# Sprawozdanie z laboratorium:
# Uczenie Maszynowe i Sieci Neuronowe

Część I: Generowanie drzew decyzyjnych
Część II: Generowanie reguł decyzyjnych

20 lutego 2012

Prowadzący: dr inż. Maciej Komosiński


| | | | | |
|---|---|---|---|---|
| Autorzy: | **Krzysztof Urban** | inf84896 | ISWD | krz.urb@gmail.com |
| | **Tomasz Ziętkiewicz** | inf84914 | ISWD | tomek.zietkiewicz@gmail.com |


Zajęcia poniedziałkowe; Krzysztof: 16:50, Tomasz: 13:30

# 1 Generowanie drzew decyzyjnych

## 1.1 Generowanie drzewa

- **Obejrzyj zawartość plików *golf.nam*, *golf.dat* i *golf.tst*; ile przykładów zawiera zbiór uczący? Iloma atrybutami są one opisane?**
  Zbiór uczący zawiera 14 przykładów. Są one opisane pięcioma atrybutami, w tym jednym atrybutem decyzyjnym.

- **Wygeneruj drzewo dla zbioru przykładów *golf*; ustawienia standardowe.**
  Drzewo przed pruningiem:

```
outlook = overcast: Play (4.0)
outlook = sunny:
|   humidity <= 75 : Play (2.0)
|   humidity > 75 : Don't Play (3.0)
outlook = rain:
|   windy = true: Don't Play (2.0)
|   windy = false: Play (3.0)
```

  Drzewo po pruningu:

```
outlook = overcast: Play (4.0/1.2)
outlook = sunny:
|   humidity <= 75 : Play (2.0/1.0)
|   humidity > 75 : Don't Play (3.0/1.1)
outlook = rain:
|   windy = true: Don't Play (2.0/1.0)
|   windy = false: Play (3.0/1.1)
```

- **Przeanalizuj wyniki; czy udało się przeprowadzić pruning?**
  Nie udało się przeprowadzić pruningu. Drzewa są identyczne.

- **Obejrzyj drzewo; ile ma węzłów decyzyjnych, a ile liści?**
  Wygenerowane drzewo ma 3 węzły decyzyjne i 5 liści.

- **Prześledź ścieżkę od korzenia do wybranego liścia.**
  W korzeniu znajduje się test związany z atrybutem outlook. Jeśli dla klasyfikowanego przykładu wartość atrybutu outlook wynosi "sunny", to następny węzeł na ścieżce to ten związany z atrybutem humidity. Zakładając, że dla omawianego przykładu wilgotność jest większa niż 75% , liściem w analizowanej ścieżce będzie liść odnoszący się decyzji "Don't Play".
  Inaczej mówiąc, ścieżka ta klasyfikuje wszystkie dni w które jest słoneczna pogoda i wilgotność większa niż 75% jako dni, w które nie gra się w golfa.

- **Porównaj estymaty błędu dla drzewa oryginalnego (*Unpruned*) i uproszczonego (*Pruned*).**
  Estymata błędu dla oryginalnego drzewa wynosi 0% a dla uproszczonego $38,5\%$ .

- **Obejrzyj macierz pomyłek.**
  Ponieważ drzewo oryginalne nie generuje błędów dla zbioru uczącego, to też jedyne co można z niego odczytać, to właśnie, że żadne obiekty nie są błędnie klasyfikowane, oraz że w przypadku 9 przykładów drzewo poprawnie zaklasyfikowało przykłady do klasy "Play" a w 5 przypadkach do klasy „Don't Play".

## 1.2 Konsultowanie

- **Dokonaj konsultacji wymyślonego przykładu dla wygenerowanego drzewa.**

  Dokonano konsultacji przykładu o wartościach atrybutów: outlook = overcast, temperature = 5, humidity = 100, windy = true. W wyniku konsultacji podanego przykładu otrzymano decyzję "Play". Ponieważ wartości atrybutów zostały podane w sposób deterministyczny to decyzja ta była pewna (prawdopodobieństwo, że analizowany przykład należy do klasy "Play" wynosiło 1). Wynik jest zgodny z oczekiwaniami, poniważ wygenerowane drzewo decyzyjne na podstawi galęzi "Outlook = overcast" przydziela wszystkie przykłady z taką wartością atrybutu do klasy "Don't play", niezależnie od wartości pozostałych atrybutów.

- **Konsultowanie przykładu „niepełnego"; dokładnie przeanalizuj wynik.**

  Przykład niepełny miał postać: (overcast = sunny, temperature = 15, humidity = ?, windy = false). Przechodząc ścieżkę klasyfikującą ten przykład dochodzimy do atrybutu humidity. Nie jest znana wartość analizowanego przykładu na tym atrybucie, więc zostają obliczone prawdopodobieństwa przyjęcia konkretnych wartości. W analizowanym węźle (za gałęzią "sunny") znajduje się 5 przykładów uczących, w tym 2 ( 40% ) przyjmują wartość "humidity" większą niż 75% a 3 (60%) mniejszą. Dlatego przyjmuje się, że analizowany przykład będzie miał z prawdopodobieństwem 60% wartość humidity większą niż 75% a mniejszą z prawdopodobieństwem 40% . Dalej w drzewie sa juz tylko liście, można więc obliczyć rpawdopodobieństwo całych kompletów wartości: z prawdopodobieństwem 40% analizowany przykład przyjmie wartości (overcast = sunny, temperature = 15, humidity > 75%, windy = false) a z prawdopodobieństwem 60% wartości (overcast = sunny, temperature = 15, humidity > 75% , windy = false). Drugie prawdopodobieństwo jest wyższe, więc z prawdopodobieństwem 60% konsultowany przykład zostaje przypisany do klasy "Don't Play"

- **Konsultowanie, gdy znany jest rozkład prawdopodobieństwa.**
  W przypadku przykładu ze znanym rozkładem prawdopodobieństwa analizowane są wszystkie gałęzie o niezerowym prawdopodobieństwie z wagami odpowiadającymi przypisanym danm wartośćiom prawdopodobieństwom. Analizowany przykład miał postać: P(sunny) = 0.8 , P(overcast) = 0.2 . Dlatego, jeśli wartość humidity była większa niż 75% to algorytm przydzielał przykład z 20% prawdopodobieństwem do klasy "Play" i 80% prawdopodobieństwem do klasy "Don't play". Ponieważ dla "humidity" mniejszej niż 75% i "outlook" = "sunny" drzewo podejmuje decyzję "Play", taką samą jak przy "outlook" = "overcast", to prtawdopodobieństwa wtedy się sumują i lagorytm podejmuje decyzję "Play" z prawdopodobieństwem 100% .

