Inteligencja obliczeniowa — Zgłębianie danych

Jakub Ronkiewicz, 238155

12 grudnia 2018

1 Baza danych

Wybrałem bazę danych zawierająca informację na temat zachorowań na choroby serca (**Heart Disease**, `https://archive.ics.uci.edu/ml/datasets/Heart+Disease`). Główna baza danych zawierała, aż 76 kolumn. Jednak specjaliści od machine learningu pracowali wyłącznie na bazie Clevaland, która zawiera tylko 14 z 76 początkowych kolumn. Poniżej przedstawiam, jakie 14 kolumn znajduję się w bazie Cleveland i co one przechowują:

Nazwa kolumny	Opis
age	wiek jako int
sex	płeć (1=mężczyzna, 0=kobieta)
cp	Rodzaj bólu klatki piersiowej: • 1: typowa dusznica • 2: atypowa dusznica • 3: ból nie-dławicowy • 4: bezobjawowy
trestbps	spoczynkowe ciśnienie krwi `[mmHg]`
chol	cholesterol całkowity w `[mg/dl]`
fbs	poziom glukozy we krwi na czczo (1 = fbs >120 mg/dl)
restecg	spoczynkowe EKG • 0: normalne • 1: z nieprawidłością fali ST-T • 2: przerost lewej komory według kryteriów Estesa
thalach	maksymalne tętno
exang	dusznica wywołana wysiłkiem (1=tak, 0=nie)
oldpeak	poziom ST wywołany wysiłkiem w porównaniu do odpoczynku
slope	nachylenie szczytowego odcinka ST (1: wzrost, 2: płaski, 3: spadek)
ca	liczba głównych naczyń (0-3) zabarwionych za pomocą fluoroskopii
thal	3 = normalny, 6 = ustalony defekt, 7 = wada odwracalna
num	diagnoza równa 0 oznacza, że jest zdrowy, (1,2,3,4) określa stopień zachorowania

Oczywiście kolumna `num` najbardziej nadaję się na wybranie jako klasa. Zawiera wartości liczbowe od 0 (choroba nie występuje) do 4. Celem projektu będzie znalezienie jak najdokładniejszego sposobu na przewidywania zachorowań pacjenta na chorobę serca na podstawie 13 wcześniej podanych kolumn.

2 Przetwarzanie bazy danych

Pierwszym zadaniem była obróbka kolumny którą wybraliśmy na klasę `num`. Ma ona zbyt wiele wartości, chcemy dostać informacje, czy pacjent jest zdrowy czy chorzy, a nie w jakim stopniu jest chory. Prosty warunek `ifelse` pozwala wcielić to w życie.

```
data$num = ifelse(data$num == 0, "No", "Yes")
```

W bazie znajdują się trochę pustych wartości w kolumnach, oznaczone są "?". Wartości te zaburzały by przewidywanie wyników, także musimy się ich pozbyć.Szukam ich i oznaczam jako bez wartości, następnie usuwam te wiersze z bazy.

```
data[data == "?"] <- NA
data <- na.omit(data)
```

Klasyfikator knn wymaga by wszystkie kolumny były numeryczne, a kolumna określająca klasę była `factorem`.

```
> str(data)
'data.frame':   297 obs. of  14 variables:
 $ age     : num  63 67 67 37 41 56 62 57 63 53 ...
 $ sex     : num  1 1 1 1 0 1 0 0 1 1 ...
 $ cp      : num  1 4 4 3 2 2 4 4 4 4 ...
 $ trestbps: num  145 160 120 130 130 120 140 120 130 140 ...
 $ chol    : num  233 286 229 250 204 236 268 354 254 203 ...
 $ fbs     : num  1 0 0 0 0 0 0 0 0 1 ...
 $ restecg : num  2 2 2 0 2 0 2 0 2 2 ...
 $ thalach : num  150 108 129 187 172 178 160 163 147 155 ...
 $ exang   : num  0 1 1 0 0 0 0 1 0 1 ...
 $ oldpeak : num  2.3 1.5 2.6 3.5 1.4 0.8 3.6 0.6 1.4 3.1 ...
 $ slope   : num  3 2 2 3 1 1 3 1 2 3 ...
 $ ca      : Factor w/ 5 levels "?","0.0","1.0",..: 2 5 4 2 2 2 4 2 3 2 ...
 $ thal    : Factor w/ 4 levels "?","3.0","6.0",..: 3 2 4 2 2 2 2 2 4 4 ...
 $ num     : chr  "No" "Yes" "Yes" "No" ...
```

