Inteligencja Obliczeniowa
Praca domowa nr 2 – Zgłębianie danych

Krzysztof Kulewski, 238149, grupa 1, 08.12.2018

1. 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.

2. 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)
```

3. 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)
```

Wylosowany zbiór testowy to populacja o następujących parametrach:

Populacja	
Stan pozytywny (sick)	Stan negatywny (healthy)
103	59

Wyniki dla poszczególnych klasyfikatorów przedstawiono za pomocą **macierzy błędu** i trzech liczb:
- **ACC** (Accuracy) – dokładność,
- **TPR** (True Positive Rate) – czułość (prawdopodobieństwo wykrycia),
- **FPR** (False Positive Rate) – prawdopodobieństwo fałszywego alarmu.

Drzewa decyzyjne

		Klasa rzeczywista	
	Populacja	Stan poz.	Stan neg.
Klasa predykowana	Klas. poz.	90	28
	Klas. neg.	13	31

ACC: 0.7469136 TPR: 0.8737864 FPR: 0.4745763

KNN

		Klasa rzeczywista	
	Populacja	Stan poz.	Stan neg.
Klasa predykowana	Klas. poz.	83	25
	Klas. neg.	20	34

ACC: 0.7222222 TPR: 0.8058252 FPR: 0.4237288

Naive-Bayes

		Klasa rzeczywista	
	Populacja	Stan poz.	Stan neg.
Klasa predykowana	Klas. poz.	89	21
	Klas. neg.	14	38

ACC: 0.7839506 TPR: 0.8640777 FPR: 0.3559322

Lasy Losowe

		Klasa rzeczywista	
	Populacja	Stan poz.	Stan neg.
Klasa predykowana	Klas. poz.	91	24
	Klas. neg.	12	35

ACC: 0.7777778 TPR: 0.8834951 FPR: 0.4067797

Używając powyższych macierzy błędów, wyliczyć możemy następujące wartości:
- **TP** (True Positive) – liczba osób poprawnie sklasyfikowanych jako chorzy,
- **FP** (False Positive) – liczba osób błędnie sklasyfikowanych jako chorzy (błąd pierwszego rodzaju),
- **TN** (True Negative) – liczba osób poprawnie sklasyfikowanych jako zdrowi,
- **FN** (False Negaive) – liczba osób błędnie sklasyfikowanych jako zdrowi (błąd drugiego rodzaju) .

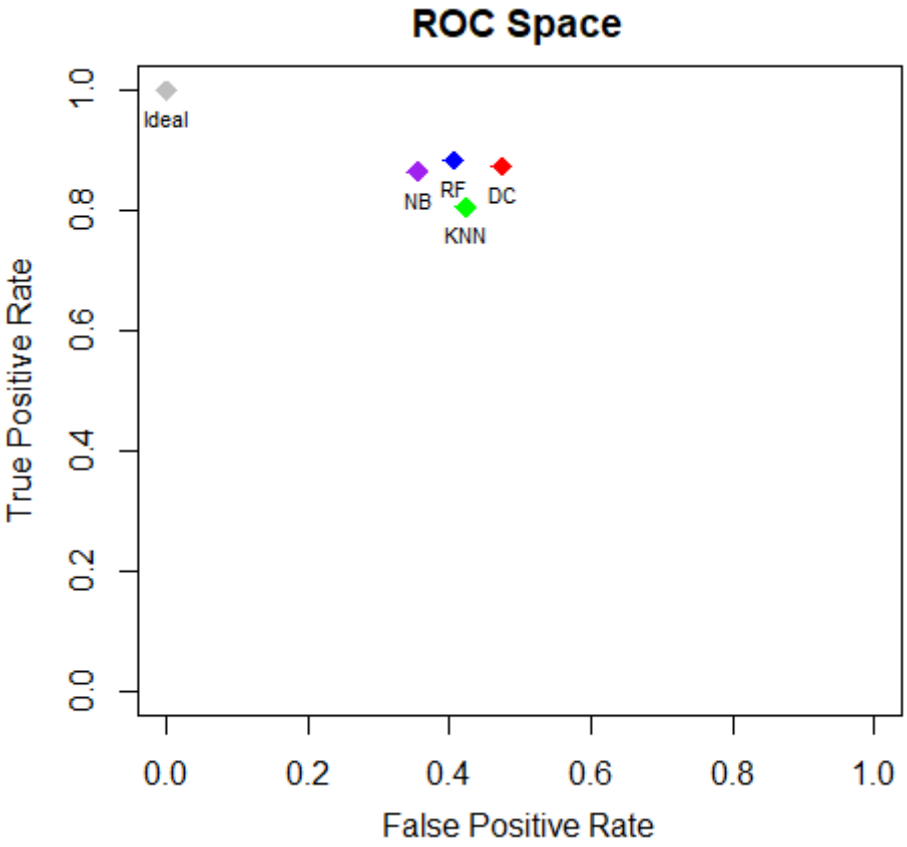
Wartości te posłużyć mogą do wyliczenia odsetek:
- **TPR** (True Positive Rate) – odsetek osób poprawnie sklasyfikowanych jako chorzy (czułość),
- **FPR** (False Positive Rate) – odsetek osób błędnie sklasyfikowanych jako chorzy (fałszywy alarm)
- **TNR** (True Negative Rate) – odsetek osób poprawnie sklasyfikowanych jako zdrowi, możliwy do wyliczenia ze wzoru **1 – TPR**.
- **FNR** (False Negative Rate) – odsetek osób błędnie sklasyfikowanych jako zdrowi, możliwy do wyliczenia ze wzoru **1 – FPR**.