## 1.3 Różnica między *gain ratio* a *info gain* w praktyce

- **Obejrzyj zbiór *testgain* (*dat* i *nam*).**

- **Wygeneruj dla niego dwukrotnie drzewo z użyciem opcji *gain ratio* i *info gain*; skomentuj wyniki.**

  Drzewo wygenerowane z użyciem opcji InfoGain:

```
a1 = 1: A (2.0)
a1 = 2: A (2.0)
a1 = 3: B (2.0)
a1 = 4: B (2.0)
```

Drzewo wygenerowane z użyciem opcji GainRatio:

```
a2 = 1: A (4.0)
a2 = 2: B (4.0)
```

Oba atrybuty są typu wyliczeniowego (pzyjmują wartości z określonego w definicji problemu zbioru). Zbiór wartości atrybutu a1 ma wielkość 4 a atrybutu a2 wielkość 2. Współczynnik InfoGain dla obu atrybutów wynosi 1, nie preferuje więc atrybutu a2, którego dziedzina ma mniej wartości niż dziedzina atrybutu a1. Inaczej jest w przypadku współczynnika GainRatio: preferuje on atrybuty o mniejszych dziedzinach. Współczynnik ten wynosi 1/2 dla atrybutu a1 i 1 dla atrybytu a2, co za tym idzie w drzewie zbudowanym przy jego pomocy brany jest pod uwagę atrybut a2. Jak widać wpływa to pozytywnie na rozmiar drzewa a co za tym idzie jego prostotę i łatwość interpretacji.

## 1.4 Grupowanie wartości atrybutów

- **Wygeneruj drzewo dla zbioru *testgain*, zaznaczając opcję *Subsetting*.**

```
a1 in {1,2}: A (4.0)
a1 in {3,4}: B (4.0)
```

Jak widać grupowanie wpłynęło pozytywnie na wielkość generowanego drzewa.

- **Analogicznie dla *CRX*: opisać problem (przyznawanie kard kretytowych), obejrzeć zbiór (atrybuty *A4*, *A6* i *A7* mają wiele wartości); wygenerować drzewo bez i z grupowaniem.**

Ponieważ dane dotyczące tego problemu zawierały poufne dane etykiety atrybutów zostały zmienione na nie nieznaczące symbole.
W problemie tym występują dwie klasy decyzyjne: "+" i "-". W zbiorze uczącym mogą mieć one interpretację klientów, którzy odpowiednio: spłacali lub niespłacali zaciągniętych kredytów w terminie spłaty. Przy klasyfikacji mogą mieć one interpretację: pozytywne lub negatywne rozpatrzenie wniosku o przyznanie karty kredytowej. Każdy klient opisany jest za pomocą 15 atrybutóW, w tym 6 atrybutów przyjmujących wartości ciągłe i 9 atrybutów przyjmujących wartości ze znanych 2 do 14 elementowych zbiorów.

- **Obejrzeć macierz pomyłek dla zbioru uczącego i testującego; czy w tym zastosowaniu przydałaby się macierz kosztów pomyłek?**

Dla drzewa bez grupowania procent błędnych klasyfikacji w zbiorze trenującym wynosi $6,5\%$, w zbiorze testowym $20,5\%$.
Dla drzewa z grupowaniem wartości te wynoszą odpowiednio $6,7\%$ i $17,5\%$. Na podstawie tych wartości można się domyślać, że występuje zjawisko przeuczenia: dla większego drzewa (bez grupowania) różnica w trafność klasyfikacji przykładów ze zbioru uczącego i testowego jest większa niż dla mniejszego drzewa.
To czy warto wprowadzić macierz kosztów pomyłek zależy od analiz ekonomicznych banku - może okazać się, że mniejszą stratę przynosi wydanie karty "złemu" klientowi niż niewydanie karty "dobremu" klientowi - np. klient niespłącający w terminie może być wbrew pozorom zjawiskiem korzystnym, ponieważ jeśli w końcu zapłaci karne odsetki to mogą one stanowić dla banku dodatkowy zysk, większy niż koszty obsługi kredytu. Zgodnie z wiedzą banku macierz kosztów pomyłek powinna być skonstruowana

tak, żeby minimalizować błędy pierwszego (false positive) lub drugiego (false negative) rodzaju, w zależności od tego które z nich są dla banku bardziej kosztowne.

## 1.5 Poszukiwanie optymalnej wielkości drzewa uproszczonego

- **Poszukiwanie optymalnej wielkości drzewa uproszczonego przez dobór poziomu ufności procedury upraszczającej (*Pruning confidence level*); przeprowadź serię eksperymentów *10-fold cross-validation* dla zbioru *Monk2*, ze zmieniającym się poziomem ufności od 0.05 do 0.5, z krokiem co najwyżej 0.05. Sporządź wykres zależności:**

  – średniego (po *cross-validation*) rozmiaru drzewa uproszczonego,

  – średniej trafności klasyfikowania drzewa uproszczonego na zbiorze testującym,

  – średniej estymaty błędu dla drzewa uproszczonego...
  w funkcji poziomu ufności (odnieś te wyniki do średniej charakterystyki drzewa oryginalnego, nieuproszczonego).

  Zgodnie z dokumentacją programu c4.5, im mniejsza wartość parametru "Pruning confidence level" tym mniejszy rozmiar drzewa po pruningu. Widać to n wykresie 1. Dla porównania, średnia wielkość drzewa przed pruningiem wynosiła 63.3 .

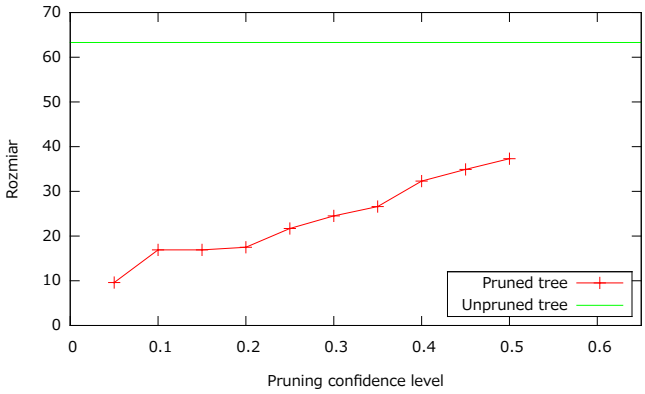
  Wraz ze zmniejszaniem wielkości drzewa różnie estymata błędu klasyfikacji (rys. 3). Jest to zrozumiałe, ponieważ estymata jest obliczana na podstawie zbioru uczącego, a oryginalne drzewo jest do niego najlepiej dopasowane. Inaczej jest w przypadku zbioru testowego: trafność klasyfikacji przykładów z tego zbioru jext najwyższa dla Pruning confidence level = 0.2. Dla mniejszych drzew trafność maleje ponieważ są one zbyt uproszczone, dla większych, ponieważ są one przeuczone (dopasowane za bardzo do zbioru uczącego). Trafność dla drzewa nieprzyciętego jest dla wszystkich badanych rozmiarów drzewa mniejsza niż dla drzewa przyciętego - uwidacznia to przeuczenie drzewa.

