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.
Nasza interpretacja treści zadania:
Naszym celem jest napisanie programu, który na podstawie zadanego zbioru danych testowych tworzy las losowy drzew decyzyjnych. Następnie, z jego wykorzystaniem, jest w stanie klasyfikować podany przykład. Ruletkę zaimplementujemy na poziomie doboru warunku dla kolejnych węzłów drzew.

2 Wstęp teoretyczny

2.1 Drzewa decyzyjne

Struktura danych złożona z węzłów, połączonych krawędziami (gałęzie), prowadzącymi do kolejnych poziomów drzewa - bezpośrednio do liści, w taki sposób że dwa dowolne węzły połączone są tylko jedną ścieżką. Macierzysty węzeł (bez rodzica) jest korzeniem, natomiast ostatni poziom węzłów nazywany jest liśćmi. W kontekście algorytmów drzewa można interpretować w następujący sposób: węzły - testy przeprowadzone na wartościach atrybutów przykładów, gałęzie - możliwe wyniki przeprowadzonych testów, liście - etykiety kategorii. Dzięki temu pozwalają klasyfikować przykłady o zadanych atrybutach. Pojedyńcze drzewa są podatne na nadmierne dopasowanie, dlatego są bardzo dobrym przykładem do wykorzystania lasów losowych. Dodatkowo korzystne jest stosowanie warunków ograniczających tj. maksymalna głębokość drzewa (liczba poziomów drzewa). [1]

2.2 Lasy losowe

Stanowią kolekcję modeli, w tym przypadku utworzonych drzew decyzyjnych, i wykorzystywane są w koncepcji **Ensemble learning**. Takie podejście pozwala uzyskać zamiast kilku klasyfikatorów, jeden dokładniejszy (czyli minimalizuje niedokładność pojedyńczych modeli). W celu utworzenia lasu wykorzystywana jest metoda *bagging'u*.

2.2.1 Bagging

Metoda generuje nowe modele drzew decyzyjnych, które następnie agreguje, aby uzyskać jeden klasyfikator. Pierwszym jej etapem jest *Bootstraping*. Generowane są nowe zestawy danych (z możliwymi powtórzeniami) bazując na danych trenujących. Następnie na podstawie tak utworzonych zbiorów uczone są nowe modele (nowe drzewa decyzyjne dla T_i zestawu, uczony jest M_i model). Końcowym etapem jest *aggregation*, gdzie w celu sklasyfikowania zadanego przykładu wyniki z wszystkich modeli w lasie są 'agregowane' i finalnie przypisywana przykładowi jest najliczniejsza klasa. [2]

2.2.2 Gini index

Wskaźnik używany przy budowie drzew decyzyjnych, pozwala na obliczenie jakości, a więc na wybranie optymalnego warunku, sprawdzanego w kolejnym węźle modelu po podziele. Wyraża się wzorem:

$$G = 1 - \sum_{i=1}^{n} p_i^2 \tag{1}$$

gdzie p_i oznacza prawdopodobieństwo wybrania elementuz $i - tej$ klasy. W tym celu można wykorzystać też entropię, jednak użyjemy wyżej opisanego index'u z uwagi na lepszą optymalizację - uproszczone obliczenia (nie zawiera logarytmów jak entropia). [3] W dalszej części posługujemy się także określeniem całkowitej wartości wskaźnika Gini:

$$Gw = \frac{a}{a+b} \cdot G_1 + \frac{b}{a+b} \cdot G_0$$

gdzie: Gw oznacza całkowitą wartość wskaźnika Gini, a oznacza liczbę elementów spełniających dany warunek, b oznacza liczbę elementów niespełniających danego warunku, G_1 oznacza wskaźnik Gini dla podzbioru spełniającego dany warunek, G_0 oznacza wskaźnik Gini dla podzbioru niespełniającego dany warunek. Posługujemy się także terminem zmiany wskaźnika Gini:

$$\Delta G = G_r - Gw$$

gdzie: ΔG oznacza zmianę wskaźnika Gini, G_r oznacza wskaźnik Gini dla całego zbioru danych, Gw oznacza całkowitą wartość wskaźnika Gini dla danego wyboru.