W kontekście analizowanej bazy danych, **błąd pierwszego rodzaju (FP)** to sklasyfikowanie osoby zdrowej jako chora na cukrzycę. Rzeczywiste zbadanie takiej osoby powinno rozwiać wątpliwości, toteż realnym efektem błędu pierwszego rodzaju dla sklasyfikowanej osoby jest konieczność stawienia się na badaniu.

Błąd drugiego rodzaju (FN), czyli fałszywe wykluczenie choroby, wydaje się być dużo poważniejszy w skutkach. Osoba taka, będąc przekonana, że jest zdrowa, nie podejmie leczenia, co może skończyć się utratą zdrowia lub śmiercią.

Warto zauważyć, że:
- **zwiększenie FP** (błędy pierwszego rodzaju) powoduje **zwiększenie wartości FPR** oraz **zmniejszenie TNR**,
- **zwiększenie FN** (błędy drugiego rodzaju) powoduje **zwiększenie wartości FNR** oraz **zmniejszenie TPR.**

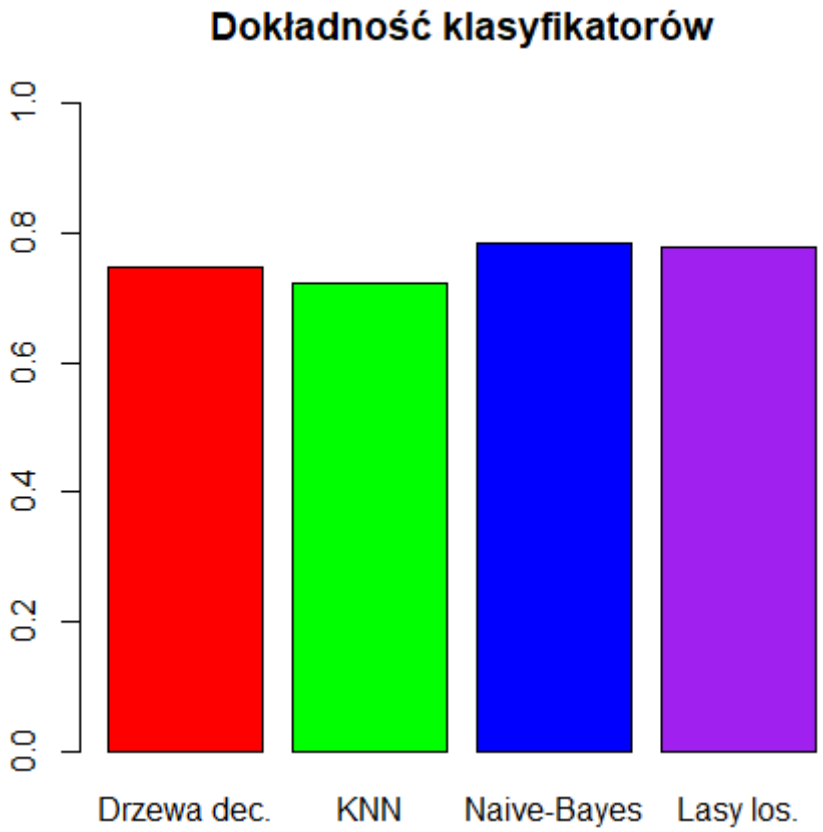
Poniższy wykres przedstawia zestawienie TPR i FPR dla każdego z czterech klasyfikatorów.



