**Inteligencja Obliczeniowa**

Praca domowa nr 2 – Zgłębianie danych

Krzysztof Kulewski, 238149, grupa 1, 08.12.2018

## Opis bazy danych

Do realizacji zadania wybrano bazę ***Pima Indians Diabetes Database***, pochodzącą ze zbiorów *National Instutite of Diabetes and Digestive and Kidney Diseases*. Jest ona wynikiem badań przeprowadzonych na żeńskiej części populacji Indian z plemienia Pima, zamieszkujących stan Phoenix w USA. Plemię te jest pod ciągłą obserwacją z powodu wysokiego wskaźnika zachorowań na cukrzycę. Występowanie cukrzycy było określane na podstawie standardowych kryteriów *WHO*.   
  
Badana populacja składała się z **768** osób, które opisano za pomocą **9 atrybutów**:  
**- Pregnancies** – (liczba naturalna) – przebyte ciąże,  
**- Glucose** – (liczba naturalna) – poziom glukozy we krwi po 2 godzinach od zażycia dawki testowej,  
**- BloodPressure** – (liczba naturalna) – rozkurczowe ciśnienie krwi (mm/Hg),  
**- SkinThickness** – (liczba naturalna) – grubość fałdu skórnego zmierzonego na tricepsie (mm),  
**- Insulin** – (liczba naturalna) – poziom insuliny po 2 godzinach od zażycia dawki testowej,  
**- BMI** – (liczba zmiennoprzecinkowa) – indeks masy ciała, pośrednio opisuje zawartość tkanki tłuszczowej w organizmie na podstawie proporcji masy ciała mierzonej w kilogramach do wzrostu w metrach,  
**- DiabetesPedigreeFunction** (liczba zmiennoprzecinkowa) – funkcja wpływu genetycznego na występowanie cukrzycy (wyliczona na podstawie historia występowania cukrzycy u krewnych i stopień ich pokrewieństwa),  
**- Age** – (liczba całkowita) – wiek badanej osoby,  
**- Outcome** – (liczba binarna) –wynik badania, określający występowanie (1) cukrzycy lub jej brak (0).  
Atrybut ten wydaje się być naturalnym kandydatem na klasę zbioru danych.  
  
Celem projektu będzie stworzenie modelu, który na podstawie 8 atrybutów będzie w stanie trafinie przewidzieć wartość kolumny „Outcome”, czyli stwierdzić, czy dana osoba choruje na cukrzycę.

## Obróbka danych

Bardzo szybko możemy dostrzec, że w bazie są pewne niespójności. O ile w przypadku liczby ciąży wartość 0 jest jak najbardziej prawdopodobna, tak zdecydowanie nie jest to dobra wartość dla kolumn *Glucose*, *BloodPressure*, *SkinThickness* itd.

W takim przypadku możemy albo zignorować rekordy, które zawierają nieprawidłowe wartości, albo uzupełnić je wartościami średnimi. Baza stanowiąca przedmiot zadania jest relatywnie mała, stąd dużo lepsze będzie rozwiązanie drugie.  
  
Wyliczanie średnich wartości dla poszczególnych kolumn przebiegało według następującego schematu:

# baza jest załadowana do zmiennej ‘db’  
# wybieramy wartości większe od zera i tworzymy wektor  
non.zero.glucose = db$Glucose[which(db$Glucose>0)]  
# obliczamy średnią wartość  
glucose.mean = mean(non.zero.glucose)

Powyższą operację zastosowano dla kolumn: Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction i Age.  
  
Obliczone średnie zostały następnie zaokrąglone do odpowiedniej (wynikającej z danych) liczby miejsc po przecinku i podstawione w miejsca wystąpienia liczby 0:

# kopiujemy bazę do zmiennej db.nz (non-zero)   
db.nz = db  
# iterujemy po wszystkich wierszach  
for (i in 1:nrow(db))  
{  
 # jeśli wartość wynosi 0 – podstawiamy zaokrągloną średnią  
 if (db$Glucose[i] == 0) db.nz$Glucose[i] = round(glucose.mean)  
 if (db$BloodPressure[i] == 0) db.nz$BloodPressure[i] = round(bloodPressure.mean)  
 if (db$SkinThickness[i] == 0) db.nz$SkinThickness[i] = round(skinThickness.mean)  
 if (db$Insulin[i] == 0) db.nz$Insulin[i] = round(insulin.mean)  
 if (db$BMI[i] == 0) db.nz$BMI[i] = round(bmi.mean, digits = 1)  
 if (db$DiabetesPedigreeFunction[i] == 0) db.nz$DiabetesPedigreeFunction[i] =  
round(dpf.mean, digits = 3)  
 if (db$Age[i] == 0) db.nz$Age[i] = round(age.mean)  
}

