecenia:

```
python -m <folder>.<plik>
```

przy czym plik należy podać bez rozszerzenia `.py`. Przykładowo, poprawne jest wywołanie:

```
python -m tests.test_forest
```

a niepoprawne jest:

```
python -m /tests/test_forest.py
```

Taki sposób wykonania, podobnie jak samo rozwiązanie, jest uniwersalne dla wszystkich systemów operacyjnych. Postaraliśmy się utworzyć czytelny, samodokumentujący się kod. Nieoczywiste fragmenty kodu zostały dodatkowo skomentowane.

8 Wykorzystywane narzędzia

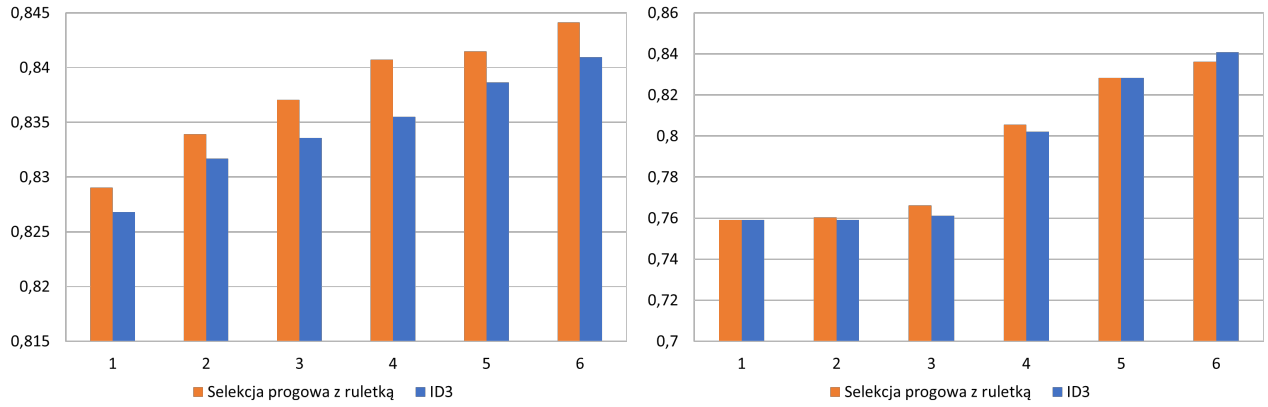
Rozwiązanie wykonaliśmy w języku Python w wersji 3.10 z wykorzystaniem bibliotek `numpy`, `pandas` (2.0.2), `matplotlib`, `seaborn` i `scikit-learn 1.2.2`.

9 Eksperymenty

Uwaga: dbając o estetykę sprawozdania wyniki poszczególnych eksperymentów przedstawione w postaci tabelarycznej zostały umieszczone w ostatniej sekcji dokumentu. Wyniki dotyczące konkretnego eksperymentu znajdują się w odpowiednio opisanej sekcji. W tej sekcji wyniki są przedstawione w formie wykresów.

9.1 Weryfikacja

Działanie naszego algorytmu wstępnie zweryfikowaliśmy porównując go z modelem referencyjnym. Wybraliśmy ID3 zaimplementowane w paczce `sklearn` – obiekt TreeClassifier działający w trybie `entropy` (podział na podstawie zysku informacyjnego). Aby rzetelnie porównać oba modele konieczne było odpowiednie skalibrowanie naszego rozwiązania: zysk informacyjny atrybutów ciągłych wyznaczamy przez kubełkowanie, wybieramy najlepszy podział. Taka konfiguracja najwierniej odwzorowuje zachowanie modelu TreeClassifier. Następnie przeprowadziliśmy 20 symulacji badając wpływ zmiany liczby atrybutów wykorzystywanych do budowy drzew. Ustaliliśmy liczbę wykorzystywanych modeli bazowych na 200. Na rysunku 1a przedstawiamy wyniki dla zbioru `Wine Quality`, a na rysunku 1b dla zbioru `Adult`.