  Analizując otrzymane wyniki można stwierdzić, że optymalna wielkość drzewa uproszczonego to ta, którą osiąga ono dla Pruning confidence level = 0.2, czyli 17.5. Dla tej wartości trafność klasyfikacji na zbiorze testowym jest największa, jednocześnie jest to małe drzewo, niemal czterokrotnie mniejsze od drzewa oryginalnego. Oczywiście to, którą wielkość uznamy za najlepszą zależy od przyjętej miary i tego w jakim stopniu minimalizuje ona wilekość drzewa a w jakim stopniu zachowuje jego trafność.

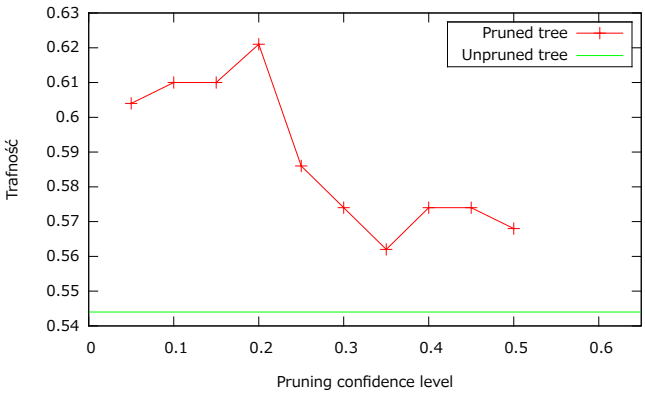
- **Poszukiwanie optymalnej wielkości drzewa uproszczonego poprzez preprуning, tj. manewrowanie minimalną licznością węzła (*Minimum objects*). Dla zbioru *CRX* przebadać przedział od 2 do 10.**

  Jak widać na rysunku 4 rozmiar drzewa maleje wraz ze wzrostem minimalnej wielkości węzła, przy czym powyżej wielkości węzła 6 dynamika spadku wielkości drzewa znacząco maleje. Jednocześnie z wykresu 5 wynika, że dla minimalnej wielkości węzłą 7 drzewo osiąda najwyższą trafność klasyfikacji. W związku z tym optymalną wielkością drzewa uproszczonego przez preprouning jest wielkość 38, która jest osiągana dla minimalnej liczności węzła równej 7.

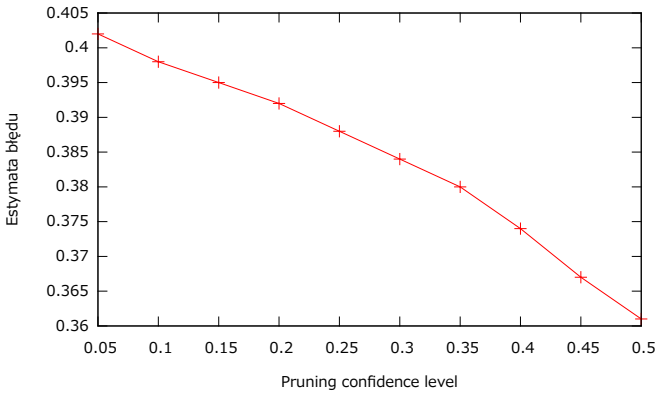
- **Analiza wygenerowanego drzewa: poszukiwanie słabych punktów (liści o małym wsparciu, poddrzew które generują szczególnie dużo błędów, etc.).**



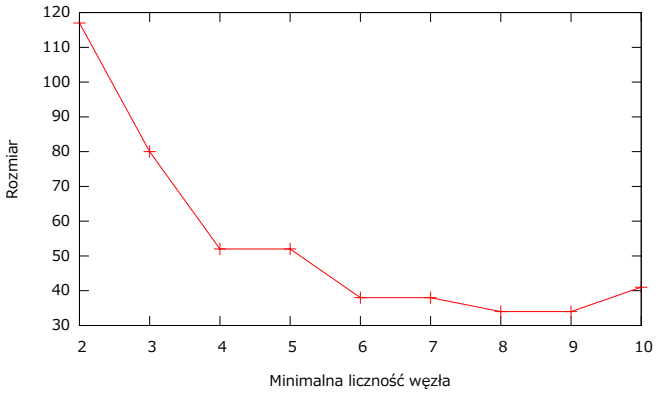

Rysunek 1: Wykres zależności średniego (po *cross-validation*) rozmiaru drzewa uproszczonego w funkcji poziomu ufności. Rozmiar drzewa nieprzyciętego pokazano dla porównania, jest on niezależny od parametru "Pruning confidence level"



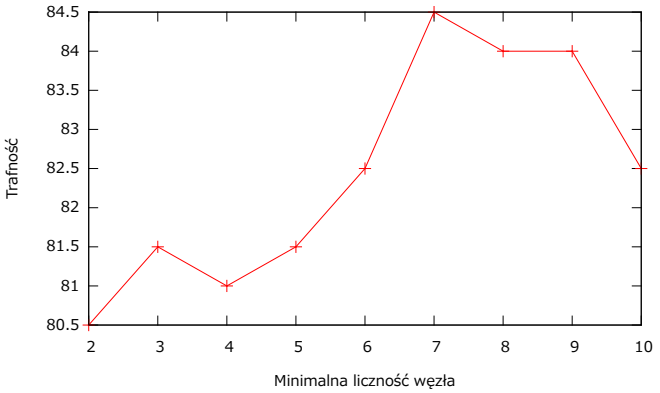

Rysunek 2: Wykres zależności średniej (po *cross-validation*) trafności klasyfikacji dla drzewa uproszczonego na zbiorze tesującym w funkcji poziomu ufności. Trafność dla drzewa nieprzyciętego pokazano dla porównania, jest ona niezależna od parametru "Pruning confidence level"




Rysunek 3: Wykres zależności średniej (po *cross-validation*) estymaty błędu dla drzewa uproszczonego w funkcji poziomu ufności




Rysunek 4: Wykres zależności średniego (po *cross-validation*) rozmiaru drzewa w funkcji minimalnej liczności węzła."




Rysunek 5: Wykres zależności średniej (po *cross-validation*) trafności klasyfikacji na zbiorze tesującym w funkcji minimalnej liczności węzła.

## 1.6 *Windowing*

- **Wyjaśnić zasadę i opcje: *Trials*, *Initial window size*, *Window increment*.**

  Technika "okien" ("Windowing") polega na konstruowaniu drzewa na podstawie podzbioru zbioru uczącego zwanego oknem. Początkowa wielkość okna ustalana jest za pomocą parametru "Initial window size". Po skonstruowania drzewa w oparciu o aktualne okno pozostałe przykłady ze zbioru uczącego są za jego pomocą klasyfikowane. Część z błednie sklasyfikowanych przykładów trafia do nowego "okna". Ich liczbę określa parametr "Window Increment". Na podstawie nowego okna drzewo jest aktualizowane. Ponieważ początkowe okno jest losowane, to takich okien, a co za tym idzie powstających z nich drzew może być wiele. Ilość generowanych okien ustala parametr "Trials". P otworzeniu wszystkich okien wybierane jest najlepsze z nich.