Kolejnym krokiem jest normalizacja kolumn oraz zamiana wartości ‘0’ i ‘1’ na ‘healthy’ i ‘sick’:

# zmieniamy wartości kolumny Outcome z ‘0’ i ‘1’ na ‘healthy’ i ‘sick’  
db.nz$Outcome = factor(db.nz$Outcome, level=0:1, labels=c("healthy", "sick"))  
# normalizujemy dane numeryczne  
fnorm = function(x) { return ((x-min(x))/(max(x)-min(x))) }  
db.nz$Pregnancies = fnorm(db.nz$Pregnancies)  
db.nz$Glucose = fnorm(db.nz$Glucose)  
db.nz$BloodPressure = fnorm(db.nz$BloodPressure)  
db.nz$SkinThickness = fnorm(db.nz$SkinThickness)  
db.nz$Insulin = fnorm(db.nz$Insulin)  
db.nz$BMI = fnorm(db.nz$BMI)  
db.nz$DiabetesPedigreeFunction = fnorm(db.nz$DiabetesPedigreeFunction)  
db.nz$Age = fnorm(db.nz$Age)

## Klasyfikatory

Do klasyfikacji użyte zostały 4 metody – drzewa decyzyjne, K najbliższych sąsiadów (kNN), algorytm Naive-Bayes oraz Lasy Losowe.

Przed rozpoczęciem klasyfikacji, zbiór podzielony został na:  
- dane treningowe – wykorzystane do uczenia klasyfikatora,  
- dane testowe – wykorzystane do oceny klasyfikatora.

# Podzial na grupe treningowa i testowa   
set.seed(1234)  
ind <- sample(2, nrow(db.nz.norm), replace=TRUE, prob=c(0.8, 0.2))  
db.training <- db.nz.norm[ind==1, 1:9]  
db.test <- db.nz.norm[ind==2, 1:9]

Następnie utworzono klasyfikatory (używając danych treningowych) oraz oceniono je (używając danych testowych).   
Dla każdego z algorytmów zestawiono macierz błędów oraz wyliczono dokładność.

## 3.1 Klasyfikacja drzewami decyzyjnymi (paczka party)   
db.ctree = ctree(Outcome ~ ., data = db.training)  
ctr.predicted = predict(db.ctree, db.test[,1:8])   
ctr.conf.matrix = table(ctr.predicted, db.test[,9])  
ctr.accuracy = sum(diag(ctr.conf.matrix)) / sum(ctr.conf.matrix)

## 3.2 Klasyfikacja KNN (paczka class)   
db.knn3 = knn(db.training[,1:8], db.test[,1:8], cl=db.training[,9], k = 3, prob=FALSE)  
knn.predicted = db.knn3  
knn.conf.matrix = table(knn.predicted, db.test[,9])  
knn.accuracy = sum(diag(knn.conf.matrix)) / sum(knn.conf.matrix)

## 3.3 Klasyfikacja Naive-Bayes (paczka e1071)   
db.naiveBayes = naiveBayes(db.training[,1:8], db.training[,9])  
nbs.predicted = predict(db.naiveBayes, db.test[,1:8])   
nbs.conf.matrix = table(nbs.predicted, db.test[,9])  
nbs.accuracy = sum(diag(nbs.conf.matrix)) / sum(nbs.conf.matrix)

## 3.4 Klasyfikacja Lasy Losowe (paczka randomForest)  
db.rfo = randomForest(Outcome ~ ., data=db.training)  
rfo.predicted = predict(db.rfo, db.test[,1:8])  
rfo.conf.matrix = table(rfo.predicted, db.test[,9])  
rfo.accuracy = sum(diag(rfo.conf.matrix)) / sum(rfo.conf.matrix)

> ctr.conf.matrix

ctr.real

ctr.predicted healthy sick

healthy 90 28

sick 13 31

> knn.conf.matrix

knn.real

knn.predicted healthy sick

healthy 83 25

sick 20 34

> nbs.conf.matrix

nbs.real

nbs.predicted healthy sick

healthy 89 21

sick 14 38

> rfo.conf.matrix

rfo.real

rfo.predicted healthy sick

healthy 91 24

sick 12 35

> ctr.accuracy

[1] 0.7469136

> knn.accuracy

[1] 0.7222222

> nbs.accuracy

[1] 0.7839506

> rfo.accuracy

[1] 0.7777778

## Źródła

1. Dane benchmarkowe  
<https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html>  
  
2. Dokumentacja biblioteki GenAlg  
<https://cran.r-project.org/web/packages/genalg/genalg.pdf>

## Załączniki

1. Pliki źródłowe w języku R: fitness.r, parser.r, program.r  
2. Formuły CNF w formacie DIMACS  
3. Treść zadania (PDF)