



Wprowadzenie do sztucznej inteligencji, edycja II-2021.

Inżynieria biomedyczna, specjalność: sztuczna inteligencja

Projekt - wykorzystanie teorii zbiorów przybliżonych

Piotr Szczuko, Katedra Systemów Multimedialnych. AI Tech: Akademia Innowacyjnych Zastosowań Technologii Cyfrowych

06 maja 2021 (datę zmienić odpowiednio na dzień wykonania projektu)

|  |  |  |
| --- | --- | --- |
| Nazwisko wykonawcy | Nr indeksu | Termin zajęć projektowych |
| wpisać tu | wpisać | wpisać |

Spis treści

[1 Wprowadzenie 2](#_Toc71193739)

[1.1 Przygotowanie biblioteki Rough Sets 2](#_Toc71193740)

[1.2 Wczytanie zbioru danych 3](#_Toc71193741)

[2 Tożsamość obiektów względem podzbioru cech 3](#_Toc71193742)

[2.1 Wstępne przetwarzanie danych - czyszczenie danych i dyskretyzacja 3](#_Toc71193743)

[2.2 Wyznaczanie klas abstrakcji 5](#_Toc71193744)

[2.2.1 Przykład 1 - zbiór pacjentów 5](#_Toc71193745)

[2.2.2 Przykład 2 - zbiór kandydatów do pracy 6](#_Toc71193746)

[2.3 Dobór analizowanych cech 7](#_Toc71193747)

[3 Przybliżenia zbiorów związane z decyzjami - przybliżenie dolne i górne 7](#_Toc71193748)

[4 Obszar B-pozytywny 8](#_Toc71193749)

[4.1 Ocena obszaru pozytywnego 9](#_Toc71193750)

[5 Generowanie reguł decyzyjnych 9](#_Toc71193751)

[5.1 Generowanie i zastosowanie reguł na własnym zbiorze danych 12](#_Toc71193752)

[6 Algorytmy generowania reguł 12](#_Toc71193753)

[6.1 Modyfikacje algorytmu 13](#_Toc71193754)

[7 Algorytmy generowania reduktów 15](#_Toc71193755)

[8 Oczyszczanie danych i wykorzystanie teorii zbiorów przybliżonych 17](#_Toc71193756)

[8.1 Modyfikacje algorytmu 21](#_Toc71193757)

[9 Podsumowanie 22](#_Toc71193758)

# 1 Wprowadzenie

Celem poniższego materiału do projektu jest zapoznanie z możliwością praktycznego wykorzystana teorii zbiorów przybliżonych w zadaniach analizy danych oraz tworzenia i zastosowania modeli decyzyjnych. Przedstawione są przykłady działania, które można dostosować do własnych zagadnień, obejmujące dyskretyzację danych, wyznaczanie zbiorów obiektów tożsamych, tworzenie reduktu, generowanie reguł, oczyszczanie danych, ocenę dokładności przybliżenia.

## 1.1 Przygotowanie biblioteki Rough Sets

W środowisku R możliwe jest automatyczne pobieranie najnowszej wersji pakietu i instalowanie go, poleceniem install.packages(). Następnie należy go aktywować, tj. włączyć, w celu korzystania z niego, poleceniem library().

## 1.2 Wczytanie zbioru danych

Jednym z typowych formatów przechowywania danych tabelarycznych jest comma separated values, CSV. Jest obsługiwany przez bazy danych, arkusze kalkulacyjne, popularne biblioteki z dziedziny uczenia maszynowego i AI. W języku R odczyt pliku i zapis wykonać można w następujący sposób:

UWAGA: jeśli do zapisu dziesiętnego użyto przecinka, a jako separatora - średnika, to w agrumentach funkcji read.cvs podać należy dodatkowo sep = “;”, dec = “,”:

#przykład:  
#testread = read.csv(file = 'example.csv', sep = ";", dec = ",")

Dowolną zmienną lub obiekt z pamięci aktualnej sesji zapisać można w formacie R poleceniem save (odczyt poleceniem load).

Przez poniższy kod przejdź wiersz po wierszu skrótem Ctrl+Enter.

# 2 Tożsamość obiektów względem podzbioru cech

Podstawowym pojęciem w teorii zbiorów przybliżonych jest tożsamość obiektów (ang. indiscernibility).

Poniższe przykłady prezentują sposób przygotowania danych do analizy metodą zbiorów przybliżonych i wyznaczania zbiorów obiektów tożsamych (tzw. klas abstrakcji), będący konsekwencją analizy wybranych cech różnicujących bądź nie obiekty.

## 2.1 Wstępne przetwarzanie danych - czyszczenie danych i dyskretyzacja

Przykład przedstawia ręcznie zaprojektowany zbiór danych “data”, z cechami pacjentów: temperatura, ból głowy, zawroty i decyzja - grypa. Zostaje on oczyszczony z wierszy, w których wystąpiły wartości nieokreślone NA. Następnie poddany jest dyskretyzacji. Należy zwrócić uwagę na stosowanie metod asDecisionTable() i applyDecTable().

## dane zawarte w strukturze data.frame  
data <- data.frame(  
 c(38.0, 38.1, NA, 37.0, 37.8, 36.5, 36.7, NA),  
 c(NA, "yes", "no", "yes", NA, "yes", "no", "yes"),  
 c("no", "yes", "no", "yes", "yes", "no", "yes", NA),  
 c("yes", "yes", "no", "yes", "no", "no", "no", "yes"))  
  
## nadanie nazw kolumnom, w celu łatwego zarządzania danymi  
colnames(data) <- c("Temp","Headache","Nausea","Flu")  
  
## Z danych utworzona zostaje tablica decyzyjna  
## co polega na wskazaniu, która cecha jest decyzją,   
## oraz, które cechy nie są liczbowe i ich wartości nie mogą być dyskretyzowane ("nominal").  
decTable <-  
 SF.asDecisionTable(  
 dataset = data,  
 decision.att = 4,  
 indx.nominal = c(2:4)  
 )  
  
## Tablica zawiera wartości brakujące NA, dlatego konieczne jest ich odpowiednie przetworzenie.  
## W tym przykładzie zastosowane zostaje usunięcie.  
objNA <-  
 MV.missingValueCompletion(decTable, type.method = "deletionCases")  
  