- **Analiza wyników (*CRX*)**

  Wśród 10 wygenerowanych drzew (Trials = 10) ( z wielkościami parametrów Initital window size i window increment dobieranymi przez program) większość okazała się mniejsza od drzewa wygenerowanego bez użycia techniki windowing (Normalne drzewo - wielkość 72, drzewa wygenerowane przy użyciu okien - średnio 67,8; minimalnie 62, maksymalnie 75). Trafność klasyfikacji drzew konstruowanych przy pomocy okien nie różniła się znacząco od drzewa konstruowanego tradycyjnie: różnice wynosiły maksymalnie ok $\pm 1\%$ . Ogólnie wydaje się, że w testowanym zbiorze technika okien nie przyniosła znacznych korzyści. Może być ona jednak przydatna jeżeli ostotne są ograniczenia pamięciowe.

## 1.7 Generowanie krzywej uczenia

- **Dla zbioru *vote* przygotować kilka[naście] zbiorów uczących o liczności $n$ zmieniającej się od 50 do 300, ze skokiem np. 50 przypadków, poprzez wybieranie pierwszych $n$ ze zbioru *vote.dat*.**

- **Wykreślić jako funkcję $n$ rozmiar drzewa uproszczonego oraz trafność klasyfikowania drzewa uproszczonego na zbiorze testującym.**

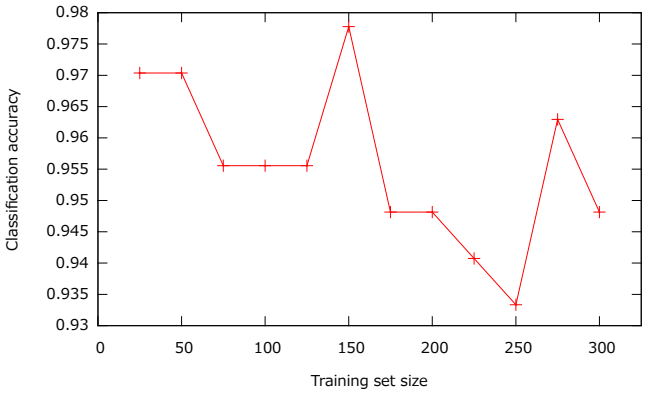
  Jak widać na wykresie 7 rozmiar generowanego drzewa rośnie wraz z wielkością zbioru uczącego, aż do wielkości 300 kiedy zaczyna maleć. Po analizie drzew wygenerowanych na podstawie zbiorów o liczności 275 i 300 można zauważyć, że drzewo wygenerowane dla większego zbioru jest mniejsze, ponieważ rozkład klas w węźle jest w nim na tyle mało równomierny (3 / 94 przypadki w porównaniu z 3 / 85 przypadków dla mniejszego zbioru), że entropia w tym węźle jest zbyt mała by opłacało się dokonywać dalszego podziału.

## 1.8 Maksymalizacja trafności

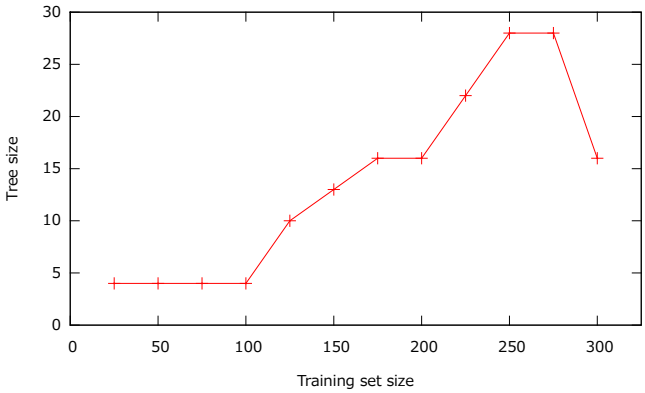
- **Uzyskaj jak najwyższą trafność klasyfikowania ze zbioru *GERMAN* w eksperymencie *10-fold CV*. Jakimi parametrami i mechanizmami można manipulować, by szukać najwyższej trafności? Kiedy można ufać tak uzyskanej trafności, a kiedy można mówić o nadużyciu?**

  Opcje dostępne w menu programu *C4.5 for Windows*:

  - **Criterion** Kryterium oceny wyboru atrybutu na którym ma nastąpić podział w danym węźle drzewa. Dostępne opcje to InfoGain oraz GainRatio, przy czym




Rysunek 6: Wykres zależności trafności klasyfikacji na zbiorze tesującym w funkcji wielkości zbioru uczącego.




Rysunek 7: Wykres zależności rozmiaru generowanego drzewa w funkcji wielkości zbioru uczącego.

InfoGain faworyzuje atrybuty dyskretne z dużą liczbą wartości. Wady tej pozbawiony jest GainRatio.

- **Subseting** W sytuacji gdy występują atrybuty dyskretne z dużą liczbą wartości, subseting prowadzi do prób połączenia kilku wartości w jedną grupę. Im więcej bowiem różnych wartości atrybutu, tym większe prawdopodobieństwo, że uzyskane podzbiory będą do siebie podobne – może to doprowadzić do *fragmentacji*, tzn. powstawania w drzewie wielu bardzo podobnych poddrzew. *Subseting* pozwala na zażegnanie tego problemu i uproszczenie drzewa.
- **Probabilistic thresholds** Wybór tej opcji pozwala na uwzględnienie przy wyborze progów dla atrybutów ciągłych rozkładu prawdopodobieństwa danego atrybutu; powstałe w ten sposób progi charakteryzują się tym, że prawdopodobieństwo przypisania wartości do danego przedziału jest takie samo dla wszystkich przedziałów.
- **Pruning confidence level** Poziom istotności testu statystycznego wykorzystywanego w postpruningu. Węzeł zastępowany jest liściem, jeżeli zależność między rozkładem klas decyzyjnych i wynikiem przeprowadzonego w węźle testu *nie jest*

istotna statystycznie. Im niższa wartość, tym większa redukcja rozmiaru drzewa.

– **Minimum objects** Minimalna liczba obiektów, z których można utworzyć liść. Podanie wartości większej niż 1 zapobiega powstawaniu bardzo wyspecjalizowanych ścieżek w drzewie opisujących pojedyncze obiekty.

Uzyskanej w skutek manipulowania powyższymi parametrami trafności możemy ufać tylko wtedy, gdy nie istnieje podejrzenie *przespecjalizowania* drzewa. Mechanizm *10-fold CV* zapobiega przespecjalizowaniu poprzez wielokrotne budowanie drzewa dla różnych podzbiorów zbioru uczącego, nie daje nam to jednak gwarancji uzyskania wysokiej trafności na „nie widzianych" wcześniej przez drzewo przypadkach.

Dla domyślnych parametrów (wykorzystanie *Info Gain*, bez użycia mechanizmów *subsetting* i *probabilistic thresholds*, wartość *minimum objects* ustawiona na 2) średni estymowany błąd wynosi 27.3%.

Pierwszym krokiem ku poprawieniu jakości klasyfikacji jest zmiana metody wyboru atrybutu na *Gain Ratio*. Przy takich ustawieniach średni estymowany błąd wynosi 23.5%, co stanowi istotną poprawę.