3 Implementacja

3.1 Planowana implementacja - narzędzia

Poniżej opisaną implementację alogrytmu chcielibyśmy wykonać w języku Python, z wykorzystaniem bibliotek do manipulacji danymi i ich prezentacji, tj. NumPy, Pandas, Matplotlib/Seaborn.
Kod będziemy przechowywać/wersjonować na wydziałowym GitLab'ie. Wyniki naszej pracy porównamy z rozwiązaniem przykładowym dla algorytmu CART wybierającego najlepsze atrybutu z biblioteki $scikit - learn\_1.3.2$.

3.2 Drzewa decyzyjne

Po przeanalizowaniu używanych algorytmów drzew decyzyjnych zdecydowaliśmy się oprzeć nasze rozwiązanie na popularnym algorytmie CART. Oryginalna implementacja algorytmu zakłada podział danych testowych na węźle decyzyjnym na dwa podzbiory wedle warunku wskazanego przez wskaźnik Gini. Ze wszystkich wartości wskaźnika obliczanej dla każdego atrybuty wybierany jest ten z najmniejszą wartością. W naszym wariancie algorytmu, dla każdego atrybutu rozdzielamy nasz zbiór testowy na dwa podzbiory wedle warunku utworzonego na podstawie wartości atrybutu, następnie dla każdego takie podziały, wyliczana jest wartość wskaźnika Gini, kolejno wyliczana jest wartość całkowitej wartości wskaźnika Gini poprzez sumę iloczynów wag wskaźnika Gini oraz wskaźnika Gini. Po tym kroku usuwane są te warunki, których wskaźnik Gini znajduje się ponad progiem, a z pozostałych warunków losowo wybierany (ruletka) jest ostateczny warunek z prawdopodobieństwem równym proporcji wartości różnicy wskaźnika Gini dla całego zbioru oraz wskaźnika Gini dla danego wyboru. Następnie, wedle tego wyboru dane są rozdzielane na dwa podzbiory, spełniające dany warunek i niespełniające. Kolejno, przekazywane są rekurencyjnie do następnych węzłów w celu utworzenia drzewa. W procesie tworzenia drzewa decyzyjnego istnieje kilka warunków zakończenia algorytmu: przekroczenie przez wartość głębokości maksymalnej wartości głębokości drzewa, osiągnięcie przez liczność podzbioru minimalnej wartości liczby elementów, utworzenie przez wybór podzbioru pustego. Wystąpienie, któregokolwiek z tych warunków, powoduje zakończenie dalszej generacji węzłów decyzyjnych i ustalenie tego węzła jako liść z etykietą wybieraną większościowo z etykiet w zbiorze, lub losowo w momencie braku możliwości wyboru większościowego. Kolejnym warunkiem zakończenia procesu generacji węzłów decyzyjnych jest utworzenie podzbioru danych z tylko jedną etykietą, w tym przypadku węzeł przerabiany jest na liść z wartością etykiety równą wartości jedynej występującej klasy w podzbiorze.

Algorithm 1 Pseudokod budowy drzewa decyzyjnego

```
1: function CREATEDECISIONTREE(T,X,d) return root
2:     decisionTreeNode = new Node
3:     if all elements of T are of some class c then
4:         Set decisionTreeNode value = c
5:         return decisionTreeNode
6:     else if number of elements of T < min_elements OR d > max_depth then
7:         Set DecisionTreeNode value = selectBestClass(T)
8:         return decisionTreeNode
9:     end if
10:    dataBaseIndex = calculateGeneralGiniIndex(T)
11:    for all elementType in X do
12:        for all element in elementType do
13:            temp = calculateGiniWeightedIndex(element,T)
14:            indexChangeCollection = dataBaseIndex - temp
15:        end for
16:    end for
17:    dataBaseIndexPruned = threshold(indexChangeCollection)
18:    if dataBaseIndexPruned is empty then
19:        Set decisionTreeNode value = selectBestClass(T)
20:    end if
21:    selection = rouletteSelection(dataBaseIndexPruned)
22:    for all element value x of selection type do
23:        if x = selection then
24:            T_1 add x
25:        else
26:            T_0 add x
27:        end if
28:    end for
29:    if T_1 is empty OR T_0 is empty then
30:        Set DecisionTreeNode value = selectBestClass(T)
31:        return decisionTreeNode
32:    end if
33:    Set decisionTreeNode leftNode = createDecisionTree(T_1,X,d+1)
34:    Set decisionTreeNode rightNode = createDecisionTree(T_0,X,d+1)
35:    return decisionTreeNode
36: end function
```

gdzie:

- T - zestaw danych, X - zbiór atrybutów, d - depth (liczba poziomów drzewa)
- selectBestClass(T): Wybiera najczęściej występującą klasę w zbiorze, jeżeli nastąpi remis wybiera losowo
- calculateGeneralGiniIndex(T) - oblicza Gini index dla całego zbioru danych
- calculateGiniWeightedIndex(element,T): Oblicza Gini weighted Index dla każdego elementu, ustalonego typu
- indexChangeCollection - zbiór różnic bieżącego indexu Gini z wartością ogólną
- dataBaseIndexPruned - zbiór wyboru po zastosowaniu selekcji progowej
- rouletteSelection() - funkcja wybierająca warunek z użyciem ruletki
- selection - wybrany warunek za pomocą rouletteSelection()
- T_1, T_0 - podzbiory danych odpowiednio spełniajace i nie spełniające danego warunku węzła

Atrybuty mogą być dyskretne lub ciągłe, wpływa to na sposób dla nich wartości gini index'u. W przypadku dyskretnych podejście jest jedno, ale dla atrybutów ciągłych spotkaliśmy się z kilkoma sposobami tj. liczenie średnich lub mediany. Zdecydowaliśmy się na opcję ze średnimi, dwóch kolejnych wartości, wykorzystywaną w algorytmie C4.5.

3.3 Las losowy

Pierwszym etapem budowy lasu losowego jest utworzenie N modeli. Dla każdego drzewa, tworzymy osobny zbiór treningowy (bootstrapping) oraz wybieramy losowe atrybuty i ich liczbę. Warty zaznaczenia jest fakt, że wszelkie losowania odbywają się ze zwracaniem, tzn. w drzewie możemy mieć np. kilka warunków z wykorzystaniem tego samego atrybutu. Następnie, gdy mamy stworzony las, każdy przykład ze zbioru K klasyfikujemy za pomocą wszystkich drzew. Finalną klasą przypisywaną do przykładu k, jest etykieta najczęsciej występująca w zbiorze po agregacji odpowiedzi ze wszystkich modeli.

Algorithm 2 Pseudokod budowy lasu losowego

```
1:  function RANDOMFOREST(T, N, K) return finalClasses
2:      listOfModels = empty array
3:      for i in 1...N do
4:          T_i = bootstrap(T)
5:          choosenAtributes = random(T_i, m)
6:          add createDecisionTree(T_i, choosenAtributes, 0) to listOfModels
7:      end for
8:      finalClasses = empty array
9:      for all k in K do
10:         classes = empty array
11:         for all model in listOfModels do
12:             class <- classification of k of one model
13:             add class value to classes
14:         end for
15:         finalClass = getTheMostCommonValue(classes)
16:         add finalClass to finalClasses
17:     end for
18:     return finalClasses
19: end function
```

gdzie:

- T - zbiór trenujący, N - liczba drzew do utworzenia, K - przykłady do sklasyfikowania
- listOfModels - struktura danych przechowująca wszystkie utworzone modele w ramach lasu
- bootstrap(T) - metoda tworząca nowy zestaw danych w każdej iteracji, na podstawie wejściowego zestawu danych - z możliwością powtarzania
- m - liczba atrybutów do wylosowania
- choosenAtributes - losowo wybrane atrybuty ze zbioru T_i przez funkcję random(), przekazane jako parametr do utworzenia drzewa
- classes - sturktura danych przechowująca wszystkie etykiety klas, które wystąpiły
- getTheMostCommonValue() - funkcja agregująca te same wartości i zwracająca liczbę wystąpień najczęstszej. W przypadku takiej samej wartości, finalna wybierana jest w sposób losowy.
- finalClasses - zwraca wszystkie klasy wyznaczone przez algorytm

3.4 Przykład obliczeniowy

Najlepszym opisem sposobu działania algorytmu budowy lasu losowego, będzie prześledzenie procesu budowy jednego z jego drzew decyzyjnych. Za zbiór danych treningowych posłuży nam lekko zmodyfikowany zbiór Zbiór danych, zamienione w nim zostały wartości temperatury z wartości dyskretnych na wartości ciągłe.
Atrybuty:

- Outlook $\in$ { Sunny, Overcast, Rain }
- Temperature $\in$ [5,20]
- Humidity $\in$ { High, Normal }

- Windy $\in$ { Weak, Strong }

Etykiety:

- Play $\in$ { Yes, No }

Nr.	Outlook	Temperature	Humidity	Windy	Play
1	Sunny	19	High	Weak	No
2	Sunny	15	High	Strong	No
3	Overcast	11	High	Weak	Yes
4	Rain	9	High	Weak	Yes
5	Rain	7	Normal	Weak	Yes

Idąc zgodnie z założeniami algorytmu, początkowo losowo wybieramy $\sqrt{4} = 2$ atrybutów dla danego drzewa. Załóżmy, że wybrane zostały atrybuty: outlook oraz temperature.Najpierw wyliczmy wskaźnik Gini dla całego zbioru danych

$$G(T) = 1 - (\frac{2}{5})^2 - (\frac{3}{5})^2 = 0,48$$

Wyliczamy teraz całkowity wskaźnik Gini, który jest średnią ważoną ilości elementów zbioru danych oraz wskaźnika Gini, dla każdej z dyskretnych wartości atrybutu Outlook. Jako że jest to algorytm drzewa binarnego, przyjmujemy prosty warunek, albo wartość jest równa wartości argumentu, albo nie (Sunny == Sunny, Rain != Sunny):

$$G(Sunny, T) = (1 - (\frac{2}{2})^2) \cdot \frac{2}{5} + (1 - (\frac{3}{3})^2) \cdot \frac{3}{5} = 0,00$$
$$G(Overcast, T) = (1 - (\frac{1}{1})^2) \cdot \frac{1}{5} + (1 - (\frac{2}{4})^2 - (\frac{2}{4})^2) \cdot \frac{4}{5} = 0,40$$
$$G(Rain, T) = (1 - (\frac{2}{2})^2) \cdot \frac{2}{5} + (1 - (\frac{2}{3})^2 - (\frac{1}{3})^2) \cdot \frac{3}{5} = 0,27$$

Następnie dla atrybutu o wartości ciągłej sortujemy dane rosnąco: 7, 9, 11, 15, 19 oraz wyliczamy średnie arytmetyczne dwóch kolejnych temperatur $t_1 = 8, t_2 = 10, t_3 = 13, t_4 = 17$ (jest to przykład jednego ze sposobów podejścia do tego typu danych, może być to jeden z kilku trybów pracy algorytmu) i następnie dla każdej z wartości t_i wyliczamy całkowity wskaźnik Gini:

$$G(temperature < t_1, T) = (1 - (\frac{1}{1})^2) \cdot \frac{1}{5} + (1 - (\frac{2}{4})^2 - (\frac{2}{4})^2) \cdot \frac{4}{5} = 0,40$$
$$G(temperature < t_2, T) = (1 - (\frac{2}{2})^2) \cdot \frac{2}{5} + (1 - (\frac{1}{3})^2 - (\frac{2}{3})^2) \cdot \frac{3}{5} = 0,27$$
$$G(temperature < t_3, T) = (1 - (\frac{3}{3})^2) \cdot \frac{3}{5} + (1 - (\frac{2}{2})^2) \cdot \frac{2}{5} = 0,00$$
$$G(temperature < t_4, T) = (1 - (\frac{1}{4})^2 - (\frac{3}{4})^2) \cdot \frac{4}{5} + (1 - (\frac{1}{1})^2) \cdot \frac{1}{5} = 0,30$$

Następnie wyliczamy zmianę wskaźnika Gini dla każdego warunku:

$$\Delta G(Sunny, T) = 0,48 - 0,00 = 0,48$$
$$\Delta G(Overcast, T) = 0,48 - 0,40 = 0,08$$
$$\Delta G(Rain, T) = 0,48 - 0,27 = 0,21$$
$$\Delta G(temperature < t_1, T) = 0,48 - 0,40 = 0,08$$
$$\Delta G(temperature < t_2, T) = 0,48 - 0,27 = 0,21$$
$$\Delta G(temperature < t_3, T) = 0,48 - 0,00 = 0,48$$
$$\Delta G(temperature < t_4, T) = 0,48 - 0,30 = 0,18$$