## objNA z poprzedniego kroku to lista wierszy wskazanych do usunięcia  
## zastosowana zostaje do przetworzenia obecnej tablicy  
new.decTable <- SF.applyDecTable(decTable, objNA)  
print(new.decTable)

## Temp Headache Nausea Flu  
## 2 38.1 yes yes yes  
## 4 37.0 yes yes yes  
## 6 36.5 yes no no  
## 7 36.7 no yes no

## wartości numeryczne (nie "nominal") wymagają dyskretyzacji. Poniżej wyznaczone zostają cięcia  
cuts <-  
 D.discretization.RST(new.decTable, type.method = "local.discernibility")  
  
## Cięcia są wykorzystane do dyskretyzacji cech, które mają ciągłe wartości  
new.decTable <- SF.applyDecTable(new.decTable, cuts)  
print(new.decTable)

## Temp Headache Nausea Flu  
## 1 [36.9, Inf] yes yes yes  
## 2 [36.9, Inf] yes yes yes  
## 3 [-Inf,36.9) yes no no  
## 4 [-Inf,36.9) no yes no

print(cuts$cut.values)

## $Temp  
## [1] 36.85  
##   
## $Headache  
## numeric(0)  
##   
## $Nausea  
## numeric(0)

Oczyszczanie danych wymagane jest w przypadku braku wartości. W praktyce może być to skutkiem niekompletnych pomiarów, błędów w akwizycji danych lub braku odpowiedzi w ankietach. Wykorzystać można różne warianty metody MV.missingValueCompletion. Szczegóły dostępne są w dokumentacji po wpisaniu polecenia ?MV.missingValueCompletion.

Dyskretyzacja danych jest wymagana dla obliczeń z użyciem teorii zbiorów przybliżonych, gdyż bazuje ona na podprzedziałach lub kategoriach (cechach jakościowych a nie ilościowych), które pozwalają wyznaczyć zbiory obiektów wzajemnie tożsamych i na tej podstawie budować reguły. Szczegóły przestawione zostały w materiałach wykładowych oraz dostępne są w dokumentacji po wpisaniu polecenia ?D.discretization.RST.

## 2.2 Wyznaczanie klas abstrakcji

Z uzyskanych powyżej danych zostaną wyznaczone zbiory obiektów tożsamych (tzw. klas abstrakcji), będące konsekwencją wyboru cech. Jeśli analizowane cechy prowadzą do rozróżnienia obiektów, to umieszczone będą one w rozłącznych klasach abstrakcji, jeśli zaś obiekty mają takie same wartości cech, to będą elementami tych samych klas abstrakcji, co zaobserwować można w przykładach poniżej.

### 2.2.1 Przykład 1 - zbiór pacjentów

new.decTable

## Temp Headache Nausea Flu  
## 1 [36.9, Inf] yes yes yes  
## 2 [36.9, Inf] yes yes yes  
## 3 [-Inf,36.9) yes no no  
## 4 [-Inf,36.9) no yes no

## wybór cech, które skutkują podziałem uniwersum na zbiory elementarne:  
## cechy od 1 do 3 są opisem pacjenta, cecha 4 jest decyzją  
cechy <- c(1, 2)  
## relacja równoważności, zwraca numery obiektów, które są tożsame pod względem wybranych cech  
IND <- BC.IND.relation.RST(new.decTable, feature.set = cechy)  
## niektóre klasy abstrakcji są więcej niż 1-elementowe  
print("Klasy abstrakcji powstałe w wyniku wybrania cech:")

## [1] "Klasy abstrakcji powstałe w wyniku wybrania cech:"

names(new.decTable)[cechy]

## [1] "Temp" "Headache"

str(IND$IND.relation) #numery obiektów, które są nierozróżnialne z powodu przyjętych cech. str() wyświetla listę: nazwy wartości cech i numery obiektów, które są nierozróżnialne

## List of 3  
## $ [-Inf,36.9) no : int 4  
## $ [-Inf,36.9) yes: int 3  
## $ [36.9, Inf] yes: int [1:2] 1 2

### 2.2.2 Przykład 2 - zbiór kandydatów do pracy

Powyższe kroki są powtórzone dla innego zbioru. Nie posiada on cech liczbowych, więc nie ma konieczności wykonywania etapu dyskretyzacji.

## W zbiorze danych RoughSetData znajduje się kilka przykładów  
data(RoughSetData)  
## hiring.dt to tabela z danymi o kandydatach do pracy:  
decision.table <- RoughSetData$hiring.dt  
  
## wybór cech, które skutkują podziałem uniwersum na zbiory elementarne:  
## cechy od 1 do 4 są opisem kandydata, cecha 5 jest decyzją  
cechy <- c(3,4)  
## relacja równoważności, zwraca numery obiektów, które są tożsame pod względem wybranych cech  
IND <- BC.IND.relation.RST(decision.table, feature.set = cechy)  
## niektóre klasy abstrakcji są więcej niż 1-elementowe  
print("Klasy abstrakcji powstałe w wyniku wybrania cech:")

## [1] "Klasy abstrakcji powstałe w wyniku wybrania cech:"

names(decision.table)[cechy]

## [1] "French" "Reference"

str(IND$IND.relation) #numery obiektów, które są nierozróżnialne z powodu przyjętych cech. str() wyświetla listę: nazwy wartości cech i numery obiektów, które są nierozróżnialne

## List of 5  
## $ No Excellent : int 8  
## $ No Good : int 4  
## $ Yes Excellent: int [1:2] 1 3  
## $ Yes Good : int 6  
## $ Yes Neutral : int [1:3] 2 5 7

print("klasy kolejnych obiektów są następujące:")

## [1] "klasy kolejnych obiektów są następujące:"

t(decision.table$Decision)

## [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]   
## [1,] Accept Accept Accept Accept Reject Reject Reject Reject  
## Levels: Accept Reject

print("UWAGA: klasa abstrakcji {2, 5, 7} zawiera obiekty o różnych decyzjach!")

## [1] "UWAGA: klasa abstrakcji {2, 5, 7} zawiera obiekty o różnych decyzjach!"

Jeśli rozpatrywana klas abstrakcji zawiera obiekty o tych samych decyzjach, to nie ma konieczności modyfikacji wektora cech, ponieważ w późniejszej klasyfikacji nie będą popełniane błędy. Jeśli natomiast nierozróżnialne obiekty (z tej samej klasy abstrakcji) mają decyzje różne, to potencjalnie prowadzić może to do błędów w podejmowanych decyzjach (np. w klasyfikacji).