Ponieważ w zbiorze atrybutów występują atrybuty ciągłe, postanawiamy wykorzystać dodatkowo mechanizmy *subsetting* i *probabilistic thresholds*. Udaje nam się uzyskać zmniejszenie wartości średniego estymowanego błędu do 20.8%.

# 2 Generowanie reguł decyzyjnych

## 2.1 Metoda pośrednia generowania reguł (*C4.5rules*)

- **Wygeneruj reguły dla zbioru *GOLF* za pomocą programu *C4.5 for Windows*.**

  Wyniki pokazane są na Rys. 8.

- **Porównaj wygenerowane reguły z wyjściowym drzewem decyzyjnym. Czy reguły odzwierciedlają precyzyjnie drzewo?**

  Wyjściowe drzewo zostało przedstawione na Rys. 9. Drzewo to nie może być odtworzone przy użyciu samych tylko uzyskanych reguł (Rys. 8). Jest tak, ponieważ nie pokrywają one wszystkich możliwych ścieżek od korzenia do liścia – brakuje reguły dla `outlook = sunny AND humidity <= 75`. Ponieważ jednak w wyniku zawarta jest również klasa domyślna, w tym wypadku `Play`, możemy ją wykorzystać jako liść dla ścieżek w drzewie odpowiadającym brakującym regułom. W ten sposób, w tym konkretnym przypadku, możliwe jest zrekonstruowanie drzewa. Stąd reguły odzwierciedlają drzewo w sposób precyzyjny.

## 2.2 Porównanie klasyfikowania za pomocą drzew decyzyjnych i reguł decyzyjnych (*C4.5rules*)

- **Przeprowadź testy *10-fold CV* na wybranych zbiorach dla drzew i reguł.**

  Wyniki uzyskane z programu *C4.5 for windows* zostały przedstawione następująco: dla zbioru *golf* w Tab. 1 oraz Tab. 2, dla zbioru *vote* w Tab. 3 oraz Tab. 4.

- **Porównaj wyniki pod kątem trafności klasyfikowania na zbiorze testującym oraz rozmiaru opisu.**

```
Rule 1: [70.7%]
    IF    outlook = overcast
    THEN  Play

Rule 2: [63.0%]
    IF    outlook = rain
    AND   windy = false
    THEN  Play

Rule 3: [63.0%]
    IF    outlook = sunny
    AND   humidity > 75
    THEN  Don't Play

Rule 4: [50.0%]
    IF    outlook = rain
    AND   windy = true
    THEN  Don't Play

Default class: Play

Errors in training set: 0 (0.0%)
```

Rysunek 8: Reguły decyzyjne dla zbioru *golf* uzyskane algorytmem *C4.5rules*

```
Pruned decision tree:

outlook = overcast: Play (4.0/1.2)
outlook = sunny:
|   humidity <= 75 : Play (2.0/1.0)
|   humidity > 75 : Don't Play (3.0/1.1)
outlook = rain:
|   windy = true: Don't Play (2.0/1.0)
|   windy = false: Play (3.0/1.1)
```

Rysunek 9: Drzewo decyzyjne dla zbioru *golf* uzyskane algorytmem *C4.5*

| | Before pruning | | | | | After pruning | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Tree | Size | | Errors | Errors (test) | | Size | | Errors | Errors (test) | | Estimate |
| 1 | 8 | 0 | 0.0% | 0 | 0.0% | 8 | 0 | 0.0% | 0 | 0.0% | 43.5% |
| 2 | 8 | 0 | 0.0% | 1 | 50.0% | 8 | 0 | 0.0% | 1 | 50.0% | 43.1% |
| 3 | 6 | 2 | 16.7% | 1 | 50.0% | 1 | 4 | 33.3% | 1 | 50.0% | 47.5% |
| 4 | 6 | 1 | 8.3% | 1 | 50.0% | 6 | 1 | 8.3% | 1 | 50.0% | 44.5% |
| 5 | 6 | 1 | 7.7% | 1 | 100.0% | 6 | 1 | 7.7% | 1 | 100.0% | 42.1% |
| 6 | 8 | 0 | 0.0% | 0 | 0.0% | 8 | 0 | 0.0% | 0 | 0.0% | 41.0% |
| 7 | 8 | 0 | 0.0% | 0 | 0.0% | 8 | 0 | 0.0% | 0 | 0.0% | 41.0% |
| 8 | 8 | 0 | 0.0% | 0 | 0.0% | 8 | 0 | 0.0% | 0 | 0.0% | 40.6% |
| 9 | 8 | 0 | 0.0% | 0 | 0.0% | 8 | 0 | 0.0% | 0 | 0.0% | 40.6% |
| 10 | 6 | 1 | 7.7% | 1 | 100.0% | 6 | 1 | 7.7% | 1 | 100.0% | 42.1% |
| Avg. | 7.2 | 0.5 | 4.0% | 0.5 | 35.0% | 6.7 | 0.7 | 5.7% | 0.5 | 35.0% | 42.6% |

Tabela 1: *Cross-validation* dla drzew utworzonych ze zbioru *golf*

| Ruleset | Size | | Errors | Errors (test) | |
|---|---|---|---|---|---|
| 1 | 2 | 0 | 0.0% | 0 | 0.0% |
| 2 | 3 | 0 | 0.0% | 1 | 50.0% |
| 3 | 1 | 4 | 33.3% | 1 | 50.0% |
| 4 | 3 | 1 | 8.3% | 2 | 100.0% |
| 5 | 4 | 1 | 7.7% | 1 | 100.0% |
| 6 | 4 | 0 | 0.0% | 0 | 0.0% |
| 7 | 4 | 0 | 0.0% | 0 | 0.0% |
| 8 | 3 | 0 | 0.0% | 0 | 0.0% |
| 9 | 3 | 0 | 0.0% | 0 | 0.0% |
| 10 | 2 | 1 | 7.7% | 1 | 100.0% |
| Avg. | 2.9 | 0.7 | 5.7% | 0.6 | 40.0% |