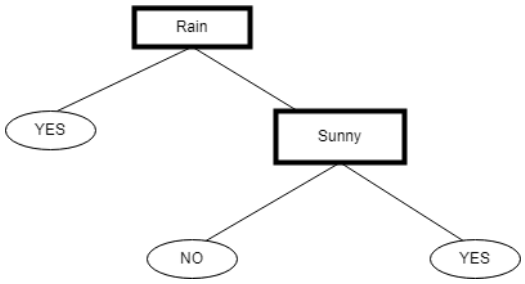
W tym momencie zachodzi selekcja wedle ustawionego progu np. usunięcie najgorszych 40% warunków, a następnie na zasadzie ruletki wybór ostatecznego warunku. Przykładowy wzór na wyliczenie prawdopodobieństwa wyboru danego warunku:

$$p(x) = \frac{\Delta G(x, T)}{\sum_{y \in X} \Delta G(y, T)}$$

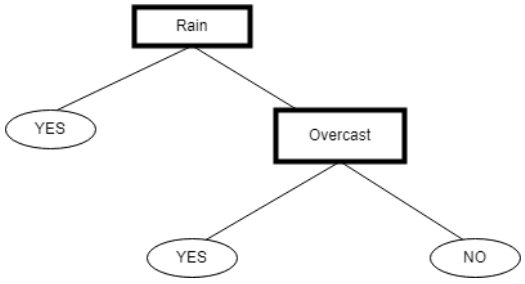
W naszym przypadku usunięcie 40% warunków oznacza usunięcie ≈ 3 najgorszych warunków. w ten sposób pozostaną nam następujące warunki z wyliczonymi szansami na zostanie wybranym:

$$p(Sunny) = \frac{0,48}{0,48+0,21+0,21+0,48} = 35\%$$
$$p(Rain) = \frac{0,21}{0,48+0,21+0,21+0,48} = 15\%$$
$$p(temperature < t_2) = \frac{0,21}{0,48+0,21+0,21+0,48} = 15\%$$
$$p(temperature < t_3) = \frac{0,48}{0,48+0,21+0,21+0,48} = 35\%$$

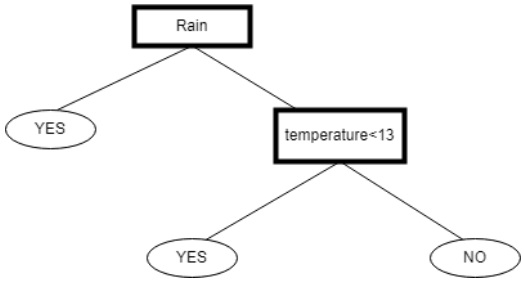
Na podstawie tych wartości prawdopodobieństwa losujemy, który warunek zostanie wybrany do węzła decyzyjnego. Następnie, z utworzonych przez kryterium wyboru dwóch podzestawów danych rekurencyjnie wybierane są następne warunki z wylosowanego wcześniej zestawu atrybutów, aż do momentu, w którym wszystkie elementy w zbiorze mają tę samą etykietę, wtedy węzeł taki staje się liściem o danej etykiecie, lub do momentu osiągnięcia maksymalnej, wcześniej ustalonej, głębokości drzewa lub osiągnięcia przez podzbiór danych minimalnej liczności elementów. W momencie osiągnięcia kryterium liczności węzeł taki staje się liściem, gdzie etykieta wybierana jest według większości elementów danego typu w podzbiorze. W przypadku równej liczby takich elementów etykieta wybierana jest losowo.
W wyżej przedstawionym przypadku mamy łącznie 70% (podział ze względu na Sunny oraz ze względu na t_3) na to, że zostanie wybrany warunek dzielący zestaw treningowy na dwa podzbiory z tylko jedną klasą etykiet w danym zbiorze, oraz 30% (podział ze względu na Rain oraz ze względu na t_2). Załóżmy, że wybrany został warunek ze względu na wartość atrybutu 'Rain'. Możemy otrzymać wtedy następujące drzewo:



Lub takie:



Lub takie:



To jakie drzewo ostatecznie otrzymalibyśmy po wywołaniu algorytmu zależy od ruletki.