## 2.3 Dobór analizowanych cech

W powyższym przykładzie należy w celu sprawdzenia wpływu cech modyfikować zawartość wektora “cechy”, zmieniając numery kolumn branych pod uwagę w procesie wyznaczania klas abstrakcji.

# 3 Przybliżenia zbiorów związane z decyzjami - przybliżenie dolne i górne

Klasy abstrakcji służą jako tzw. zbiory atomowe, z których budowane są zbiory przybliżające grupy obiektów o tej samej decyzji.

* **Dolne przybliżenie**, to zbiór tych obiektów, których klasy abstrakcji całe mają jedną decyzję.
* **Górne przybliżenie**, to zbiór tych obiektów, których klasy abstrakcji mają co najmniej jeden element o rozważanej decyzji.

Precyzja przybliżeń zależeć będzie zawsze od tego jak bardzo granularne (drobne, najlepiej jednoelementowe) są zbiory klas abstrakcji.

W poniższym przykładzie zwrócić należy uwagę na to, które obiekty są tylko w dolnych przybliżeniach, które tylko w górnych. Obiekty, które są tylko w górnym przybliżeniu (a nie w dolnym) tworzą tzw. obszar graniczny.

Optymalna sytuacja to dolne przybliżenie zawierające wszystkie obiekty o wybranej decyzji, które występują w zbiorze treningowym oraz górne przybliżenie równe dolnemu przybliżeniu.

Zachodzi zależność: .

Należy zwrócić uwagę, że różnica między dolnym a górnym przybliżeniem to tzw. obszar graniczny. Jeśli jest on pusty, czyli przybliżenia są równe, to mamy do czynienia z przybliżeniem idealnym.

decision.table <- RoughSetData$hiring.dt  
cechy <- c(2,3) ## inny zbiór cech, obiekty są całkowicie rozróżnialne  
IND <- BC.IND.relation.RST(decision.table, feature.set = cechy)  
## na podstawie relacji IND, czyli utworzonych klas abstrakcji, wyznaczane są dolne i górne przybliżenia związane z decyzjami Accept i Reject. Uzyskujemy zbiory przybliżone.  
roughset <- BC.LU.approximation.RST(decision.table, IND)  
print("Przybliżenia zbioru Accept")

## [1] "Przybliżenia zbioru Accept"

print(" Dolne:")

## [1] " Dolne:"

roughset$lower.approximation$Accept

## High No High Yes1 High Yes2   
## 4 2 3

print(" Górne:")

## [1] " Górne:"

roughset$upper.approximation$Accept

## High No High Yes1 High Yes2 Medium Yes1 Medium Yes2   
## 4 2 3 1 7

print("")

## [1] ""

print("Przybliżenia zbioru Reject")

## [1] "Przybliżenia zbioru Reject"

print(" Dolne:")

## [1] " Dolne:"

roughset$lower.approximation$Reject

## Low No Low Yes1 Low Yes2   
## 8 5 6

print(" Górne:")

## [1] " Górne:"

roughset$upper.approximation$Reject

## Low No Low Yes1 Low Yes2 Medium Yes1 Medium Yes2   
## 8 5 6 1 7

# 4 Obszar B-pozytywny

Obszar B-pozytywny to zbiór zawierający tylko te obiekty, które leżą w **dowolnych dolnych przybliżeniach** (mają decyzję pewną). Im większy ten zbiór, tym dokładniejszy system decyzyjny, optymalnie, gdy liczba obiektów w obszarze jest równa liczbie obiektów w (całym zbiorze, uniwersum).

## 4.1 Ocena obszaru pozytywnego

Oceń dokładność przybliżeń, jako odsetek obiektów z całej tablicy decyzyjnej, które znalazły się w obszarze B-pozytywnym (ręcznie policzyć można jako stosunek długości wektora region.RST$positive.reg i liczby wierszy w decision.table).

region.RST <- BC.positive.reg.RST(decision.table, roughset)  
print("Region B-pozytywny, wszystkie obiekty o jednoznacznej decyzji, tj. znajdujące się w którymkolwiek przybliżeniu dolnym:")

## [1] "Region B-pozytywny, wszystkie obiekty o jednoznacznej decyzji, tj. znajdujące się w którymkolwiek przybliżeniu dolnym:"

region.RST$positive.reg

## [1] 2 3 4 5 6 8

dokladnosc=region.RST$degree.dependency  
sprintf("Odsetek obiektów z decyzją pewną: %.2f",dokladnosc)

## [1] "Odsetek obiektów z decyzją pewną: 0.75"

Powyższe kroki to przygotowanie do wygenerowania reguł decyzyjnych, które operują na klasach abstrakcji. Obszar B-pozytywny pozwala wstępnie oszacować jak wysoka może być trafność klasyfikacji uzyskana z reguł w kolejnych etapach.

# 5 Generowanie reguł decyzyjnych

Poniżej przedstawiono w całości proces analizy tablicy decyzyjnej:

* określenie cech nominalnych,
* określenie cechy decyzyjnej,
* podział na dane treningowe i testowe,
* dyskretyzację,
* wyznaczenie reduktu i reguł.

Ostatecznie reguły wykorzystane zostają do klasyfikacji obiektów ze zbioru testowego.

## należy utworzyć tablicę decyzyjną, wskazując, że cechą decyzyjną jest nr 14  
idx.tra=sample(dim(RoughSetData$wine.dt)[1],dim(RoughSetData$wine.dt)[1]\*0.65)  
wine.decTable <- SF.asDecisionTable(dataset = RoughSetData$wine.dt[idx.tra, ], decision.attr = 14, indx.nominal = 14)  
  
## wybrany zostaje też zbiór do testu i zapamiętane oryginalne wartości klas  
tst.wine <- SF.asDecisionTable(dataset = wine.decTable[-idx.tra, -ncol(wine.decTable)])  
tst.classes = wine.decTable[-idx.tra,ncol(wine.decTable)]

Dane zostają poddane dyskretyzacji, podziałowi na przedziały, które maksymalizują rozróżnialność obiektów różnych klas (tylko w oparciu o dane uczące). Pierwszym krokiem jest wyznaczyć **miejsca cięć** dziedzin. Cechy, które nie mają podanych cięć (numeric(0)) zostały na tym etapie wyeliminowane. Nastąpiła w ten sposób pierwsza selekcja cech przydatnych dla podejmowania trafnych decyzji.