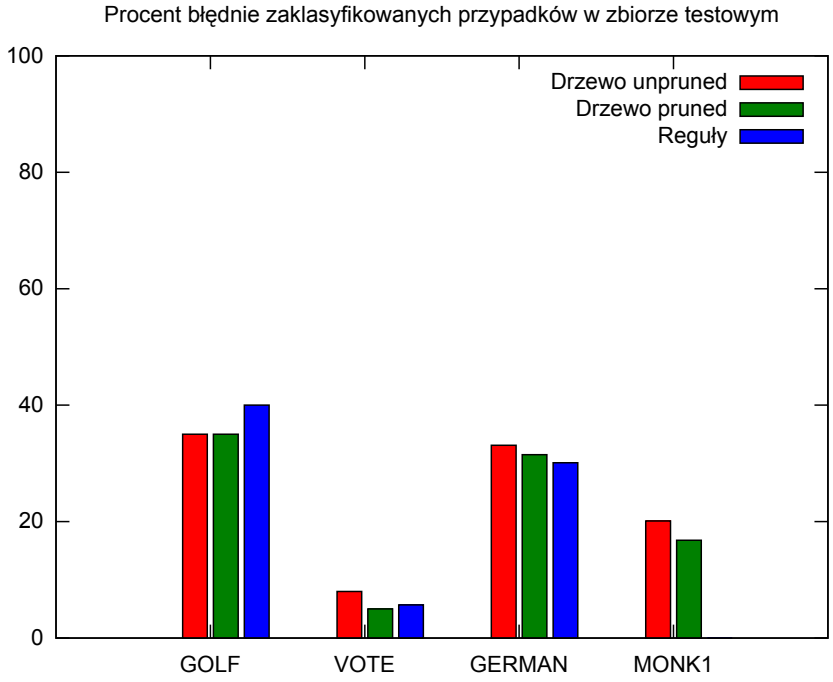
Tabela 2: *Cross-validation* dla reguł utworzonych ze zbioru *golf*

| | Before pruning | | | | | After pruning | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Tree | Size | | Errors | Errors (test) | | Size | | Errors | Errors (test) | | Estimate |
| 1 | 25 | 7 | 2.6% | 4 | 13.3% | 7 | 12 | 4.4% | 1 | 3.3% | 7.3% |
| 2 | 25 | 7 | 2.6% | 1 | 3.3% | 7 | 12 | 4.4% | 1 | 3.3% | 7.2% |
| 3 | 16 | 9 | 3.3% | 0 | 0.0% | 7 | 13 | 4.8% | 0 | 0.0% | 7.7% |
| 4 | 19 | 10 | 3.7% | 1 | 3.3% | 7 | 13 | 4.8% | 0 | 0.0% | 7.7% |
| 5 | 16 | 8 | 3.0% | 1 | 3.3% | 7 | 11 | 4.1% | 2 | 6.7% | 6.9% |
| 6 | 16 | 8 | 3.0% | 4 | 13.3% | 7 | 11 | 4.1% | 2 | 6.7% | 6.9% |
| 7 | 31 | 5 | 1.9% | 4 | 13.3% | 4 | 13 | 4.8% | 2 | 6.7% | 7.0% |
| 8 | 28 | 5 | 1.9% | 4 | 13.3% | 4 | 12 | 4.4% | 3 | 10.0% | 6.6% |
| 9 | 16 | 7 | 2.6% | 2 | 6.7% | 7 | 11 | 4.1% | 2 | 6.7% | 6.8% |
| 10 | 13 | 8 | 3.0% | 3 | 10.0% | 7 | 11 | 4.1% | 2 | 6.7% | 6.8% |
| Avg. | 20.5 | 7.4 | 2.8% | 2.4 | 8.0% | 6.4 | 11.9 | 4.4% | 1.5 | 5.0% | 7.1% |

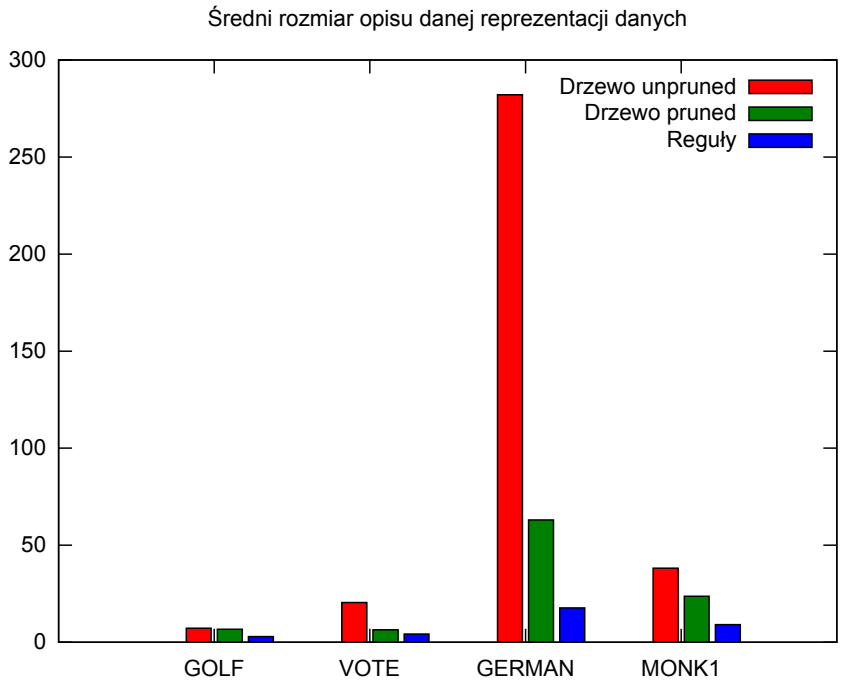
Tabela 3: *Cross-validation* dla drzew utworzonych ze zbioru *vote*

| Ruleset | Size | Errors | | Errors (test) | |
|---|---|---|---|---|---|
| 1 | 4 | 12 | 4.4% | 1 | 3.3% |
| 2 | 3 | 12 | 4.4% | 1 | 3.3% |
| 3 | 5 | 11 | 4.1% | 0 | 0.0% |
| 4 | 4 | 13 | 4.8% | 0 | 0.0% |
| 5 | 5 | 9 | 3.3% | 2 | 6.7% |
| 6 | 4 | 11 | 4.1% | 2 | 6.7% |
| 7 | 4 | 7 | 2.6% | 2 | 6.7% |
| 8 | 4 | 9 | 3.3% | 4 | 13.3% |
| 9 | 4 | 9 | 3.3% | 2 | 6.7% |
| 10 | 5 | 8 | 3.0% | 3 | 10.0% |
| Avg. | 4.2 | 10.1 | 3.7% | 1.7 | 5.7% |

Tabela 4: *Cross-validation* dla reguł utworzonych ze zbioru *vote*




Rysunek 10: Błędy klasyfikowania na zbiorze testującym.




Rysunek 11: Rozmiar opisu różnych reprezentacji wiedzy.

Na Rys. 10 przedstawiono zależność między trafnością klasyfikowania a typem reprezentacji (drzewa, reguły). Dla wszystkich zbiorów danych z wyjątkiem *golf* reprezentacja regułowa daje lepszą klasyfikację niż drzewa. (Drobne różnice między drzewem po pruningu a regułami dla zbioru *vote* pomijamy.)
Odmienną zależność dla zbioru *golf* możemy tłumaczyć małą liczbą przypadków uczących w tym zbiorze – zaledwie 14 przykładów – przez co zbiory testowe zawierają 1-2 przypadków testowych. W tej sytuacji pojedynczy błąd ma istotny wpływ na końcową jakość klasyfikacji.

Na Rys. 11 dokonano porównania rozmiarów różnych reprezentacji wiedzy dla kilku wybranych zbiorów danych. Jako jednostkę przyjęto w przypadku drzew pojedynczy liść, zaś w przypadku reguł pojedynczą regułę. Widać znaczną redukcję rozmiaru przy przejściu od drzew bez pruningu przez drzewa z pruningiem do reguł.