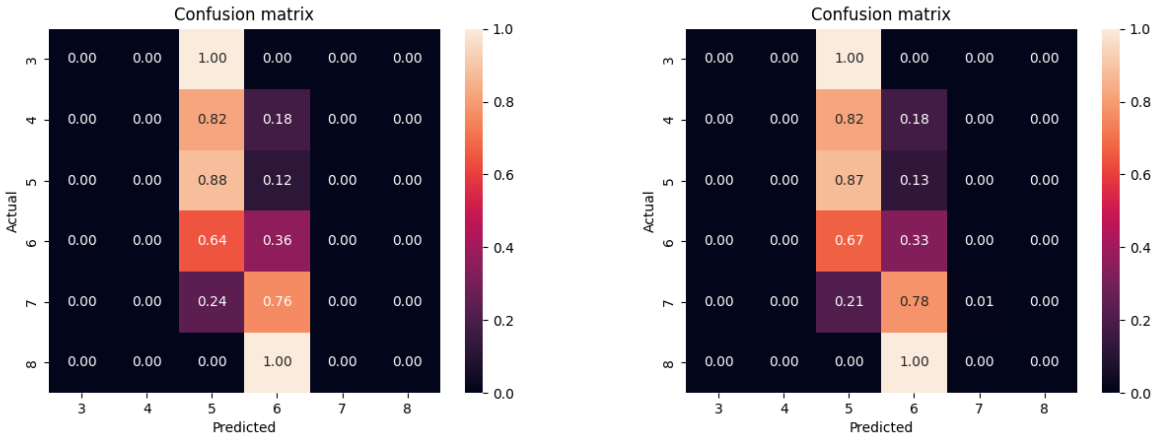


(a) *Wykres dokładności ACC od liczby atrybutów – zbiór* ***`Wine Quality`***

(b) *Wykres dokładności ACC od liczby atrybutów – zbiór* ***`Adult`***

Rys. 1: *Weryfikacja modelu – manipulowanie liczbą atrybutów*

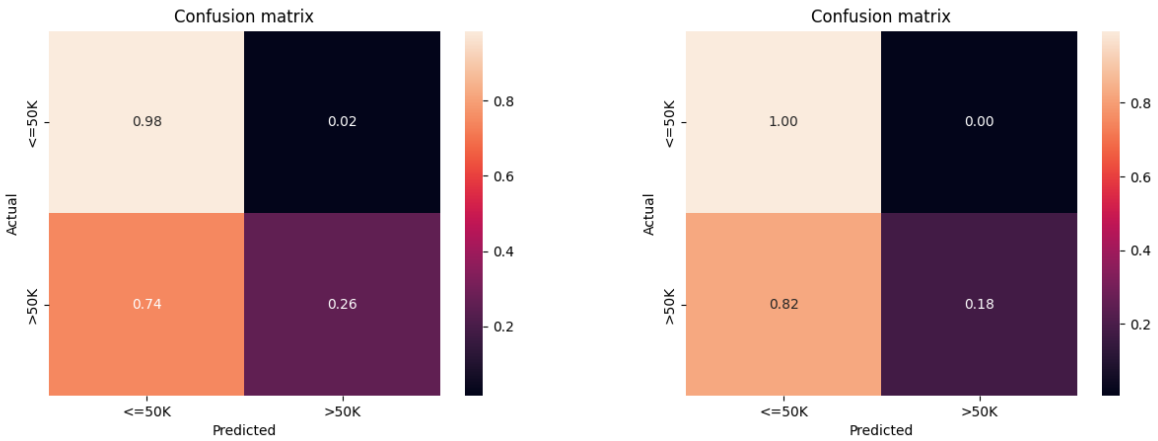
W obu testach nasze rozwiązanie zachowało się w sposób podobny do modelu referencyjnego. Dla zbioru `Wine Quality` dla każdej liczby atrybutów poradziło sobie nieznacznie lepiej. Dla zbioru `Adult` oba modele są porównywalnie dokładne. Podobne wnioski możemy wyciągnąć analizując tablice pomyłek. Dla każdego zbioru danych wartości w odpowiednich komórkach są do siebie bardzo zbliżone. Tablice pomyłek obu modeli dla zbioru `Wine Quality` przedstawiają rysunki 2a i 2b, a dla zbioru `Adult` rysunki 3a i 3b. Na podstawie powyższych obserwacji (i dalszych eksperymentów) stwierdzamy, że zaimplementowany przez nas model działa poprawnie.



(a) *Tablica pomyłek – selekcja progowa z ruletką*

(b) *Tablica pomyłek – model wykorzystujący ID3*

Rys. 2: *Weryfikacja modelu – porównanie tablic pomyłek dla zbioru* ***`Wine Quality`*** *(wybór 4 atrybutów)*



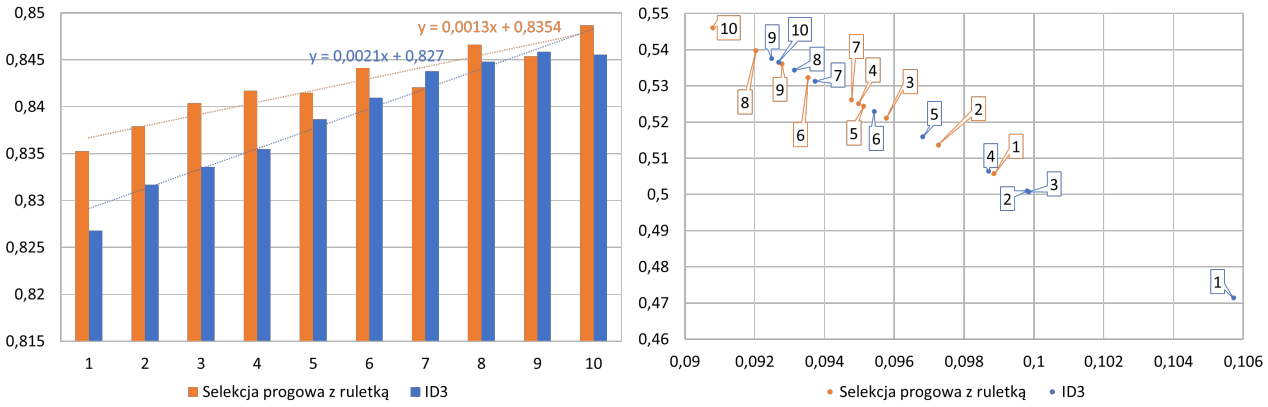
(a) *Tablica pomyłek – selekcja progowa z ruletką*

(b) *Tablica pomyłek – model wykorzystujący ID3*

Rys. 3: *Weryfikacja modelu – porównanie tablic pomyłek dla zbioru* `Adult` *(wybór 4 atrybutów)*

9.2 Liczba wykorzystywanych atrybutów

Zbadaliśmy wpływ liczby atrybutów wykorzystywanych do budowy modeli bazowych na jakość lasu losowego. Wyniki dla modelu wykorzystującego selekcję progową z ruletką zestawiliśmy z modelem referencyjnym (wykorzystującym ID3). Testy przeprowadziliśmy dla następujących ustawień naszego modelu: `split_strategy='mean'`, `threshold='half'` i dla liczby drzew równej 200. Po zebraniu zagregowanych wyników 15 symulacji dla zbioru `Wine Quality`, utworzyliśmy wykresy dokładności ACC oraz wykres punktów w przestrzeni ROC w zależności od liczby wykorzystywanych atrybutów. Wyniki przedstawia rysunek 4a i 4b.



(a) *Wykres dokładności ACC od liczby atrybutów*

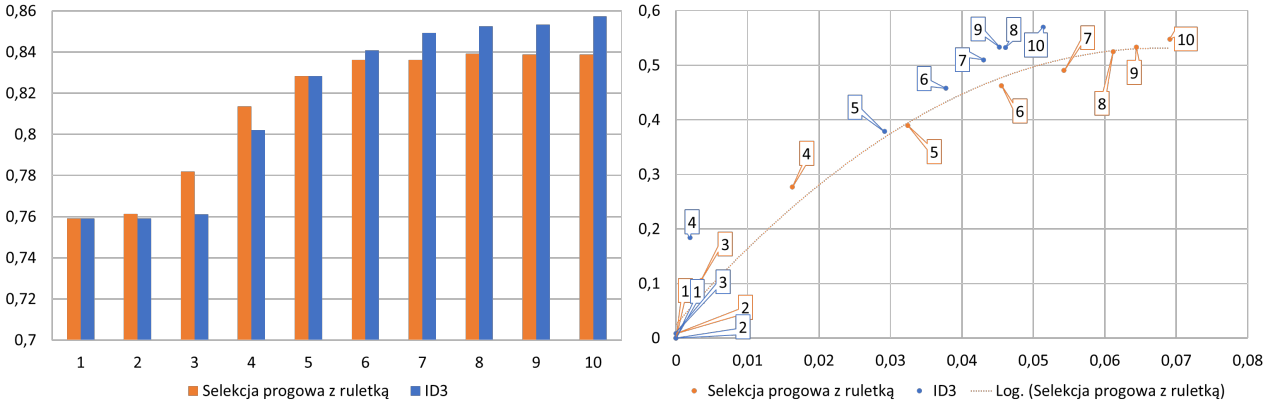
(b) *Punkty w przestrzeni ROC*

Rys. 4: *Zmiana liczby wykorzystywanych atrybutów – wyniki dla zbioru* `Wine Quality`

Oba testowane modele zachowują się bardzo podobnie. Nasze rozwiązanie cechuje się jednak nieznacznie większą dokładnością – szczególnie dla mniejszej liczby atrybutów. Dokładność obu modeli jest liniowo zależna od liczby wykorzystywanych atrybutów. Model ID3 wykazał tempo wzrostu dokładności większe ok. 1,6 razy. Nasze rozwiązanie jest jednocześnie mniej stabilne – nie udało się nam zaobserwować idealnego trendu rosnącego. Wnioskujemy, że jest to wynik zastosowania mechanizmu ruletki. Dokładność obu modeli dla testowanego zbioru jest porównywalnie dobra. Warto zaznaczyć, że przez odpowiednie kryteria stopu (i potencjalnie przez mechanizm ruletki) udało uniknąć się nam nadmiernego dopasowania do zbioru treningowego.