Na wykresie zaznaczono dodatkowo piąty punkt z etykietą Ideal, który reprezentuje idealny klasyfikator, nie popełniający błędów.
Najbliżej niego znajdują się klasyfikatory *Naive-Bayes* (NB) i *Lasy Losowe* (RF – Random Forest). Nieznacznie gorzej wypadł klasyfikator *Drzewa Decyzyjne*, a najgorzej - *KNN*.

Jeśli wziąć pod uwagę potencjalne skutki błędów pierwszego i drugiego rodzaju, powinniśmy skupić się na znalezieniu klasyfikatora, który w pierwszej kolejności będzie popełniał mniej błędów drugiego rodzaju. Na wykresie taka sytuacja jest reprezentowana przez oś Y, tj. im wyższa wartość TPR, tym mniej błędów drugiego rodzaju. **Najlepiej spisał się klasyfikator *Lasy Losowe***.

Dokładność poszczególnych klasyfikatorów została przedstawiona na wykresie:



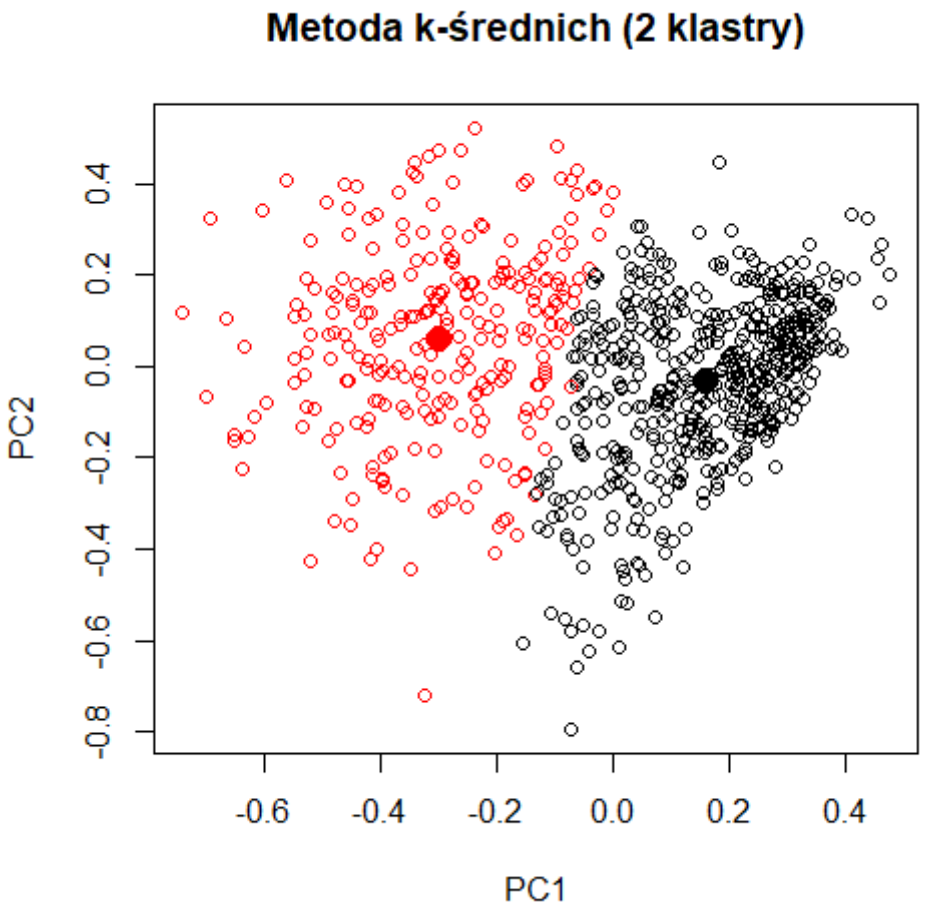
Pod względem dokładności, najlepszym klasyfikatorem dla wylosowanej próbki danych okazał się być *Naive-Bayes*. Osiągnął on około:
- 1% wyższą dokładność od klasyfikatora *Lasy Losowe*,
- 4% wyższą dokładność od klasyfikatora *Drzewa Decyzyjne*
- 6% wyższą dokładność od klasyfikatora *KNN*.