- **Przeprowadzając kilka eksperymentów uczenia i testowania przeanalizuj wpływ parametrów *Confidence Level* i *Redundancy Factor* na otrzymywany zbiór reguł.**

  Parametr *Confidence Level* steruje liczbą i rozmiarem reguł. Dla zbioru *vote* zależność między wartością tego parametru a rozmiarem zbioru reguł i liczbą błędów przy klasyfikacji przedstawia tabela 5. Jak widać, zwiększenie wartości parametru prowadzi do zwiększenia rozmiaru zbioru reguł. Wprawdzie oznacza to polepszenie klasyfikacji dla zbioru uczącego, jednak (w przypadku zbioru *vote*) nie zauważono wpływu na jakość klasyfikacji dla zbioru testowego.

  Parametr *Redundancy Factor* steruje wyborem atrybutów do reguł. Dla zbioru *vote* zależność między wartością parametru a rozmiarem zbioru reguł i liczbą błędów przy

| Confidence Level | Avg. Size | Avg. Errors | | Avg. Errors (test) | |
|---|---|---|---|---|---|
| 0.10 | 2.9 | 11.4 | 4.2% | 1.8 | 6.0% |
| 0.30 | 4.2 | 10.0 | 3.7% | 1.8 | 6.0% |
| 0.50 | 4.8 | 9.1 | 3.4% | 1.8 | 6.0% |
| 0.60 | 5.2 | 8.8 | 3.2% | 1.8 | 6.0% |
| 0.75 | 7.3 | 8.2 | 3.0% | 1.8 | 6.0% |
| 1.00 | 7.9 | 8.2 | 3.0% | 1.8 | 6.0% |

Tabela 5: Wpływ parametru *Confidence Level* na rozmiar zbioru reguł wygenerowanego dla zbioru *vote*

| Redundancy Factor | Avg. Size | Avg. Errors | | Avg. Errors (test) | |
|---|---|---|---|---|---|
| 0.10 | 2.0 | 13.5 | 5.0% | 1.5 | 5.0% |
| 0.30 | 2.0 | 13.5 | 5.0% | 1.5 | 5.0% |
| 0.40 | 3.0 | 11.3 | 4.2% | 1.7 | 5.7% |
| 0.50 | 3.5 | 11.3 | 4.2% | 1.7 | 5.7% |
| 0.75 | 4.4 | 9.7 | 3.6% | 1.8 | 6.0% |
| 1.00 | 4.8 | 9.1 | 3.4% | 1.8 | 6.0% |

Tabela 6: Wpływ parametru *Redundancy Factor* na rozmiar zbioru reguł wygenerowanego dla zbioru *vote*

klasyfikacji przedstawiona jest w tabeli 6. Im mniejsza wartość parametru, tym mniej atrybutów zostanie wybranych do pojedynczej reguły decyzyjnej, co wpływa bezpośrednio na rozmiar samego zbioru reguł. W zbiorze *vote* zaobserwowano polepszenie klasyfikacji dla małych wartości *Redundancy Factor*.

Podsumowując, sterując tymi parametrami można uniknąć przeuczenia systemu, efekty jednak zależeć będą od konkretnego zbioru przykładów uczących.

## 2.3 Generowanie reguł z użyciem algorytmu *LEM*

- **Wygeneruj reguły dla zbioru *HPAP.ISF*.**

  Wygenerowane reguły zostały przedstawione na rysunku 12.

- **Przyjrzyj się regułom możliwym; opisz je i „wydedukuj", skąd się wzięły.**

  Reguły 1–4 to reguły pewne, natomiast reguła 5 jest regułą przybliżoną. Reguła 1 pokrywa 2 z 3 obiektów klasy $l$, stąd jej pokrycie wynosi 66%, reguła 2 pokrywa dwa obiekty z klasy $m$ (pokrycie 50%), reguła 3 pokrywa jeden obiekt z klasy $m$ (pokrycie 25%), natomiast reguła 4 pokrywa oba obiekty z klasy $v1$. Pozostały dwa obiekty nie pokryte przez żadną regułę pewną – wynika to z faktu, że mimo iż posiadają identyczne wartości atrybutów, należą do dwóch różnych klas. Stąd reguła 5, przybliżona, która przypisuje obiekty jednocześnie do dwóch klas. Gdyby chcieć zamienić ją na reguły możliwe, należy usunąć jedną z klas umieszczonych w alternatywie.

## 2.4 Porównanie reguł generowanych za pomocą algorytmu *LEM* i *C4.5*

- **Wygeneruj reguły przy użyciu obu podejść dla zbiorów: *HPAP*, *VOTE* i *MONK*.**

```
rule 1.
(Hem = f) & (Temp = l) => (Comfort = l);

rule 2.
(Temp = l) & (Hem = g) => (Comfort = m);

rule 3.
(Blood = n) & (Temp = n) => (Comfort = m);

rule 4.
(Blood = h) => (Comfort = vl);

! Approximate rules
rule 5.
(Blood = l) & (Temp = n) => (Comfort = l) OR (Comfort = m);
```

Rysunek 12: Reguły decyzyjne dla zbioru *hpap* uzyskane algorytmem *LEM*

```
Rule 1: [50.0%]
    IF    blood = h
    THEN  vl

Default class: m

Errors in training set: 3 (33.3%)
```

Rysunek 13: Reguły decyzyjne dla zbioru *hpap* uzyskane algorytmem *C4.5-rules*

Reguły dla zbioru *hpap* znajdują się na rysunkach 12 i 13, dla zbioru *vote* na rysunkach 14 i 15, zaś dla zbioru *monk1* na rysunkach 16 oraz 17. Jak widać, algorytm *C4.5* generuje znacznie mniejsze zbiory reguł niż *LEM*. W przypadku zbioru *hpap C4.5* tworzy tylko jedną regułę, podczas gdy *LEM* pięć. Podobnie w przypadku zbioru *vote*, algorytm *LEM* generuje aż pięć razy więcej reguł niż *C4.5*; dla zbioru *monk1* generowanych jest dwa razy więcej reguł. Reguły *LEM* wprawdzie pokrywają wszystkie obiekty, przyporządkowując im odpowiednie klasy, jednak reguły *C4.5* są ogólniejsze, wykorzystują mniejszą liczbę atrybutów, przez co unikają przespecjalizowania.