Analiza ROC potwierdza, że wraz ze wzrostem liczby atrybutów zwiększa się jakość modelu (zmniejsza się odległość euklidesowa do punktu (0,1)). Niewielką przewagą cechuje się model wykorzystujący selekcję progową z ruletką. Mogliśmy zauważyć, że dla obydwu modeli punkty układają się wzdłuż jednej prostej. Tabelaryczne zestawienie wyników eksperymentu znajduje się w sekcji 12.1.1.

Identyczne symulacje przeprowadziliśmy dla zbioru `Adult`. Wyniki przedstawia rysunek 5a i 5b.



(a) *Wykres dokładności ACC od liczby atrybutów*

(b) *Punkty w przestrzeni ROC*

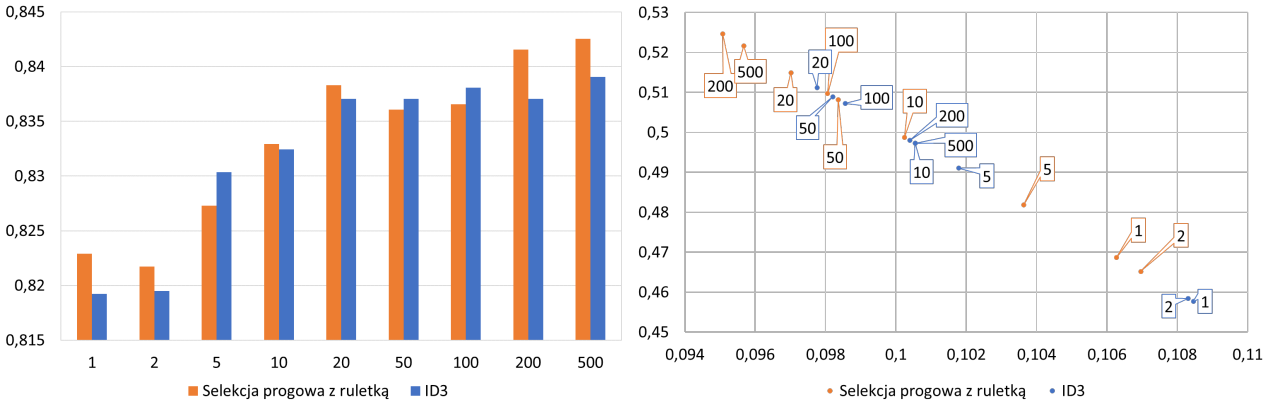
Rys. 5: *Zmiana liczby wykorzystywanych atrybutów – wyniki dla zbioru* `Adult`

Ponownie mogliśmy zaobserwować, że testowane modele zachowują się podobnie. W tym wypadku las wykorzystujący algorytm ID3 cechuje się nieznacznie większą dokładnością – przewaga zwiększa się wraz ze wzrostem atrybutów. Nasze rozwiązanie dla liczby atrybutów większej od 6 zachowało praktycznie identyczną dokładność – przypuszczalnie przez nadmierne dopasowanie. W tym wypadku nie można zauważyć liniowej zależności dokładności ACC od liczby atrybutów. Mimo wszystko dokładność jest w ogólności tym większa, im większa jest liczba wykorzystywanych atrybutów.

Analiza ROC również potwierdza spostrzeżenia dot. jakości. Dla zwiększania liczby atrybutów od 1 do 6 możemy zauważyć znaczną jej poprawę dla obu modeli. Dla liczby atrybutów większej od 6 jakość modelu wykorzystującego ID3 jest nieznacznie większa od modelu wykorzystującego selekcję progową z ruletką. Dla tego zakresu odległości od punktu (0,1) pozostają w przybliżeniu stałe. W tym wypadku punkty układają się wzdłuż krzywych logarytmicznych. Tabelaryczne zestawienie wyników eksperymentu znajduje się w sekcji 12.1.2.

9.3 Liczba modeli bazowych

Zbadaliśmy wpływ liczby modeli bazowych na jakość lasu losowego. Wyniki dla modelu wykorzystującego selekcję progową z ruletką zestawiliśmy z modelem referencyjnym (wykorzystującym ID3). Testy przeprowadziliśmy dla standardowej liczby atrybutów (pierwiastek z liczby wszystkich atrybutów) i dla następujących ustawień naszego modelu: `split_strategy='mean'`, `threshold='half'`. Po zebraniu zagregowanych wyników 20 symulacji dla zbioru `Wine Quality`, utworzyliśmy wykresy dokładności ACC oraz wykres punktów w przestrzeni ROC w zależności od liczby modeli bazowych. Wyniki dla charakterystycznych punktów pracy przedstawia rys. 6a i 6b.



(a) *Wykres dokładności ACC od liczby modeli bazowych*

(b) *Punkty w przestrzeni ROC*

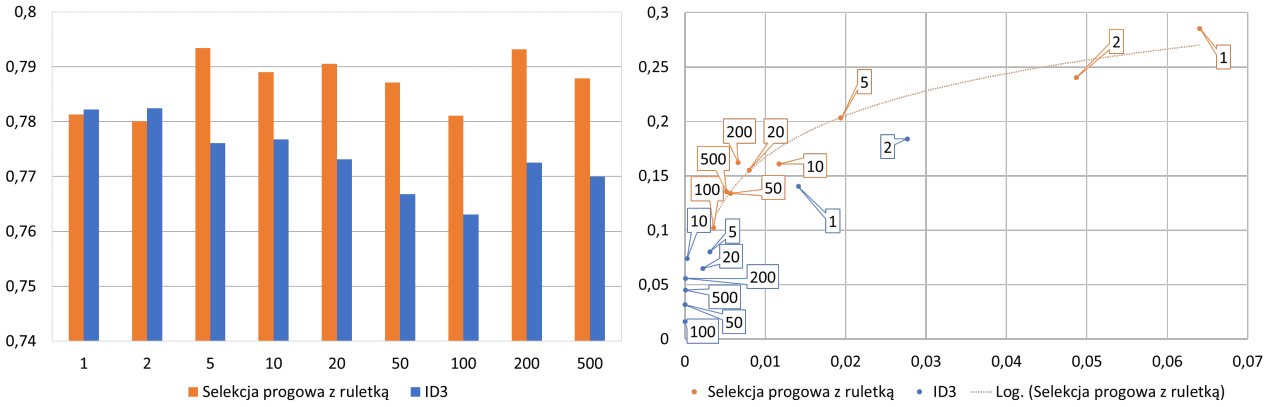
Rys. 6: *Zmiana liczby modeli bazowych – wyniki dla zbioru* `Wine Quality`