## Dane ciągłe wymagają dyskretyzacji, wybrany w tym celu przykładowy algorytm "global.discernibility"  
cut.values.tra <- D.discretization.RST(wine.decTable, type.method = "global.discernibility")  
str(cut.values.tra$cut.values)

## List of 13  
## $ alcohol : num 12.8  
## $ malid\_acid : num(0)   
## $ ash : num(0)   
## $ alcalinity\_of\_ash : num(0)   
## $ magnesium : num(0)   
## $ total\_phenols : num(0)   
## $ flavanoids : num 1.38  
## $ nonflavanoid\_phenols: num(0)   
## $ proanthocyanins : num(0)   
## $ color\_intensity : num 3.65  
## $ hue : num(0)   
## $ od : num(0)   
## $ proline : num 676

Cięcia z algorytmu zostają wykorzystane do wygenerowania nowych tabel z danymi, tym razem w pełni dyskretnymi i tylko z wybranymi cechami.

Kolejnym krokiem selekcji cech jest wyliczenie reduktu (patrz wykład). Często prowadzi on do dalszego usunięcia nieistotnych cech.

## generowanie reduktu w oparciu o dane dyskretne, wybrana metoda "quickreduct"  
red.rst <- FS.feature.subset.computation(d.new.tra.rst, method = "quickreduct.rst")  
  
## zastosowanie listy cech zawartej w redukcie do wygenerowania zestawu reguł:  
rules.rst <- RI.indiscernibilityBasedRules.RST(d.new.tra.rst, red.rst)  
summary(rules.rst)

## The type of the considered model:   
## [1] "RST"  
## The type of the considered method:   
## [1] "indiscernibilityBasedRules"  
## A set consisting of 14 rules:  
## 1. IF alcohol is [-Inf,12.8) and flavanoids is [-Inf,1.38) and color\_intensity is [-Inf,3.65) and proline is [-Inf,676) THEN class is 2;  
## (supportSize=4; laplace=0.7143)   
## 2. IF alcohol is [-Inf,12.8) and flavanoids is [-Inf,1.38) and color\_intensity is [-Inf,3.65) and proline is [676, Inf] THEN class is 2;  
## (supportSize=2; laplace=0.6)   
## 3. IF alcohol is [-Inf,12.8) and flavanoids is [-Inf,1.38) and color\_intensity is [3.65, Inf] and proline is [-Inf,676) THEN class is 3;  
## (supportSize=2; laplace=0.6)   
## 4. IF alcohol is [-Inf,12.8) and flavanoids is [-Inf,1.38) and color\_intensity is [3.65, Inf] and proline is [676, Inf] THEN class is 3;  
## (supportSize=3; laplace=0.6667)   
## 5. IF alcohol is [-Inf,12.8) and flavanoids is [1.38, Inf] and color\_intensity is [-Inf,3.65) and proline is [-Inf,676) THEN class is 2;  
## (supportSize=27; laplace=0.9333)   
## 6. IF alcohol is [-Inf,12.8) and flavanoids is [1.38, Inf] and color\_intensity is [-Inf,3.65) and proline is [676, Inf] THEN class is 2;  
## (supportSize=3; laplace=0.6667)   
## 7. IF alcohol is [-Inf,12.8) and flavanoids is [1.38, Inf] and color\_intensity is [3.65, Inf] and proline is [-Inf,676) THEN class is 2;  
## (supportSize=5; laplace=0.75)   
## 8. IF alcohol is [-Inf,12.8) and flavanoids is [1.38, Inf] and color\_intensity is [3.65, Inf] and proline is [676, Inf] THEN class is 2;  
## (supportSize=2; laplace=0.6)   
## 9. IF alcohol is [12.8, Inf] and flavanoids is [-Inf,1.38) and color\_intensity is [-Inf,3.65) and proline is [676, Inf] THEN class is 2;  
## (supportSize=1; laplace=0.5)   
## 10. IF alcohol is [12.8, Inf] and flavanoids is [-Inf,1.38) and color\_intensity is [3.65, Inf] and proline is [-Inf,676) THEN class is 3;  
## (supportSize=17; laplace=0.9)   
## ... and 4 other rules.

W powyższej liście reguł są dwie uzupełniające informacje: 1. *support* - ile obiektów w zbiorze treningowym potwierdza daną regułę, 2. *laplace* - waga reguły, która zostanie użyta w procesie głosowania:

gdzie to oceniana reguła z konkluzją wskazującą na klasę , miara to liczba obiektów klasy sklasyfikownaych poprawnie, to liczba obiektów pasujących do reguły (bez względu na ich klasy), to liczba wszystkich klas.

Uzyskane reguły stosowane są do części testowej (już poddanej dyskretyzacji).

## predicting newdata  
res.1 <- predict(rules.rst, d.new.tst.rst)  
wynik = sum(as.array(tst.classes) == t(res.1))/length(tst.classes)\*100  
sprintf("Wynik klasyfikacji: %.1f proc.",wynik)

## [1] "Wynik klasyfikacji: 100.0 proc."

## 5.1 Generowanie i zastosowanie reguł na własnym zbiorze danych

Cały powyższy przykład, polegający na przetworzeniu zbioru wine.dt, dostosować można do własnego zbioru. Należy pamiętać, że niezbędne kroki do wykonania to: określenie cech nominalnych, cechy decyzyjnej, następnie podział na dane treningowe i testowe, dyskretyzacja, wyznaczenie reduktu i reguł. Ostatecznie reguły wykorzystane powinny być do klasyfikacji obiektów ze zbioru testowego.

# 6 Algorytmy generowania reguł

Dla przykładu powyżej, reguły generowane są w oparciu o wszystkie wybrane ręcznie kolumny.

Metoda RI.name.RST (ang. RI - rule induction) pozwala wybrać inny sposób generowania reguł. W powyższym kodzie reguły tworzone były na podstawie wszystkich przypadków z tabeli i testowane także na wszystkich. Powstało kilka reguł, które można poddać samodzielnej interpretacji.