- **Przyjrzyj się niezależnie regułom pewnym i możliwym (*LEM*).**

  Wśród trzech analizowanych zbiorów jedynie dla *hpap* pojawiły się sprzeczności w przypisaniu do klas. Przypadek ten omówiono już w punkcie drugim sekcji 2.3.

```
rule 1. (A9 = n) & (A3 = n) & (A7 = n) & (A12 = y) & (A11 = n) => (A17 = republican)
rule 2. (A13 = y) & (A3 = n) & (A1 = n) & (A12 = y) & (A11 = y) &
        (A2 = y) => (A17 = republican)
rule 3. (A4 = y) & (A2 = n) & (A10 = y) => (A17 = republican)
rule 4. (A9 = n) & (A13 = y) & (A3 = n) & (A2 = y) & (A11 = n) => (A17 = republican)
rule 5. (A3 = n) & (A8 = ?) => (A17 = republican)
rule 6. (A4 = y) & (A6 = y) & (A15 = n) & (A16 = y) => (A17 = republican)
rule 7. (A4 = y) & (A9 = n) & (A7 = y) => (A17 = republican)
rule 8. (A14 = y) & (A12 = ?) & (A1 = n) => (A17 = republican)
rule 9. (A4 = y) & (A1 = y) & (A16 = n) & (A3 = n) => (A17 = republican)
rule 10. (A5 = ?) & (A9 = ?) => (A17 = republican)
rule 11. (A4 = ?) & (A16 = ?) & (A14 = n) => (A17 = republican)
rule 12. (A4 = y) & (A5 = n) & (A10 = y) => (A17 = republican)
rule 13. (A4 = n) & (A3 = y) => (A17 = democrat)
rule 14. (A11 = y) & (A2 = n) & (A16 = n) & (A1 = n) => (A17 = democrat)
rule 15. (A13 = y) & (A16 = ?) & (A1 = y) => (A17 = democrat)
rule 16. (A16 = n) & (A13 = n) => (A17 = democrat)
rule 17. (A12 = y) & (A4 = n) => (A17 = democrat)
rule 18. (A12 = n) & (A7 = y) & (A2 = ?) => (A17 = democrat)
rule 19. (A16 = y) & (A4 = ?) => (A17 = democrat)
rule 20. (A15 = n) & (A4 = n) => (A17 = democrat)
rule 21. (A13 = y) & (A3 = y) & (A16 = ?) => (A17 = democrat)
rule 22. (A3 = n) & (A15 = y) & (A2 = y) => (A17 = democrat)
rule 23. (A9 = y) & (A3 = n) & (A1 = y) => (A17 = democrat)
rule 24. (A12 = ?) & (A9 = y) => (A17 = democrat)
rule 25. (A11 = y) & (A13 = y) & (A10 = y) & (A12 = n) => (A17 = democrat)
```

Rysunek 14: Reguły decyzyjne dla zbioru *vote* uzyskane algorytmem *LEM*

```
Rule 1: [98.4%]
    IF    physician fee freeze = n
    THEN  democrat

Rule 2: [97.5%]
    IF    synfuels corporation cutback = y
    AND   duty free exports = y
    THEN  democrat

Rule 3: [63.0%]
    IF    physician fee freeze = u
    AND   mx missile = n
    THEN  democrat

Rule 4: [88.7%]
    IF    physician fee freeze = y
    THEN  republican

Rule 5: [50.0%]
    IF    physician fee freeze = u
    AND   mx missile = u
    THEN  republican

Default class: democrat

Errors in training set: 11 (3.7%)
Errors in test     set: 6 (4.4%)
```

Rysunek 15: Reguły decyzyjne dla zbioru *vote* uzyskane algorytmem *C4.5-rules*

```
rule 1. (a1 = 1) & (a2 = 3) & (a5 = 3) => (d = 0)
rule 2. (a5 = 2) & (a2 = 1) & (a1 = 3) => (d = 0)
rule 3. (a5 = 4) & (a2 = 2) & (a1 = 1) => (d = 0)
rule 4. (a3 = 2) & (a1 = 2) & (a2 = 1) => (d = 0)
rule 5. (a1 = 2) & (a4 = 2) & (a5 = 4) => (d = 0)
rule 6. (a5 = 3) & (a1 = 2) & (a2 = 1) => (d = 0)
rule 7. (a2 = 3) & (a1 = 1) & (a5 = 4) => (d = 0)
rule 8. (a2 = 2) & (a1 = 1) & (a5 = 3) => (d = 0)
rule 9. (a2 = 3) & (a5 = 3) & (a1 = 2) => (d = 0)
rule 10. (a5 = 2) & (a1 = 1) & (a2 = 3) => (d = 0)
rule 11. (a2 = 2) & (a1 = 3) & (a5 = 4) => (d = 0)
rule 12. (a5 = 2) & (a6 = 1) & (a2 = 2) & (a4 = 3) => (d = 0)
rule 13. (a5 = 4) & (a4 = 3) & (a2 = 1) => (d = 0)
rule 14. (a5 = 3) & (a1 = 3) & (a2 = 1) => (d = 0)
rule 15. (a5 = 2) & (a1 = 2) & (a2 = 1) => (d = 0)
rule 16. (a5 = 2) & (a1 = 2) & (a2 = 3) => (d = 0)
rule 17. (a2 = 2) & (a5 = 3) & (a1 = 3) => (d = 0)
rule 18. (a5 = 2) & (a2 = 2) & (a1 = 1) => (d = 0)
rule 19. (a5 = 4) & (a1 = 2) & (a2 = 3) => (d = 0)
rule 20. (a1 = 2) & (a2 = 2) => (d = 1)
rule 22. (a1 = 3) & (a2 = 3) => (d = 1)
rule 23. (a2 = 1) & (a1 = 1) => (d = 1)
```

Rysunek 16: Reguły decyzyjne dla zbioru *monk1* uzyskane algorytmem *LEM*

```
Rule 1: [95.3%]
    IF    attribute#5 = 1
    THEN  1

Rule 2: [92.2%]
    IF    attribute#1 = 3 AND   attribute#2 = 3
    THEN  1

Rule 3: [91.2%]
    IF    attribute#1 = 2 AND   attribute#2 = 2
    THEN  1

Rule 4: [85.7%]
    IF    attribute#1 = 1 AND   attribute#2 = 1
    THEN  1

Rule 5: [84.1%]
    IF    attribute#1 = 1 AND   attribute#2 = 3 AND   attribute#5 = 4
    THEN  0

Rule 6: [79.4%]
    IF    attribute#1 = 1 AND   attribute#2 = 3 AND   attribute#5 = 2
    THEN  0

Rule 7: [79.4%]
    IF    attribute#1 = 2 AND   attribute#2 = 1 AND   attribute#5 = 4
    THEN  0

Rule 8: [78.4%]
    IF    attribute#1 = 1 AND   attribute#2 = 2
    THEN  0

Rule 9: [66.2%]
    IF    attribute#4 = 2 AND   attribute#5 = 2
    THEN  0

Rule 10: [63.8%]
    IF    attribute#5 = 3 AND   attribute#6 = 1
    THEN  0

Rule 11: [63.5%]
    IF    attribute#3 = 2 AND   attribute#5 = 2
    THEN  0

Rule 12: [58.7%]
    IF    attribute#3 = 2 AND   attribute#5 = 3
    THEN  0

Default class: 0

Errors in training set: 0 (0.0%)
Errors in test     set: 0 (0.0%)
```



Rysunek 17: Reguły decyzyjne dla zbioru *monk1* uzyskane algorytmem *C4.5-rules*