Ze względu na przyjętą skalę osi OX dokładność obu modeli stosunkowo dobrze układa się wzdłuż krzywej logarytmicznej i jest rosnąca. Wraz ze wzrostem liczby modeli bazowych dokładność obu modeli się stabilizuje. Mogliśmy to zauważyć dla liczby modeli bazowych z przedziału 200-500 i dla 500-600, dla których wyniki

były praktycznie identyczne jak te dla 200 i 500 (dla obu modeli). W odróżnieniu od modelu referencyjnego, przypuszczalnie przez wprowadzony mechanizm ruletki, nie jesteśmy w stanie zaobserwować stałego trendu rosnącego przy naszym modelu. Dla modelu referencyjnego dokładność stabilizuje się do 0,837 już dla liczby modeli bazowych równej 50. Dla modelu wykorzystującego selekcję progową z ruletką stabilizuje się do 0,842 dla około 200 drzew. Stabilizacji uległy również inne miary (TPR, TNR, ...).

Analiza ROC potwierdza, że wraz ze wzrostem liczby wykorzystywanych drzew zwiększa się jakość modelu. Niewielką przewagą cechuje się model wykorzystujący selekcję progową z ruletką. Dla obydwu modeli punkty układają się wzdłuż jednej prostej. Tabelaryczne zestawienie wyników eksperymentu znajduje się w sekcji 12.2.1.

Identyczne symulacje przeprowadziliśmy dla zbioru `Adult`. Wyniki dla charakterystycznych punktów pracy przedstawia rysunek 7a i 7b.



(a) *Wykres dokładności ACC od liczby modeli bazowych*

(b) *Punkty w przestrzeni ROC*

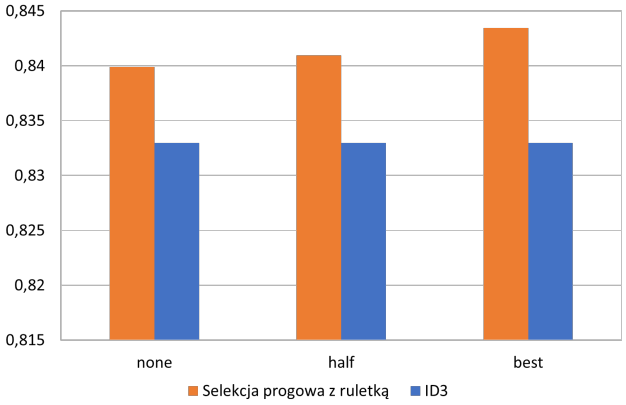
Rys. 7: *Zmiana liczby modeli bazowych – wyniki dla zbioru* `Adult`

Nasz model wykazał się nieznacznie większą dokładnością (ok. 0,02) dla zdecydowanej większości symulacji. Wyniki dla tego zbioru jednoznacznie pokazują, że dokładności modelu ulegają stabilizacji dopiero dla dużej liczby modeli bazowych (co najmniej kilkaset). W przypadku tego zbioru danych jedno drzewo decyzyjne osiąga dokładność rzędu 0,78 i manipulacja liczbą używanych modeli decyzyjnych wpływa jedynie na jej stabilizację – do wartości ok. 0,785 dla modelu wykorzystującego selekcję progową z ruletką i ok. 0,77 dla modelu referencyjnego. Ponownie mogliśmy zauważyć, że dla liczby drzew z przedziału 200-500 i dla 550 i 600 dokładność pozostawała zbliżona do dokładności dla 200 i 500 drzew (dla obu modeli). Stabilizacji uległy inne miary (TPR, TNR,...).

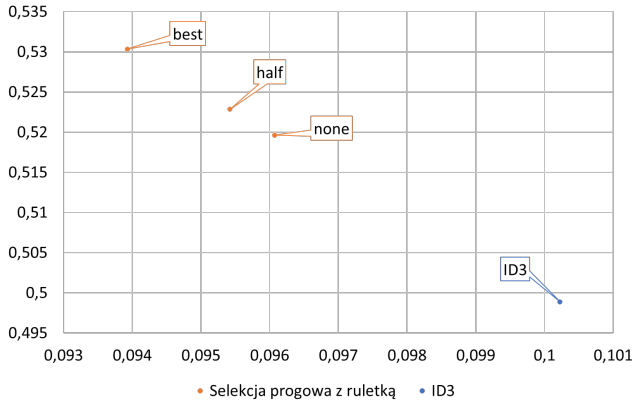
W tym przypadku zmiany dla różnej liczby modeli bazowych zdecydowanie bardziej widać przeprowadzając analizę ROC. Wraz ze wzrostem liczby modeli bazowych zmniejsza się odległość od punktu (0,1) – jakość obu modeli jednoznacznie zwiększa się. Model wykorzystujący selekcję progową z ruletką cechuje się lepszą jakością od modelu wykorzystującego ID3 dla większości przypadków testowych (porównując symulacje dla tej samej liczby modeli bazowych). Tabelaryczne zestawienie wyników eksperymentu znajduje się w sekcji 12.2.2.

9.4 Tryby selekcji progowej (`threshold`)

Zbadaliśmy wpływ trybów selekcji progowej na jakość lasu losowego. Zbadaliśmy trzy zaimplementowane przez nas tryby: "none" (odrzucenie tylko zerowego zysku informacyjnego), "half" (zachowanie połowy najlepszych podziałów), "best" (zachowanie podziału z największym zyskiem informacyjnym). Testy przeprowadziliśmy dla liczby wykorzystywanych atrybutów równej 5, liczby modeli bazowej równej 200 i dla następujących ustawień naszego modelu: `split_strategy='mean'`. Wyniki dla modelu wykorzystującego selekcję progową z ruletką zestawiliśmy z modelem referencyjnym (wykorzystującym ID3). Wyniki dla zbioru `Wine Quality` przedstawia rysunek 8a i 8b, a dla zbioru `Adult` rysunek 9a i 9b.