Wygenerowane reguły (w zależności od wybranej metody RI.name.RST) uwzględniają kilka lub wszystkie wiersze w macierzy oraz wszystkie lub kilka z cech wejściowych. Miara *support* to liczba obiektów pasujących do reguły, wartość *laplace* to miara ufności reguły (patrz: wykład).

library(RoughSets)  
data(RoughSetData)  
## hiring.dt to tabela z danymi o kandydatach do pracy, przedstawiona na wykładzie:  
decision.table <- RoughSetData$hiring.dt  
  
## wybór cech, które skutkują podziałem uniwersum na zbiory elementarne:  
## cechy od 1 do 4 są opisem kandydata, cecha 5 jest decyzją  
cechy <- c(1,2)  
  
## relacja równoważności, zwraca numery obiektów, które są tożsame pod względem wybranych cech  
IND <- BC.IND.relation.RST(decision.table, feature.set = cechy)  
## niektóre klasy abstrakcji są więcej niż 1-elementowe  
print("Klasy abstrakcji powstałe w wyniku wybrania cech:")

## [1] "Klasy abstrakcji powstałe w wyniku wybrania cech:"

(names(decision.table)[cechy])

## [1] "Diploma" "Experience"

str(IND$IND.relation) #numery obiektów, które są nierozróżnialne z powodu przyjętych cech. str() wyświetla listę: nazwy wartości cech i numery obiektów, które są nierozróżnialne

## List of 6  
## $ MBA High : int 4  
## $ MBA Low : int 5  
## $ MBA Medium: int 1  
## $ MCE Low : int [1:2] 6 8  
## $ MSc High : int [1:2] 2 3  
## $ MSc Medium: int 7

decision.table.part=decision.table[,c(cechy,5)] #cecha nr 5 zawsze musi być dodana, gdyż jest cechą decyzyjną, wymaganą w tablicy  
decision.table.part = SF.asDecisionTable(dataset = decision.table.part,   
 decision.attr = dim(decision.table.part)[2] #ostatni jest decyzją  
 )  
rules = RI.AQRules.RST(decision.table = decision.table.part)  
rules

## A set consisting of 4 rules:  
## 1. IF Experience is High THEN Decision is Accept;  
## (supportSize=3; laplace=0.8)   
## 2. IF Diploma is MBA and Experience is Medium THEN Decision is Accept;  
## (supportSize=1; laplace=0.6667)   
## 3. IF Experience is Low THEN Decision is Reject;  
## (supportSize=3; laplace=0.8)   
## 4. IF Diploma is MSc and Experience is Medium THEN Decision is Reject;  
## (supportSize=1; laplace=0.6667)

results = predict(rules, decision.table.part)  
# możliwe zastosowanie innych metod głosowania  
#pred.vals1 <- predict(rules, data.tst, votingMethod = X.laplace)  
#pred.vals1 <- predict(rules, data.tst, votingMethod = X.rulesCounting)  
  
## sprawdzanie średniej accuracy  
error = sum(results != data.frame(decision.table.part[,"Decision"]))/dim(results)[1]\*100  
print(sprintf("Błąd wynosi: %0.f proc.",error))

## [1] "Błąd wynosi: 0 proc."

## 6.1 Modyfikacje algorytmu

Należy w powyższym przykładzie zwrócić uwagę na klasy abstrakcji, które są konsekwencją doboru kolumn (cech) a następnie wpływają na możliwość rozróżniania obiektów różnych klas. W tym celu zmieniać można cechy brane pod uwagę (zmienna cechy w kodzie powyżej).

Ponadto zmienić można algorytm generowania reguł na następujące:

# RI.AQRules.RST(decision.table = decision.table.part, confidence = 1, timesCovered = 1)  
# RI.CN2Rules.RST(decision.table = decision.table.part, K = 3)  
# RI.LEM2Rules.RST(decision.table = decision.table.part)  
# RI.indiscernibilityBasedRules.RST(decision.table = decision.table.part, feature.set)

Podane powyżej argumenty o domyślnej wartości mogą być pomijane. W metodzie AQ *confidence* oznacza generowanie bazy reguł o łącznej ufności co najmniej o zadanej wartości. W metodzie AQ *timesCovered* oznacza generowanie zadanej liczby reguł do każdego przypadku ze zbioru uczącego. Jeśli liczba cech jest niewielka, to powstają reguły powielone w celu spełnienia liczności zadanej argumentem timesCovered.

RI.AQRules.RST(decision.table = decision.table.part, confidence = 1, timesCovered = 3)

## A set consisting of 12 rules:  
## 1. IF Experience is High THEN Decision is Accept;  
## (supportSize=3; laplace=0.8)   
## 2. IF Diploma is MBA and Experience is Medium THEN Decision is Accept;  
## (supportSize=1; laplace=0.6667)   
## 3. IF Experience is Medium and Diploma is MBA THEN Decision is Accept;  
## (supportSize=1; laplace=0.6667)   
## 4. IF Experience is High THEN Decision is Accept;  
## (supportSize=3; laplace=0.8)   
## 5. IF Diploma is MBA and Experience is Medium THEN Decision is Accept;  
## (supportSize=1; laplace=0.6667)   
## 6. IF Experience is High THEN Decision is Accept;  
## (supportSize=3; laplace=0.8)   
## 7. IF Diploma is MSc and Experience is Medium THEN Decision is Reject;  
## (supportSize=1; laplace=0.6667)   
## 8. IF Experience is Low THEN Decision is Reject;  
## (supportSize=3; laplace=0.8)   
## 9. IF Experience is Low THEN Decision is Reject;  
## (supportSize=3; laplace=0.8)   
## 10. IF Experience is Medium and Diploma is MSc THEN Decision is Reject;  
## (supportSize=1; laplace=0.6667)   
## ... and 2 other rules.

W metodzie CN2 argument *K* oznacza ile najlepszych reguł w każdej iteracji zostanie rozbudowanych o wszystkie dostępne przesłanki.