4. Grupowanie

Do grupowania użyto metody k-średnich.

```
# Obrobka (paczka editrules)
db.pca = prcomp(db.nz.norm[,1:8])
db.pca.predict = predict(db.pca)
# Grupowanie na 2 klastry
db.kmeans = kmeans(db.pca.predict, 2)
# Wykres
plot(db.pca.predict, col = db.kmeans[["cluster"]], main = "Metoda k-średnich (2 klastry)")
points(db.kmeans[["centers"]], col = 1:8, pch = 16, cex = 1.8)
```

W przypadku zastosowania 2 klastrów, na wykresie dosyć łatwo dostrzec linię podziału.



Warto zweryfikować, czy podział ten jest zgodny z klasą Outcome (Healthy/Sick) danych.
W tym celu policzono odsetek wierszy z klasą Healthy w klastrze 1 w odniesieniu do całkowitej ilości wierszy z klasą Healthy w bazie. Analogiczne obliczenia zastosowano do klastra 2.

```
# iteracja po danych
for (i in 1:nrow(db.nz.norm))
{
  # jeśli wiersz jest w pierwszym klastrze
  if (db.kmeans$cluster[i] == 1)
  {
    # i jest healthy - zwiększ liczbę healthy w pierwszym klastrze
    if (db.nz.norm$Outcome[i] == "healthy") first.cluster.healthy++
    # jeśli jest chory - zwiększ liczbę sick w pierwszym klastrze
    else first.cluster.sick++
  }
  else
  {
    if (db.nz.norm$Outcome[i] == "healthy") second.cluster.healthy++
    else second.cluster.sick = second.cluster.sick + 1
  }
}

# liczba zdrowych i chorych w całej bazie
healthy = nrow(subset(db.nz.norm,Outcome == "healthy"))
sick = nrow(subset(db.nz.norm,Outcome == "sick"))

# liczba osób w klastrach
first.cluster.total = length(which(db.kmeans$cluster == 1))
second.cluster.total = length(which(db.kmeans$cluster == 2))

# zliczenie odsetka zdrowych i chorych
first.cluster.healthy.ratio = first.cluster.healthy / healthy
second.cluster.healthy.ratio = second.cluster.healthy / healthy
first.cluster.sick.ratio = first.cluster.sick / sick
second.cluster.sick.ratio = second.cluster.sick / sick
```

Otrzymano następujące wyniki:

	Liczba osób	Odsetek zdrowych	Odsetek chorych
Klaster 1	503	0.756	0.244
Klaster 2	265	0.4664	0.5336

Łatwo dostrzec tutaj kilka ciekawych zależności:

1**. Klaster 1** zawiera dwukrotnie więcej osób niż klaster 2.

2. Wewnątrz **klastra 1** jest znaczna przewaga osób zdrowych nad osobami chorymi.

3. Wewnątrz **klastra 2** jest minimalna przewaga osób chorych nad osobami zdrowymi.

Przy takim rozkładzie możemy wyciągnąć kilka wniosków:

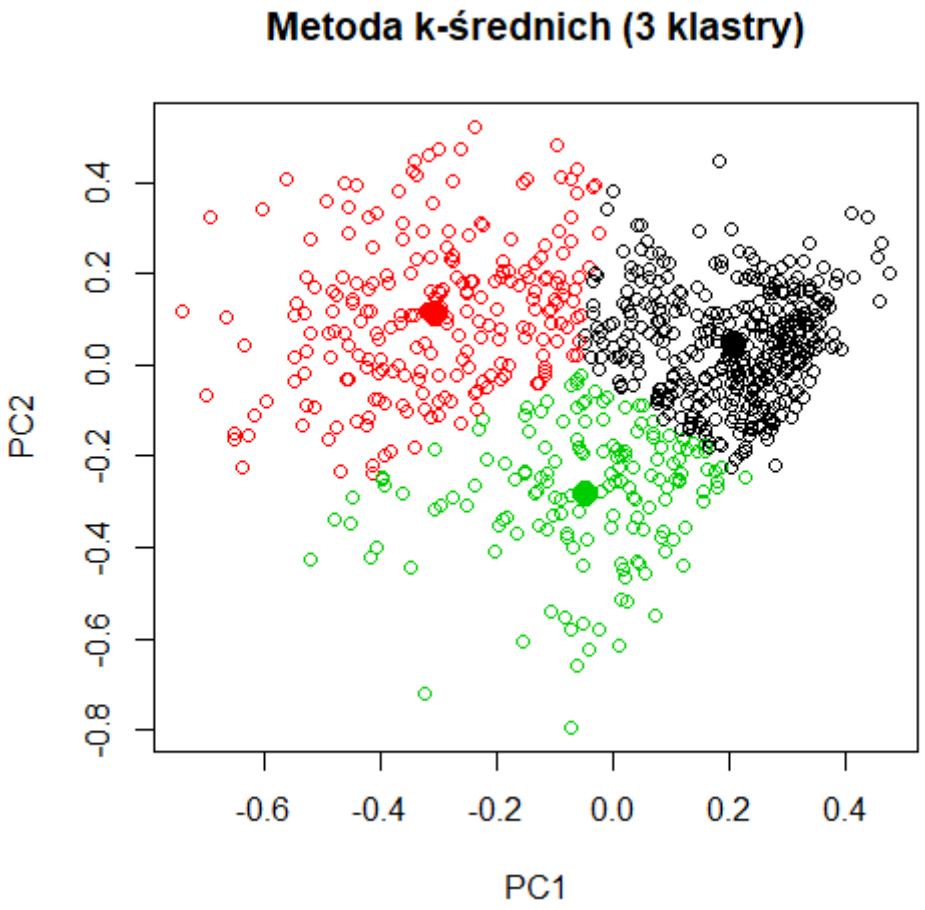
1. **Klaster 1** możemy określić jako „grupa zdrowych”.

2. **Klaster 1** może składać się z osób, które relatywnie łatwo było zidentyfikować. Mogą to być „standardowe” przypadki chorych i zdrowych. Jeśli potraktować go jako klasyfikator, to osiąga skuteczność podobną do drzew decyzyjnych.

3. **Klaster 2** możemy określić jako „grupa chorych”.

4. **Klaster 2** zawiera zapewne przypadki nietypowe, które ciężej jest sklasyfikować.

Aby zweryfikować tezy 2 i 4, dane pogrupowano ponownie, tym razem na 3 klastry i zastosowano analogiczne obliczenia .



	Liczba osób	Odsetek zdrowych	Odsetek chorych
Klaster 1	381	0.8661	0.1339
Klaster 2	230	0.4957	0.5043
Klaster 3	157	0.3567	0.6433

Analizując powyższą tabelę, zauważamy, że:

1**. Klaster 1** to prawie same osoby zdrowe, tj. dobrze reprezentuje grupę „zdrowych”.

2. **Klaster 2** zawiera praktycznie taką samą liczbę osób zdrowych i chorych.

3. **Klaster 3** składa się głównie z osób chorych, tj. reprezentuję grupę „chorzy”.

Powyższe obserwacje zdają się potwierdzać wcześniej postawione tezy – w zbiorze danych znajduje się grupa osób, które ciężko jest sklasyfikować i stanowią one klaster 2.

Lepsza „skuteczność” klastra 1 niż 3 sugeruje również, że niektóre osoby zdrowe bardzo łatwo jest sklasyfikować, tj. istnieje zapewne „typowy zdrowy pacjent”, ale już niekoniecznie „typowy chory pacjent”.