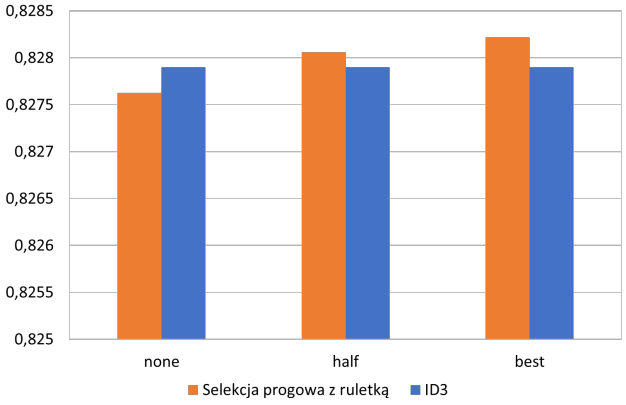


(a) *Dokładność ACC dla różnych trybów selekcji progowej*

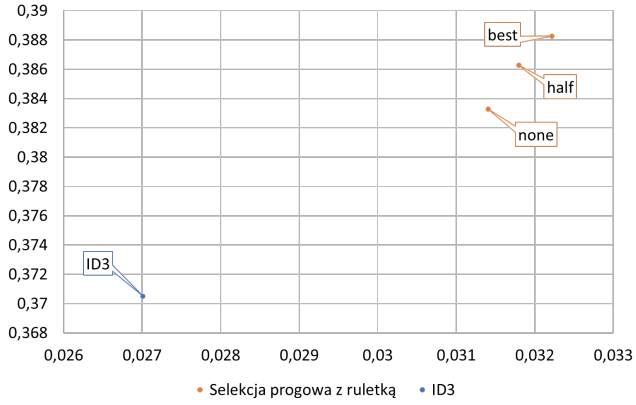


(b) *Punkty w przestrzeni ROC*

Rys. 8: *Różne tryby selekcji progowej – wyniki dla zbioru* `Wine Quality`



(a) *Dokładność ACC dla różnych trybów selekcji progowej*



(b) *Punkty w przestrzeni ROC*

Rys. 9: *Różne tryby selekcji progowej – wyniki dla zbioru* `Adult`

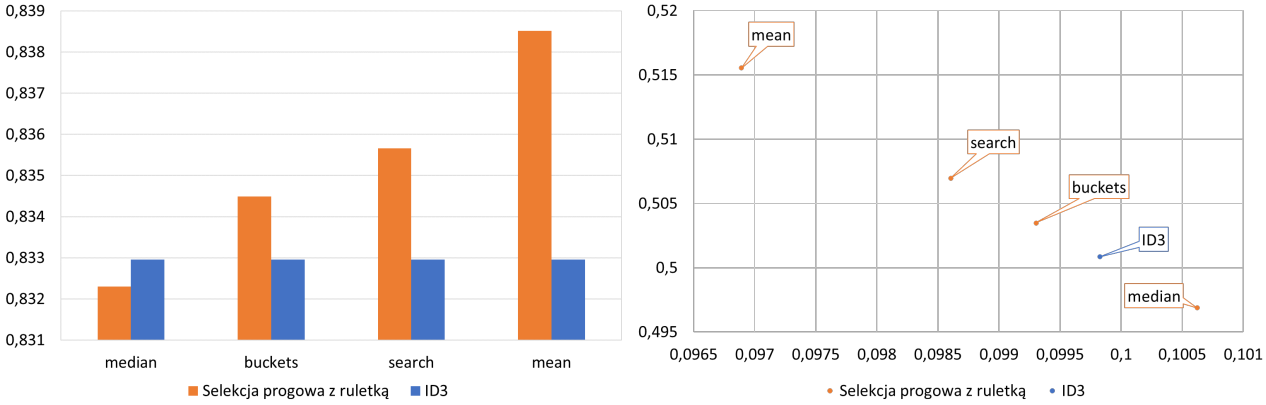
Dla obu zbiorów największą dokładnością cechuje się model wybierający najlepszy podział ("best"), potem model wybierający połowę najlepszych ("half"). Najgorszą dokładność ma model nieodrzucający żadnych podziałów ("none"). W przypadku zbioru z większą ilością atrybutów ciągłych (Wine Quality) nasz model w każdym trybie cechuje się większą dokładnością niż model referencyjny.

Analiza ROC dodatkowo potwierdza, że jakość modelu zachowuje się podobnie jak dokładność. Dla zbioru `Wine Quality` najlepszą jakością cechuje się tryb "best", potem "half" i "none". Najgorszą jakość ma model referencyjny. Dla zbioru `Adult` wszystkie modele mają zbliżoną jakość (różnice w dokładności są również nieznaczne). Nie znaleźliśmy znacznych zmian w innych metrykach (TPR, TNR,...).

Na podstawie tego eksperymentu można zauważyć, że mechanizm selekcji progowej ma największy wpływ na jakość modelu dla zbiorów, w których występuje wiele atrybutów ciągłych (Wine Quality). Tabelaryczne zestawienie wyników eksperymentu znajduje się w sekcji 12.3.1 i 12.3.2.

9.5 Tryby obsługi atrybutów ciągłych (`split_strategy`)

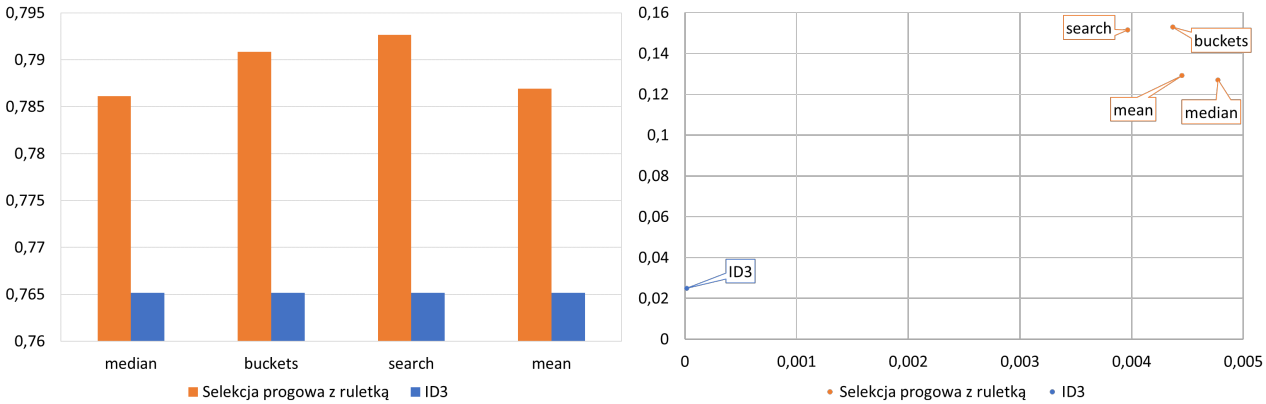
Zbadaliśmy wpływ trybów obsługi atrybutów ciągłych na jakość lasu losowego. Zbadaliśmy cztery zaimplementowane przez nas tryby: "mean" (wartość podziału – średnia), "median" (wartość podziału – mediana), "buckets" (wartość podziału wyznaczana przez kubełkowanie), "search" (liniowe poszukiwanie najlepszego podziału przez uśrednianie unikalnych wartości atrybutu). Testy przeprowadziliśmy dla liczby wykorzystywanych atrybutów równej 5, liczby modeli bazowych równej 200 i dla następujących ustawień naszego modelu: `threshold='half'`. Wyniki dla modelu wykorzystującego selekcję progową z ruletką zestawiliśmy z modelem referencyjnym (wykorzystującym ID3). Wyniki dla zbioru `Wine Quality` przedstawia rys. 8a i 8b, a dla zbioru `Adult` rys. 9a i 9b.