RI.CN2Rules.RST(decision.table = decision.table.part, K = 3)

## A set consisting of 4 rules:  
## 1. IF Experience is High THEN Decision is Accept;  
## (supportSize=3; laplace=0.8)   
## 2. IF Experience is Low THEN Decision is Reject;  
## (supportSize=3; laplace=0.8)   
## 3. IF Diploma is MBA THEN Decision is Accept;  
## (supportSize=1; laplace=0.6667)   
## 4. IF Diploma is MSc THEN Decision is Reject;  
## (supportSize=1; laplace=0.6667)

# 7 Algorytmy generowania reduktów

W przypadku zbiorów danych z bardzo dużą liczbą cech, niepraktyczne jest wybieranie cech (kolumn tabeli) ręcznie.

Poniższy przykład prezentuje możliwość wykorzystania różnych metod generowania reduktów, czyli wyliczania minimalnych podzbiorów tych cech, które zapewniają rozróżnialność obiektów różnych klas.

Dane treningowe i testowe zostają przygotowane w odpowiedni sposób i poddane dyskretyzacji:

## zbiór danych wine posiada dane ciągłe, double, kolumny od 1 do 13,  
## oraz kolumnę 14 - decyzję, klasyfikującą obiekt do jednego z trzech typów  
  
## należy utworzyć tablicę decyzyjną, wskazując, że cechą decyzyjną jest nr 14  
idx.tra=sample(dim(RoughSetData$wine.dt)[1],dim(RoughSetData$wine.dt)[1]\*0.65)  
wine.decTable <- SF.asDecisionTable(dataset = RoughSetData$wine.dt[idx.tra, ], decision.attr = 14, indx.nominal = 14)  
  
## wybrany zostaje też zbiór do testu i zapamiętane oryginalne wartości klas  
tst.wine <- SF.asDecisionTable(dataset = wine.decTable[-idx.tra, -ncol(wine.decTable)])  
tst.classes = wine.decTable[-idx.tra,ncol(wine.decTable)]  
  
## Dane ciągłe wymagają dyskretyzacji, wybrany w tym celu przykładowy algorytm "global.disc"  
cut.values.tra <- D.discretization.RST(wine.decTable, type.method = "global.discernibility")  
str(cut.values.tra$cut.values)

## List of 13  
## $ alcohol : num 12.8  
## $ malid\_acid : num(0)   
## $ ash : num(0)   
## $ alcalinity\_of\_ash : num(0)   
## $ magnesium : num(0)   
## $ total\_phenols : num(0)   
## $ flavanoids : num 1.23  
## $ nonflavanoid\_phenols: num(0)   
## $ proanthocyanins : num(0)   
## $ color\_intensity : num(0)   
## $ hue : num 0.83  
## $ od : num 2.19  
## $ proline : num 604

## generate new decision table  
d.new.tra.rst <- SF.applyDecTable(wine.decTable, cut.values.tra)  
d.new.tst.rst <- SF.applyDecTable(tst.wine, cut.values.tra)

Następnie wskazać należy metodę generowania reduktu (patrz: wykład) i na jego podstawie pozyskać bazę reguł metodą RI.indiscernibilityBasedRules.RST, która jako jedyna, bierze pod uwagę redukt.

## generowanie reduktu w oparciu o dane dyskretne, wybrana metoda "quickreduct"  
#red.rst <- FS.feature.subset.computation(d.new.tra.rst, method = "quickreduct.rst")  
red.rst <- FS.feature.subset.computation(d.new.tra.rst, method = "greedy.heuristic.superreduct")  
  
## zastosowanie listy cech zawartej w redukcie do wygenerowania zestawu reguł:  
rules.rst <- RI.indiscernibilityBasedRules.RST(d.new.tra.rst, red.rst)  
rules.rst

## A set consisting of 23 rules:  
## 1. IF alcohol is [-Inf,12.8) and flavanoids is [-Inf,1.23) and hue is [-Inf,0.83) and od is [-Inf,2.19) and proline is [-Inf,604) THEN class is 3;  
## (supportSize=4; laplace=0.7143)   
## 2. IF alcohol is [-Inf,12.8) and flavanoids is [-Inf,1.23) and hue is [-Inf,0.83) and od is [-Inf,2.19) and proline is [604, Inf] THEN class is 3;  
## (supportSize=4; laplace=0.7143)   
## 3. IF alcohol is [-Inf,12.8) and flavanoids is [-Inf,1.23) and hue is [0.83, Inf] and od is [-Inf,2.19) and proline is [-Inf,604) THEN class is 2;  
## (supportSize=1; laplace=0.5)   
## 4. IF alcohol is [-Inf,12.8) and flavanoids is [-Inf,1.23) and hue is [0.83, Inf] and od is [-Inf,2.19) and proline is [604, Inf] THEN class is 2;  
## (supportSize=1; laplace=0.5)   
## 5. IF alcohol is [-Inf,12.8) and flavanoids is [-Inf,1.23) and hue is [0.83, Inf] and od is [2.19, Inf] and proline is [604, Inf] THEN class is 2;  
## (supportSize=1; laplace=0.5)   
## 6. IF alcohol is [-Inf,12.8) and flavanoids is [1.23, Inf] and hue is [-Inf,0.83) and od is [-Inf,2.19) and proline is [-Inf,604) THEN class is 2;  
## (supportSize=1; laplace=0.5)   
## 7. IF alcohol is [-Inf,12.8) and flavanoids is [1.23, Inf] and hue is [-Inf,0.83) and od is [2.19, Inf] and proline is [-Inf,604) THEN class is 2;  
## (supportSize=2; laplace=0.6)   
## 8. IF alcohol is [-Inf,12.8) and flavanoids is [1.23, Inf] and hue is [-Inf,0.83) and od is [2.19, Inf] and proline is [604, Inf] THEN class is 2;  
## (supportSize=1; laplace=0.5)   
## 9. IF alcohol is [-Inf,12.8) and flavanoids is [1.23, Inf] and hue is [0.83, Inf] and od is [-Inf,2.19) and proline is [-Inf,604) THEN class is 2;  
## (supportSize=3; laplace=0.6667)   
## 10. IF alcohol is [-Inf,12.8) and flavanoids is [1.23, Inf] and hue is [0.83, Inf] and od is [2.19, Inf] and proline is [-Inf,604) THEN class is 2;  
## (supportSize=25; laplace=0.9286)   
## ... and 13 other rules.