5. Reguły asocjacyjne

Zanim zastosowano reguły asocjacyjne, potrzebna była dalsza obróbka danych. Wszelkie dane numeryczne zamienione zostały na dane kategoryczne (wyliczeniowe), określające czy wartość danej komórki jest niemniejsza („High”) lub mniejsza („Low”) od średniej dla wybranej kolumny.

```
# funkcja zamieniająca wartość numeryczną na string zależnie od średniej
highOrLow = function (x, mean)
{
  if (x >= mean) { return ("High") }
  else { return ("Low") }
}

# zamieniamy wartości numeryczne na stringi
db.a = db.nz.norm
db.a$Pregnancies = sapply(db.a$Pregnancies, as.character)
db.a$Glucose = sapply(db.a$Glucose, as.character)
db.a$BloodPressure = sapply(db.a$BloodPressure, as.character)
db.a$SkinThickness = sapply(db.a$SkinThickness, as.character)
db.a$Insulin = sapply(db.a$Insulin, as.character)
db.a$BMI = sapply(db.a$BMI, as.character)
db.a$DiabetesPedigreeFunction = sapply(db.a$DiabetesPedigreeFunction, as.character)
db.a$Age = sapply(db.a$Age, as.character)
db.a$Outcome = sapply(db.a$Outcome, as.character)

# iterujemy po rekordach i podstawiamy High/Low zamiast wartości
# w zmiennych xx.mean znajdują się wyliczone średnie
for (i in 1:nrow(db.nz.norm))
{
  db.a$Pregnancies[i] = highOrLow(db.nz.norm$Pregnancies[i], pregnancies.mean)
  db.a$Glucose[i] = highOrLow(db.nz.norm$Glucose[i], glucose.mean)
  db.a$BloodPressure[i] = highOrLow(db.nz.norm$BloodPressure[i], bloodPressure.mean)
  db.a$SkinThickness[i] = highOrLow(db.nz.norm$SkinThickness[i], skinThickness.mean)
  db.a$Insulin[i] = highOrLow(db.nz.norm$Insulin[i], insulin.mean)
  db.a$BMI[i] = highOrLow(db.nz.norm$BMI[i], bmi.mean)
  db.a$DiabetesPedigreeFunction[i] = highOrLow(db.nz.norm$DiabetesPedigreeFunction[i],
dpf.mean)
  db.a$Age[i] = highOrLow(db.nz.norm$Age[i], age.mean)
}

# zamieniamy stringi na typy kategoryczne
db.a$Pregnancies = sapply(db.a$Pregnancies, as.factor)
db.a$Glucose = sapply(db.a$Glucose, as.factor)
db.a$BloodPressure = sapply(db.a$BloodPressure, as.factor)
db.a$SkinThickness = sapply(db.a$SkinThickness, as.factor)
db.a$Insulin = sapply(db.a$Insulin, as.factor)
db.a$BMI = sapply(db.a$BMI, as.factor)
db.a$DiabetesPedigreeFunction = sapply(db.a$DiabetesPedigreeFunction, as.factor)
db.a$Age = sapply(db.a$Age, as.factor)
db.a$Outcome = sapply(db.a$Outcome, as.factor)
```

Dla tak obrobionych danych generujemy reguły asocjacyjne:

```
# tworzymy zestaw reguł (paczka arules)
rules = apriori(db.a, parameter = list(minlen=2, supp=0.005, conf=0.8), appearance =
list(rhs=c("Outcome=Healthy", "Outcome=Sick"), default="lhs"), control = list(verbose=F))
# usuwamy redundantne reguły
rules.sorted = sort(rules, by="lift")
subset.matrix = is.subset(rules.sorted, rules.sorted)
subset.matrix[lower.tri(subset.matrix, diag=T)] = FALSE
redundant = colSums(subset.matrix, na.rm=T) >= 1
rules.pruned = rules.sorted[!redundant]
# ponownie sortujemy reguły
rules.pruned.sorted = sort(rules.pruned, by="lift")
inspect(rules.pruned.sorted)
```

Uzyskano w ten sposób 271 unikalnych reguł. Oto kilka z nich, posortowanych względem czynnika lift:

1. {Pregnancies=High, Glucose=High, BloodPressure=High} => {Age=High}
Zasada ta wydaje się mieć solidne uzasadnienie w świecie rzeczywistym:
- starsze kobiety miały więcej czasu na rodzenie dzieci,
- starsze osoby na ogół mają wyższy poziom glukozy,
- starsze osoby na ogół mają wyższe ciśnienie krwi.