(a) *Dokładność ACC dla różnych trybów obsługi atrybutów ciągłych*

(b) *Punkty w przestrzeni ROC*

Rys. 10: *Różne tryby obsługi atrybutów ciągłych – wyniki dla zbioru* `Wine Quality`



(a) *Dokładność ACC dla różnych trybów obsługi atrybutów ciągłych*

(b) *Punkty w przestrzeni ROC*

Rys. 11: *Różne tryby obsługi atrybutów ciągłych – wyniki dla zbioru* `Adult`

Dla zbioru `Wine Quality` największą, choć nieznacznie odbiegającą od reszty, dokładnością cechuje się tryb "mean". Dla zbioru `Adult` największą dokładnością cechują się tryby "buckets" i "search". Każdy tryb poza "median" dla zbioru `Wine Quality` wykazał się większą dokładnością niż model referencyjny.

Wybieranie optymalnego podziału nie jest zatem tak ważne w algorytmie. Wybór trybów "buckets" i "search" ponosi za sobą duże koszty obliczeniowe i wydłużony czas dopasowania modelu w porównaniu z pozostałymi trybami. Dla trybu "mean" i "median" dopasowanie jest ok. 10 razy szybsze niż dla "buckets" i "search".

Z analizy ROC możemy wywnioskować, że jakość modelu zachowuje się podobnie jak dokładność. Dla zbioru `Wine Quality` najlepszą jakością cechuje się tryb "best", potem "half", "none" a następnie model referencyjny. Mimo wszystko dla obu zbiorów wszystkie tryby mają zbliżoną jakość (różnice w dokładności również są nieznaczne). Nie znaleźliśmy znacznych zmian w innych metrykach (TPR, TNR,...).

Wybór odpowiedniego trybu jest zatem ściśle związany z rozważanym zbiorem danych. Tabelaryczne zestawienie wyników eksperymentu znajduje się w sekcji 12.4.1 i 12.4.2.

10 Podsumowanie wyników eksperymentów

W zadaniach klasyfikacji, w których używa się lasu losowego wykorzystującego mechanizm selekcji progowej z ruletką, konieczny jest odpowiedni dobór poszczególnych parametrów:

- Zwiększając liczbę atrybutów jesteśmy w stanie poprawić jakość predykcyjną modelu. Należy jednak wziąć pod uwagę, że nadmierna ilość wybranych atrybutów może spowodować nadmierne dopasowanie.
- Zgodnie z założeniem metodyki baggingu zwiększenie liczby modeli bazowych wpływa pozytywnie na stabilność rozważanego modelu. Analizując lasy, w których było 200 lub więcej modeli bazowych, zauważyliśmy ustabilizowanie się ich dokładności i innych metryk (TPR, TNR, ...).

- Tryby selekcji progowej nie miały dużego wpływu na jakość predykcyjną modelu. Oczywistym jest, że zostawiając więcej podziałów do potencjalnego wyboru w procesie ruletki, jakość modelu obniży się. Z kolei zostawiając najlepszy podział, jakość maksymalizuje się. Jeśli chcemy, aby model nie dopasowywał się nadmiernie do danych, to sugerowanym trybem jest tryb "half" lub "none".
- Wybór trybu obsługi atrybutów ciągłych jest ściśle związany z rozważanym zbiorem danych i modelem. Dzięki wykorzystaniu mechanizmu ruletki nie jest konieczne szukanie maksymalnego zysku informacyjnego dla podziału wobec danego atrybutu. Dzięki temu możemy wybrać mniej obciążające kryteria wyboru ("mean", "median") i zmniejszyć czas dopasowywania modelu. Wybór najlepszego zysku informacyjnego może mieć największe znaczenie, gdy zyski informacyjne dla kilku podziałów są dla siebie bardzo zbliżone (kolumny są podobnie skorelowane z klasą przykładu).

Zgodnie z przypuszczeniami sformułowanymi w dokumentacji wstępnej, nasz model najlepiej sprawdza się dla danych, w których występuje dużo atrybutów ciągłych. Dla zbioru `Wine Quality` (gdzie występują jedynie atrybuty ciągłe) nasz model wykazał się lepszą jakością predykcji niż model referencyjny (ID3) w większości przeprowadzonych testów. Działając na zbiorze `Adult` (posiadającym atrybuty dyskretne) konieczne jest odpowiednie skalibrowanie naszego modelu. Wybierając odpowiednie opcje jesteśmy w stanie uzyskać jakość zbliżoną lub przewyższającą model referencyjny (ID3). Zastosowane kryteria stopu i mechanizm selekcji progowej z ruletką zapobiegł nadmiernemu dopasowaniu do zbioru treningowego.

11 Podsumowanie projektu

Zadanie projektowe uważamy za niezwykle ciekawe. Bardzo przyjemnie pracowało się przy samym algorytmie. Przydatna okazała się biblioteka `sklearn`, dzięki której mogliśmy cały czas śledzić, czy wprowadzone przez nas zmiany w algorytmie nie wprowadzają błędów (logicznych, obliczeniowych).

Wyniki dotyczące trybów selekcji atrybutów ciągłych nas zaskoczyły. Po analizie zebranych wyników jesteśmy w stanie stwierdzić w jakich sytuacjach nasz model może zostać wykorzystany. Najlepiej sprawdzi się dla zbiorów danych, w których występuje wiele atrybutów ciągłych.

Nauczyliśmy się jakie modele wybierać, gdy mamy do klasyfikacji konkretne typy zbiorów danych (atrybuty ciągłe, dyskretne). Z pewnością zwiększył się nasz poziom wiedzy w zakresie drzew decyzyjnych i lasów losowych.

Najbardziej czasochłonną częścią projektu było zebranie wyników eksperymentów. Dla najbardziej obciążających obliczeniowo ustawień naszego algorytmu (maksymalna liczba atrybutów, szukanie największego zysku informacyjnego, duża liczba drzew) wykonanie programu potrafiło zająć kilka godzin. Mimo wszystko uważamy, że wyniki i zdobyta wiedza były tego warte.