4 Plan eksperymentów

W celu zbadania wpływu różnych parametrów na przebieg oraz wydajność działania naszego algorytmu stworzonego na bazie algorytmu CART ze zmienionym warunkiem doboru wyborów w węzłach decyzyjnych opartych na ruletce. Nasze pomysły na przeprowadzenie eksperymentów:

- zmiana liczby modeli drzew wykorzystanych do budowy lasu losowego w danej iteracji,
- zmiana liczby atrybutów dostępnych dla danego modelu w procesie budowy drzewa
- wpływ parametru depth przy budowie drzewa
- wpływ różnych metod wyliczania warunków dla węzłów decyzyjnych dla atrybutów ciągłych
- wpływ zmiany minimalnej liczby elementów w podzbiorze danych wymaganej do przeprowadzenia procesu tworzenia nowego węzła decyzyjnego

Dodatkowo porównamy wynik działania naszego algorytmu z algorytmem typu CART wybierającym wybory w węzłach decyzyjnych na podstawie najmniejszej wartości wskaźnika Gini. Ostatecznie wyniki będziemy porównywać według metryk jakości opisanych w kolejnym rozdziale.

5 Miary jakości

5.1 Tabela pomyłek i dokładność algorytmu

Do oceny jakości, a właściwie dokładności modelu będziemy posługiwać się *confusion matrix*, czyli tabelą pomyłek. [5] Taka tabela składa się z następujących pól:

Wynik algorytmu	Klasa rzeczywista	
	Pozytywna	Negatywna
Pozytywny	TP	FN
Negatywny	FP	TN

Gdzie:

- `TP` - true positive - przykłady prawdziwie pozytywne
- `TN` - true negative - przykłady prawdziwie negatywne
- `FP` - false positive - przykłady fałszywie pozytywne
- `FN` - false negative - przykłady fałszywie negatywne

Z wykorzystaniem tych wartości możemy obliczyć następujące wskaźniki:

- Accuracy/Dokładność (ACC): $\frac{TP+TN}{TP+TN+FP+FN}$
- True positive rate/Czułość/recall (TPR): $\frac{TP}{TP+FN}$
- Precision/Precyzja (PPV):$\frac{TP}{TP+FP}$
- True negative rate (TNR): $\frac{TN}{FP+TN}$
- false alarm rate (FAR): $\frac{FP}{TN+FP}$

Poza FAR oczekujemy od dobrego algorytmu jak najwyższych wartości. Dodatkowo, możemy wyznaczyć:

F_1: $2 \cdot \frac{precision \cdot recall}{precision+recall}$

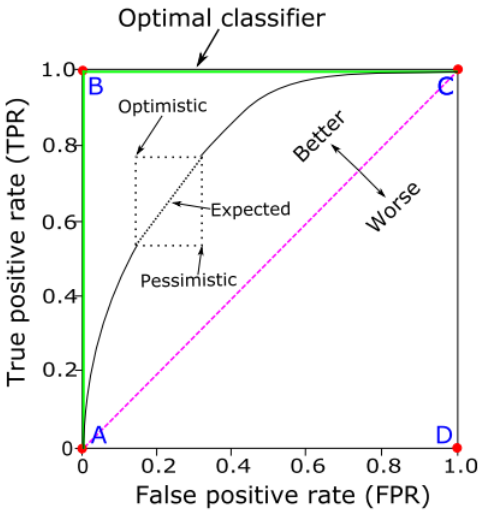
$F_1 - score$ oznacza średnią harmoniczną TPR i PPV; jej zakres wynosi od 0 do 1; Wysokie wartości wskazują na wysoką jakość klasyfikacji algorytmu.

$$MCC = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$$

Matthews correlation coefficient (MCC) oznacza: +1 perfekcyjną predykcję, -1 kompletny brak zgodnych wartości przewidzianych z rzeczywistymi, 0 wynik nie lepszy niż wartości losowe. [4]

5.2 Krzywa ROC

Receiver operating characteristics jest dwu-wymiarowym wykresem, gdzie na osi y zareprezentowane są wartości TPR, a na osi x wartości FPR, gdzie $FPR = 1 - TNR = \frac{FP}{FP+TN}$. Pozwala oszacować równowagę pomiędzy benefitami (TPR) i kosztami (FPR). Wartości szybciej zbiegające do punktu (0,1) oznaczają wyższą jakość.
Drzewa decyzyjne zwracają jedną etykietę dla każego testowanego przypadku, co tworzy jedną tabelę pomyłek, a w rezultacie daje tylko jeden punkt w przestrzeni ROC. Dlatego w celu ich porównania, konieczne jest ich umieszczenie na jednym wykresie. [4]



6 Zbiory danych

Do wykonania projektu wybraliśmy trzy zestawy danych, które zapewnią klasy: binarne, wieloklasowe jak i dziedziny atrybutów: dyskretne oraz ciągłe. Ważnym askpektem przy wyborze była liczba rekordów - staraliśmy się rozważać tylko zbiory danych zawierające powyżej 1000 wpisów.