2. {Pregnancies=High, Glucose=High, Outcome=sick} => {Age=High}
Podobnie jak zasada 1.

4. {DiabetesPedigreeFunction=Low, Age=High, Outcome=healty} => {Pregnancies=High}
Jest to dosyć ciekawa zależność, którą możemy interpretować następująco:
Jeśli osoba jest zdrowa i wśród jej rodziny również nie występuje cukrzyca, to jest uznawana za atrakcyjną kandydatkę do wydania potomstwa, stąd podczas swojego życia jest w stanie urodzić go więcej.

13. {Pregnancies=Low, SkinThickness=High, Age=Low} => {BMI=High}
Młoda osoba z niską liczbą przebytych ciąży ma „grubą skórę”, to możliwe, że jest po prostu otyła (wysokie BMI).

14. {SkinThickness=High, Outcome=sick} => {BMI=High}
Osoba chora na cukrzycę, z „grubą skórą”, jest na ogół otyła.

79. {Pregnancies=Low, Glucose=Low, SkinThickness=Low, BMI=Low} => {Outcome=healthy}
Szczupła osoba z niskim poziomem cukru i małą ilością przebytych ciąży jest na ogół zdrowa.

179. {Glucose=Low, BloodPressure=Low, BMI=High} => {Age=Low}
Ciekawa zasada: jeśli osoba jest otyła, ale jednocześnie ma niskie ciśnienie i poziom glukozy, to prawdopodobnie jest młoda i nie zdążyło się u niej rozwinąć nadciśnienie.

212. {BMI=Low, DiabetesPedigreeFunction=Low, Age=Low} => {Outcome=healthy}
Rozsądna zasada: młoda, szczupła osoba z niskim obciążeniem genetycznym rzadziej choruje.

6. Podsumowanie

Występowanie cukrzycy wśród indiańskich kobiet z plemienia Pima okazało się być całkiem ciekawym zagadnieniem. Baza danych zawierała pewne braki, które uzupełniono wartościami średnimi. Naturalnym kandydatem na klasę był atrybut Outcome, który określał, czy dana kobieta jest chora na cukrzycę. To właśnie tę informację chcemy oszacować, opierając się na pozostałych parametrach.

Do klasyfikacji zastosowano klasyfikatory takie jak Drzewa Decyzyjne, KNN, Naive-Bayes i Lasy Losowe. Uzyskane wyniki były dosyć zbliżone (ok. 75% dokładności), a najlepszą metodą okazała się być ta ostatnia, gdyż produkowała najmniej błędów drugiego rodzaju, potencjalnie doprowadzających do utraty zdrowia lub życia pacjenta, u którego niesłusznie wykluczono cukrzycę.

Ciekawych obserwacji dostarczyło grupowanie metodą k-średnich. Już przy dwóch klastrach, można było pokusić sie o kilka teorii dotyczących badanej populacji.
Podział na trzy klastry dowiódł, że z dużo większą pewnością da się stwierdzić, że pacjent jest zdrowy, niż chory. Podział taki wyodrębnił również trzecią grupę, która składała się z takiej samej ilości zdrowych i chorych osób – zapewne znalazły się tam osoby, które rychło mogą zachorować na cukrzycę, lub są w jej początkowym stadium. Metoda grupowania okazała się być zaskakująco dobra.

Reguły asocjacyjne nie dostarczyły wielu nowych informacji w kontekście oceny zdrowia pacjenta, jednak znalazło się wśród nich kilka pozycji, które bardzo dobrze oddawały rzeczywistość, takie jak fakt, że starsza osoba o dobrym zdrowiu mogła urodzić ponadprzeciętną liczbę potomstwa.

Stworzone klasyfikatory z pewnością mogłyby przysłużyć się np. w badaniach posiewowych, poprzez wyodrębnienie osób, które z największym prawdopodobieństwem mogą być chore, bądź zachorować w niedalekiej przyszłości.

Źródła

1. Baza danych
https://www.kaggle.com/uciml/pima-indians-diabetes-database

Załączniki

1. Pliki źródłowe w języku R
2. Baza danych CSV
3. Treść zadania (PDF)