12 Załączniki – wyniki poszczególnych testów

12.1 Liczba wykorzystywanych atrybutów

12.1.1 Zbiór `Wine Quality`

Liczba atrybutów	1	2	3	4	5	6	7	8	9	10
TPR	0,51	0,51	0,52	0,53	0,52	0,53	0,53	0,54	0,54	0,55
TNR	0,90	0,90	0,90	0,91	0,90	0,91	0,91	0,91	0,91	0,91
PPV	0,51	0,51	0,52	0,53	0,52	0,53	0,53	0,54	0,54	0,55
ACC	**0,83**	0,83	0,84	0,84	0,84	0,84	0,84	0,85	0,85	0,85
Micro F1	0,51	0,51	0,52	0,53	0,52	0,53	0,53	0,54	0,54	0,55
Macro F1	0,0013	0,0013	0,0013	0,0014	0,0015	0,0016	0,0016	0,0017	0,0017	0,0018
Weighted F1	0,42	0,44	0,44	0,45	0,46	0,48	0,47	0,49	0,49	0,50
FPR	0,10	0,10	0,10	0,09	0,10	0,09	0,09	0,09	0,09	0,09

Tabela 1: *Wyniki dla modelu wykorzystującego selekcję progową z ruletką*

Liczba atrybutów	1	2	3	4	5	6	7	8	9	10
TPR	0,47	0,50	0,50	0,51	0,52	0,52	0,53	0,53	0,54	0,54
TNR	0,89	0,90	0,90	0,90	0,90	0,90	0,91	0,91	0,91	0,91
PPV	0,47	0,50	0,50	0,51	0,52	0,52	0,53	0,53	0,54	0,54
ACC	**0,83**	0,83	0,83	0,84	0,84	0,84	0,84	0,84	0,85	**0,85**
Micro F1	0,47	0,50	0,50	0,51	0,52	0,52	0,53	0,53	0,54	0,54
Macro F1	0,0011	0,0013	0,0013	0,0013	0,0015	0,0016	0,0017	0,0017	0,0018	0,0019
Weighted F1	0,37	0,43	0,43	0,44	0,45	0,47	0,48	0,49	0,49	0,50
FPR	0,11	0,10	0,10	0,10	0,10	0,10	0,094	0,093	0,092	0,093

Tabela 2: *Wyniki dla modelu referencyjnego wykorzystującego ID3*

12.1.2 Zbiór `Adult`

Liczba atrybutów	1	2	3	4	5	6	7	8	9	10
TPR	0,00	0,01	0,11	0,28	0,39	0,46	0,49	0,52	0,53	0,55
TNR	1,00	1,00	1,00	0,98	0,97	0,95	0,95	0,94	0,94	0,93
PPV	0,00	0,97	0,90	0,84	0,79	0,76	0,74	0,73	0,72	0,72
ACC	**0,76**	0,76	0,77	0,81	0,83	0,84	0,84	0,84	0,84	**0,84**
F1	0,00	0,02	0,19	0,42	0,52	0,58	0,59	0,61	0,61	0,62
FPR	0,0	0,0	0,0036	0,016	0,032	0,046	0,054	0,061	0,064	0,069

Tabela 3: *Wyniki dla modelu wykorzystującego selekcję progową z ruletką*

Liczba atrybutów	1	2	3	4	5	6	7	8	9	10
TPR	0,00	0,00	0,01	0,18	0,38	0,46	0,51	0,53	0,53	0,57
TNR	1,00	1,00	1,00	1,00	0,97	0,96	0,96	0,95	0,95	0,95
PPV	0,00	0,00	1,00	0,97	0,80	0,79	0,79	0,79	0,79	0,78
ACC	**0,76**	0,76	0,76	0,80	0,83	0,84	0,85	0,85	0,85	**0,86**
F1	0,00	0,00	0,02	0,31	0,51	0,58	0,62	0,63	0,64	0,66
FPR	0,00	0,00	0,00	0,00	0,03	0,04	0,04	0,05	0,05	0,05

Tabela 4: *Wyniki dla modelu referencyjnego wykorzystującego ID3*

12.2 Liczb wykorzystywanych modeli bazowych

12.2.1 Zbiór `Wine Quality`

Liczba modeli bazowych	1	2	5	10	20	50	100	200	500
TPR	0,47	0,47	0,48	0,50	0,51	0,51	0,51	0,52	0,52
TNR	0,89	0,89	0,90	0,90	0,90	0,90	0,90	0,90	0,90
PPV	0,47	0,47	0,48	0,50	0,51	0,51	0,51	0,52	0,52
ACC	**0,82**	0,82	0,83	0,83	**0,84**	0,84	0,84	0,84	**0,84**
Micro F1	0,47	0,47	0,48	0,50	0,51	0,51	0,51	0,52	0,52
Macro F1	0,0014	0,0013	0,0012	0,0013	0,0013	0,0013	0,0013	0,0014	0,0013
Weighted F1	0,42	0,42	0,41	0,43	0,44	0,43	0,44	0,45	0,45
FPR	0,11	0,11	0,10	0,10	0,10	0,10	0,10	0,10	0,10

Tabela 5: *Wyniki dla modelu wykorzystującego selekcję progową z ruletką*

Liczba modeli bazowych	1	2	5	10	20	50	100	200	500
TPR	0,46	0,46	0,49	0,50	0,51	0,51	0,51	0,50	0,50
TNR	0,89	0,89	0,90	0,90	0,90	0,90	0,90	0,90	0,90
PPV	0,46	0,46	0,49	0,50	0,51	0,51	0,51	0,50	0,50
ACC	**0,82**	0,82	0,83	0,83	**0,84**	0,84	0,84	0,84	**0,84**
Micro F1	0,46	0,46	0,49	0,50	0,51	0,51	0,51	0,50	0,50
Macro F1	0,0013	0,0013	0,0013	0,0013	0,0013	0,0013	0,0013	0,0013	0,0013
Weighted F1	0,41	0,41	0,43	0,43	0,44	0,44	0,44	0,43	0,42
FPR	0,11	0,11	0,10	0,10	0,10	0,10	0,10	0,10	0,10

Tabela 6: *Wyniki dla modelu referencyjnego wykorzystującego ID3*

12.2.2 Zbiór `Adult`

Liczba modeli bazowych	1	2	5	10	20	50	100	200	500
TPR	0,29	0,24	0,20	0,16	0,16	0,13	0,10	0,16	0,14
TNR	0,94	0,95	0,98	0,99	0,99	0,99	1,00	0,99	0,99
PPV	0,58	0,61	0,78	0,83	0,87	0,88	0,90	0,89	0,89
ACC	**0,78**	0,78	0,79	0,79	0,79	0,79	**0,78**	0,79	**0,79**
F1	0,36	0,33	0,31	0,26	0,26	0,23	0,18	0,27	0,24
FPR	0,06	0,049	0,019	0,012	0,0080	0,0057	0,00	0,0066	0,0052