6.1 Gender Classification Dataset

(url) - binarna klasa, atrybuty dyskretne i ciągłe, zbiór zblasansowany; rekordów: 5001
Atrybuty:

- $long\_hair$ - atrybut binarny; 1 - ma długie włosy, 0 - nie ma długich włosów
- $forehead\_width\_cm$ - atrybut ciągły [cm]
- $forehead\_height\_cm$ - atrybut ciągły [cm]
- $nose\_wide$ - atrybut binarny; 1 - ma szeroki nos , 0 - nie ma szerokiego nosa
- $nose\_long$ atrybut binarny; 1 - ma długi nos , 0 - nie ma długiego nosa
- $lips\_thin$ - atrybut binarny; 1 - ma wąskie usta , 0 - nie ma wąskich ust
- $distance\_nose\_to\_lip\_long$ - atrybut binarny; 1 - ma duży dystans pomiędzy nosem i ustami , 0 - nie ma dużego dystansu między nosem i ustami

Klasa:

- gender - binarna; Male (50%) or Female (50%)

6.2 Red Wine Quality

(url) - wieloklasowy, atrybuty ciągłe, zbiór niezblasansowany; rekordów: 1599.
Atrybuty:

- $fixedacidity$ - atrybut ciągły

- $volatileacidity$ - atrybut ciągły
- $citricacid$ - atrybut ciągły
- $residualsugar$ - atrybut ciągły
- $chlorides$ - atrybut ciągły
- $freesulfurdioxide$ - atrybut ciągły
- $totalsulfurdioxide$ - atrybut ciągły
- $density$ - atrybut ciągły
- pH - atrybut ciągły
- $sulphates$ - atrybut ciągły
- $alcohol$ - atrybut ciągły

Klasa:

- quality - klasa wielowartościowa; 3 (0,63%), 4 (3,31%), 5 (42,59%), 6 (39,9%), 7 (12,45%), 8 (1,13%)

6.3 Heart Disease Classification Datase

(url) - binarna klasa, atrybuty dyskretne i ciągłe, zbiór niezblasansowany, rekordów: 1319.
Atrybuty:

- age - atrybut ciągły
- $gender$ - atrybut binarny; 0 - kobieta, 1 - Mężczyzna
- $impulse$ - atrybut ciągły
- $pressurehight$ - atrybut ciągły
- $pressurelow$ atrybut ciągły
- $glucose$ - atrybut ciągły
- kcm - atrybut ciągły
- $troponin$ - atrybut ciągły

Klasa:

- class - binarna; positive (61%) lub negative (39%)

References

[1] Paweł Cichosz. “Systemy uczące się”. In: Warszawa: Wydawnictwa Naukowo-Techniczne, 2000, p. 139. ISBN: 83-204-2544-1.

[2] Yaliang Li et al. “Ensemble Learning”. In: *Handbook of Data Mining and Knowledge Discovery*. Ed. by Charu C. Aggarwal. CRC Press, Taylor Francis Group, 2015. Chap. 19, pp. 484–494. ISBN: 978-1-4665-8675-8.

[3] Ayush Singhal Gaurav Shukla. “Decision Trees : classification and regression trees (CART)”. In: p. 7. URL: https://www.niser.ac.in/~smishra/teach/cs460/23cs460/lectures/lec9.pdf.

[4] Alaa Tharwat. “Classification assessment methods”. In: 2018, pp. 170–190. URL: https://www.emerald.com/insight/content/doi/10.1016/j.aci.2018.08.003/full/pdf?title=classification-assessment-methods.

[5] Nele Verbiest, Karel Vermeulen, and Ankur Teredesai. “Evaluation of Classification Methods”. In: *Handbook of Data Mining and Knowledge Discovery*. Ed. by Charu C. Aggarwal. CRC Press, Taylor Francis Group, 2015. Chap. 24, pp. 636–642. ISBN: 978-1-4665-8675-8.