## predicting newdata  
res.1 <- predict(rules.rst, d.new.tst.rst)  
wynik = sum(as.array(tst.classes) == t(res.1))/length(tst.classes)\*100  
sprintf("Wynik klasyfikacji: %.1f proc.",wynik)

## [1] "Wynik klasyfikacji: 100.0 proc."

# 8 Oczyszczanie danych i wykorzystanie teorii zbiorów przybliżonych

W poniższym przykładzie należy zadawać algorytmy dyskretyzacji, liczbę cięć (dla algorytmów, które na to pozwalają) i metodę wyznaczania reduktu. Następnie automatycznie generowane są wszystkimi dostępnymi algorytmami bazy reguł i wykorzystywane na zbiorze testowym.

Wybrany zbiór danych to dane pacjentów cukrzycowych w plemieniu Indian Pima.

# Odkomentować, a po pierwszym uruchomieniu zakomentować znakiem #  
#install.packages("mlbench")  
#install.packages("RoughSets")  
library(mlbench)  
library(RoughSets)  
data(PimaIndiansDiabetes)  
dataset=PimaIndiansDiabetes  
summary(dataset)

## pregnant glucose pressure triceps   
## Min. : 0.000 Min. : 0.0 Min. : 0.00 Min. : 0.00   
## 1st Qu.: 1.000 1st Qu.: 99.0 1st Qu.: 62.00 1st Qu.: 0.00   
## Median : 3.000 Median :117.0 Median : 72.00 Median :23.00   
## Mean : 3.845 Mean :120.9 Mean : 69.11 Mean :20.54   
## 3rd Qu.: 6.000 3rd Qu.:140.2 3rd Qu.: 80.00 3rd Qu.:32.00   
## Max. :17.000 Max. :199.0 Max. :122.00 Max. :99.00   
## insulin mass pedigree age diabetes   
## Min. : 0.0 Min. : 0.00 Min. :0.0780 Min. :21.00 neg:500   
## 1st Qu.: 0.0 1st Qu.:27.30 1st Qu.:0.2437 1st Qu.:24.00 pos:268   
## Median : 30.5 Median :32.00 Median :0.3725 Median :29.00   
## Mean : 79.8 Mean :31.99 Mean :0.4719 Mean :33.24   
## 3rd Qu.:127.2 3rd Qu.:36.60 3rd Qu.:0.6262 3rd Qu.:41.00   
## Max. :846.0 Max. :67.10 Max. :2.4200 Max. :81.00

# boxplot(dataset) #te same dame co summary, można zaprezentować graficznie w formie wykresów pudło-wąsy

Zapoznając się z nowymi danymi, należy zwrócić uwagę na ich poprawność, rozumianą w sposób intuicyjny lub wynikający ze znajomości dyscypliny. W powyższym przypadku glucose, pressure, triceps, insulin, mass nie mogą być równe zero. Wartości 0 zostają zastąpione NA.

cechy.zero=c("glucose", "pressure", "triceps", "insulin", "mass")  
cecha="glucose"  
for (cecha in cechy.zero) {  
 dataset[which(dataset[,cecha]==0),  
 cecha] = NA  
}  
dataset <- SF.asDecisionTable(dataset = dataset,  
 decision.attr = 9,  
 indx.nominal = 9)  
summary(dataset)

## pregnant glucose pressure triceps   
## Min. : 0.000 Min. : 44.0 Min. : 24.00 Min. : 7.00   
## 1st Qu.: 1.000 1st Qu.: 99.0 1st Qu.: 64.00 1st Qu.:22.00   
## Median : 3.000 Median :117.0 Median : 72.00 Median :29.00   
## Mean : 3.845 Mean :121.7 Mean : 72.41 Mean :29.15   
## 3rd Qu.: 6.000 3rd Qu.:141.0 3rd Qu.: 80.00 3rd Qu.:36.00   
## Max. :17.000 Max. :199.0 Max. :122.00 Max. :99.00   
## NA's :5 NA's :35 NA's :227   
## insulin mass pedigree age diabetes   
## Min. : 14.00 Min. :18.20 Min. :0.0780 Min. :21.00 neg:500   
## 1st Qu.: 76.25 1st Qu.:27.50 1st Qu.:0.2437 1st Qu.:24.00 pos:268   
## Median :125.00 Median :32.30 Median :0.3725 Median :29.00   
## Mean :155.55 Mean :32.46 Mean :0.4719 Mean :33.24   
## 3rd Qu.:190.00 3rd Qu.:36.60 3rd Qu.:0.6262 3rd Qu.:41.00   
## Max. :846.00 Max. :67.10 Max. :2.4200 Max. :81.00   
## NA's :374 NA's :11

na2common = MV.mostCommonVal(dataset)  
dataset.na2common = SF.applyDecTable(dataset,na2common)  
summary(dataset.na2common)

## pregnant glucose pressure triceps   
## Min. : 0.000 Min. : 44.00 Min. : 24.00 Min. : 7.00   
## 1st Qu.: 1.000 1st Qu.: 99.75 1st Qu.: 64.00 1st Qu.:25.00   
## Median : 3.000 Median :117.00 Median : 72.20 Median :29.15   
## Mean : 3.845 Mean :121.69 Mean : 72.41 Mean :29.15   
## 3rd Qu.: 6.000 3rd Qu.:140.25 3rd Qu.: 80.00 3rd Qu.:32.00   
## Max. :17.000 Max. :199.00 Max. :122.00 Max. :99.00   
## insulin mass pedigree age diabetes   
## Min. : 14.0 Min. :18.20 Min. :0.0780 Min. :21.00 neg:500   
## 1st Qu.:121.5 1st Qu.:27.50 1st Qu.:0.2437 1st Qu.:24.00 pos:268   
## Median :155.5 Median :32.40 Median :0.3725 Median :29.00   
## Mean :155.5 Mean :32.46 Mean :0.4719 Mean :33.24   
## 3rd Qu.:155.5 3rd Qu.:36.60 3rd Qu.:0.6262 3rd Qu.:41.00   
## Max. :846.0 Max. :67.10 Max. :2.4200 Max. :81.00

dataset=dataset.na2common #nadpisujemy wcześniejszy zbiór, teraz dataset zawiera dane oczyszczone

Po wypełnieniu niewłaściwych wartości wartościami średnimi cechy z całej z populacji, należy zadbać o równoliczność obiektów w obu klasach.