Tabela 7: *Wyniki dla modelu wykorzystującego selekcję progową z ruletką*

Liczba modeli bazowych	1	2	5	10	20	50	100	200	500
TPR	0,14	0,18	0,08	0,07	0,06	0,03	0,02	0,06	0,05
TNR	0,99	0,97	1,00	1,00	1,00	1,00	1,00	1,00	1,00
PPV	0,75	0,77	0,74	0,99	0,95	1,00	1,00	1,00	0,99
ACC	**0,78**	0,78	0,78	0,78	0,77	0,77	**0,76**	0,77	**0,77**
F1	0,23	0,28	0,14	0,13	0,12	0,06	0,03	0,11	0,09
FPR	0,014	0,028	0,0031	0,0003	0,0022	0,0	0,0	0,0	0,0

Tabela 8: *Wyniki dla modelu referencyjnego wykorzystującego ID3*

12.3 Tryby selekcji progowej

12.3.1 Zbiór `Wine Quality`

Tryb	none	half	best
TPR	0,52	0,52	0,53
TNR	0,90	0,90	0,91
PPV	0,52	0,52	0,53
ACC	0,84	0,84	0,84
Micro F1	0,52	0,52	0,53
Macro F1	0,00	0,00	0,00
Weighted F1	0,46	0,46	0,47
FPR	0,10	0,10	0,09

Tabela 9: *Wyniki dla modelu wykorzystującego selekcję progową z ruletką*

Model	ID3
TPR	0,50
TNR	0,90
PPV	0,50
ACC	0,83
Micro F1	0,50
Macro F1	0,0013
Weighted F1	0,43
fpr	0,10

Tabela 10: *Wyniki dla modelu referencyjnego wykorzystującego ID3*

12.3.2 Zbiór `Adult`

Tryb	none	half	best
TPR	0,38	0,39	0,39
TNR	0,97	0,97	0,97
PPV	0,80	0,80	0,79
ACC	0,83	0,83	0,83
F1	0,52	0,52	0,52
FPR	0,031	0,032	0,032

Tabela 11: *Wyniki dla modelu wykorzystującego selekcję progową z ruletką*

Model	ID3
TPR	0,37
TNR	0,97
PPV	0,81
ACC	0,83
F1	0,51
FPR	0,027

Tabela 12: *Wyniki dla modelu referencyjnego wykorzystującego ID3*

12.4 Tryby obsługi atrybutów ciągłych

12.4.1 Zbiór `Wine Quality`

Tryb	median	buckets	search	mean
TPR	0,50	0,50	0,51	0,52
TNR	0,90	0,90	0,90	0,90
PPV	0,50	0,50	0,51	0,52
ACC	0,83	0,83	0,84	0,84
Micro F1	0,50	0,50	0,51	0,52
Macro F1	0,0013	0,0013	0,0013	0,0013
Weighted F1	0,42	0,43	0,43	0,44
FPR	0,10	0,10	0,10	0,10

Tabela 13: *Wyniki dla modelu wykorzystującego selekcję progową z ruletką*

Model	ID3
TPR	0,50
TNR	0,90
PPV	0,50
ACC	0,83
Micro F1	0,50
Macro F1	0,0013
Weighted F1	0,43
FPR	0,10

Tabela 14: *Wyniki dla modelu referencyjnego wykorzystującego ID3*

12.4.2 Zbiór `Adult`

Tryb	median	buckets	search	mean
TPR	0,13	0,15	0,15	0,13
TNR	1,00	0,99	1,00	1,00
PPV	0,89	0,89	0,93	0,90
ACC	0,79	0,79	0,79	0,79
F1	0,22	0,26	0,26	0,23
FPR	0,0048	0,0064	0,0040	0,0044

Tabela 15: *Wyniki dla modelu wykorzystującego selekcję progową z ruletką*

Model	ID3
TPR	0,025
TNR	1,00
PPV	1,00
ACC	0,77
F1	0,048
FPR	1,60E-05

Tabela 16: *Wyniki dla modelu referencyjnego wykorzystującego ID3*

Bibliografia

[1] Bohumil Stádník, Jurgita Raudeliuniene, and Vida Davidavičienė. “Fourier Analysis for Stock Price Forecasting: Assumption and Evidence”. In: *Journal of Business Economics and Management* 17 (May 2016), pp. 365–380. DOI: 10.3846/16111699.2016.1184180.

[2] Luca (https://dsp.stackexchange.com/users/27269/luca). “The exact definition of dominant frequency?” In: (). https://dsp.stackexchange.com/questions/40180/the-exact-definition-of-dominant-frequency (version: 2017-04-12). eprint: https://dsp.stackexchange.com/questions/40180/the-exact-definition-of-dominant-frequency. URL: https://dsp.stackexchange.com/questions/40180/the-exact-definition-of-dominant-frequency.

[3] Mridula Batra and Rashmi Agrawal. “Comparative Analysis of Decision Tree Algorithms”. In: *Nature Inspired Computing*. Ed. by Bijaya Ketan Panigrahi et al. Singapore: Springer Singapore, 2018, pp. 31–36. ISBN: 978-981-10-6747-1.

[4] Leo Breiman. “Bagging predictors”. In: *Machine learning* 24 (1996), pp. 123–140.

[5] Barry De Ville. “Decision trees”. In: *Wiley Interdisciplinary Reviews: Computational Statistics* 5.6 (2013), pp. 448–455.

[6] Nicha Kaewrod and Kietikul Jearanaitanakij. “Improving ID3 Algorithm by Ignoring Minor Instances”. In: *2018 22nd International Computer Science and Engineering Conference (ICSEC)*. 2018, p. 2. DOI: 10.1109/ICSEC.2018.8712762.

[7] J Ross Quinlan. “Improved use of continuous attributes in C4. 5”. In: *Journal of artificial intelligence research* 4 (1996), p. 78.

[8] Salvatore Ruggieri. “Efficient C4. 5 [classification algorithm]”. In: *IEEE transactions on knowledge and data engineering* 14.2 (2002), p. 438.

[9] Leo Breiman. “Random forests”. In: *Machine learning* 45 (2001), pp. 5–6.

[10] César Ferri, Peter Flach, and José Hernández-Orallo. “Learning decision trees using the area under the ROC curve”. In: *Icml*. Vol. 2. 2002, p. 142.

[11] Juri Opitz and Sebastian Burst. “Macro f1 and macro f1”. In: *arXiv preprint arXiv:1911.03347* (2019).