# klasa neg jest zbyt liczna w porównaniu do pos. Spowoduje to tendencyjność odpowiedzi, bias w kierunku odpowiedzi "neg"  
summary(dataset[,"diabetes"])

## neg pos   
## 500 268

# View(PimaIndiansDiabetes)

# przygotuj dane w losowej kolejności, korygując jednocześnie brak równoliczności klas  
set.seed(14)  
# wszystkie przypadki neg i pos  
idx.neg = which(dataset[,"diabetes"]=="neg")  
idx.pos = which(dataset[,"diabetes"]=="pos")  
# liczba przypadków, która być powinna z każdej klasy:  
n.neg = length(idx.neg)  
n.pos = length(idx.pos)  
n.new = min(n.neg,n.pos)  
# indeksy losowo wybranych nowych obiektów  
idx.neg.new = sample(idx.neg,n.new)  
idx.pos.new = sample(idx.pos,n.new)  
  
tra.fraction = 0.95  
last=floor(tra.fraction\*(length(idx.neg.new)))  
# ostatecznie indeksy do treningu, reszta do testu  
idx.tra = c(idx.neg.new[1:last],  
 idx.pos.new[1:last])  
idx.tst = c(idx.neg.new[-(1:last)],  
 idx.pos.new[-(1:last)])  
# uzyskane zrównoważenie klas:  
summary(dataset[idx.tra,"diabetes"])

## neg pos   
## 254 254

summary(dataset[idx.tst,"diabetes"])

## neg pos   
## 14 14

# zbiory danych zostają utworzone:  
#   
dataset.tra <-SF.asDecisionTable(dataset[idx.tra,],  
 decision.attr = 9,  
 indx.nominal = 9)  
dataset.tst <- SF.asDecisionTable(dataset[idx.tst, -ncol(dataset)])  
# zapamietaj poprawne klasy:  
true.classes <- as.character(dataset[idx.tst, ncol(dataset)])

# wielokrotny trening z różnymi ustawieniami  
  
## Metoda dyskretyzacji: jedno z type.method odkomentowane:  
#type.method = "global.discernibility"  
type.method = "local.discernibility"  
#type.method = "unsupervised.intervals"  
#type.method = "unsupervised.quantiles"  
  
  
## Metoda wyliczania reduktu, jedno z method odkomentowane:  
#method = "permutation.heuristic" #działa tylko dla wcześniejszej dyskretyzacji met. global.disc  
method = "DAAR.heuristic"  
#method = "greedy.heuristic"  
  
## metoda niezwykle wolna, unikać:  
#method = "nearOpt.fvprs"  
  
  
  
verify = function(rules, data.tst) {  
 ## wnioskowanie  
 pred.vals1 <- predict(rules, data.tst)  
   
 # możliwe zastosowanie innych metod głosowania  
 #pred.vals1 <- predict(rules, data.tst,votingMethod = X.laplace)  
 #pred.vals1 <- predict(rules, data.tst,votingMethod = X.rulesCounting)  
   
 ## sprawdzanie średniej accuracy  
 result = (mean(pred.vals1 == true.classes))  
 return(result)  
}  
  
for (liczba\_przedzialow in 2:4){  
 ## dyskretyzacja:  
 cut.values = NULL  
 if((type.method=="local.discernibility")||(type.method=="global.discernibility")){  
 cut.values <- D.discretization.RST(dataset.tra,  
 type.method = type.method)  
 print("liczba\_przedzialow - dobierana automatycznie przez algorytm")  
 }else{  
 cut.values <- D.discretization.RST(dataset.tra,  
 type.method = type.method,  
 nOfIntervals = liczba\_przedzialow)  
 print(sprintf("liczba\_przedzialow = %d", liczba\_przedzialow))  
 }  
   
 data.tra <- SF.applyDecTable(dataset.tra, cut.values)  
 data.tst <- SF.applyDecTable(dataset.tst, cut.values)  
   
   
 rules <- RI.LEM2Rules.RST(data.tra)  
 print(sprintf("wynik dla LEM2: %1.3f %%", 100\*verify(rules,data.tst)))  
 rules <- RI.AQRules.RST(data.tra)  
 print(sprintf("wynik dla AQ: %1.3f %%", 100\*verify(rules,data.tst)))  
 rules <- RI.CN2Rules.RST(data.tra)  
 print(sprintf("wynik dla CN2: %1.3f %%", 100\*verify(rules,data.tst)))  
   
 reduct = NULL  
 try(reduct <- FS.reduct.computation(data.tra,  
 method = method),  
 silent = TRUE)  
   
 if(is.null(reduct)) {  
 print(sprintf("Przy zadanej metodzie dyskretyzacji: %s następuje błąd metody generowania reduktu: %s",type.method,method))  
 break  
 }  
  
 rules <- RI.indiscernibilityBasedRules.RST(data.tra,reduct)  
 print(sprintf("wynik dla IND: %1.3f %%", 100\*verify(rules,data.tst)))  
   
 # wyjdź z pętli for, jeśli jest to metoda, która sama dobiera liczbę przedziałów,  
 # nie rób doboru liczby przedziałów   
 if((type.method=="local.discernibility")||(type.method=="global.discernibility")) break  
}

## [1] "liczba\_przedzialow - dobierana automatycznie przez algorytm"  
## [1] "wynik dla LEM2: 60.714 %"  
## [1] "wynik dla AQ: 75.000 %"  
## [1] "wynik dla CN2: 71.429 %"  
## [1] "wynik dla IND: 71.429 %"

## 8.1 Modyfikacje algorytmu

W powyższym kodzie dobierać można metody dyskretyzacji i wyznaczania reduktu i wytypować konfigurację gwarantującą najwyższą dokładność klasyfikacji.

# 9 Podsumowanie

Przedstawione powyżej przykłady wykorzystać należy do analizy własnego zbioru danych. Jeśli dane są z dziedzin ciągłych należy je poddać dyskretyzacji. Następnie zastosować należy wybrany algorytm wyznaczania reduktu (lub kilka wariantów takich algorytmów), później algorytmy generowania reguł. Bazę reguł przetestować należy na podzbiorze danych testowych, które nie zostały użyte w krokach wcześniejszych.