Widać, że kolumna ca, thal są factorem, a powinny być numeryczne. Natomiast num jest typu char, a powinna być factorem. Poniższy kod pokazuje jak to naprawić:

```
data$ca <- as.character(data$ca)
data$ca <- as.numeric(data$ca)
data$thal <- as.numeric(data$thal)
data$num = as.factor(data$num)
```

3 Klasyfikatory i ich ewaluacja

Dzielenie na zbiór treningowy i testowy

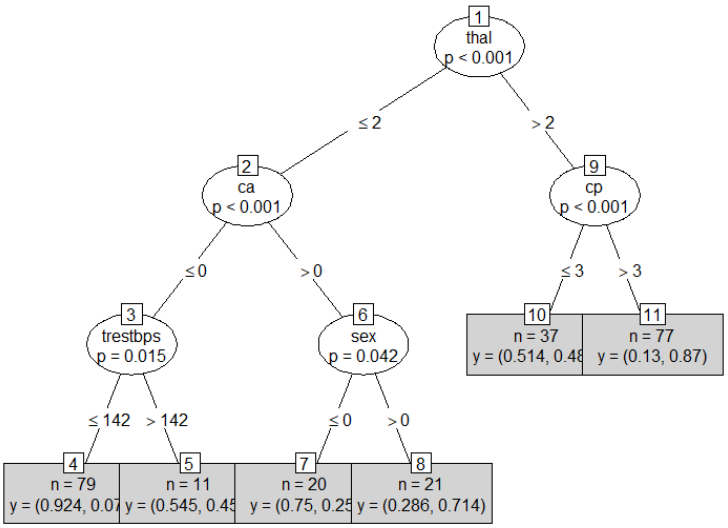
Poniższy fragment kodu, dzieli bazę danych na zbiór treningowy i testowy w stosunku 80% do 20%

```
set.seed(1234)
ind <- sample(2, nrow(data), replace=TRUE, prob=c(0.8,0.2))
data.train <- data[ind==1, 1:14]
data.test <- data[ind==2, 1:14]
real <- data.test[,14]
```

C4.5/ID3 Drzewo

Kod odpowiedzialny za utworzenie struktury drzewa

```
ctree <- ctree(num ~ ., data = data.train)

ctree.predicted <- predict(ctree, data.test[,1:13])
ctree.confMat <- table(ctree.predicted, real)[2:1, 2:1]
ctree.accuracy <- mean(real == ctree.predicted)
ctree.tpr <- get_tpr(ctree.confMat)
ctree.fpr <- get_fpr(ctree.confMat)
```



Rysunek 1: Stuktura drzewa

Wyniki

Predicted \ Real	Yes	No
Yes	16	2
No	5	29

Tabela 1: Macierz błędu dla Drzewa

Dokładność wyniosła 86.53 %

Naive Bayes

Kod odpowiedzialny za uruchomienie klasyfikatora NaiveBayes:

```
nb <- naiveBayes(num ~ ., data=data.train)
nb.predicted <- predict(nb, newdata=data.test)
nb.confMat <- table(nb.predicted, real)[2:1, 2:1]
nb.accuracy <- mean(real == nb.predicted)
nb.tpr <- get_tpr(nb.confMat)
nb.fpr <- get_fpr(nb.confMat)
```

Wyniki

Predicted \ Real	Yes	No
Yes	18	6
No	3	25

Tabela 2: Macierz błędu dla NaiveBayes

Dokładność wyniosła 82.69 %

kNN

Klasyfikator kNN wymagał przygotowania danych do jego uruchomienia. Trzeba było znormalizować bazę. Funkcja do normalizacji:

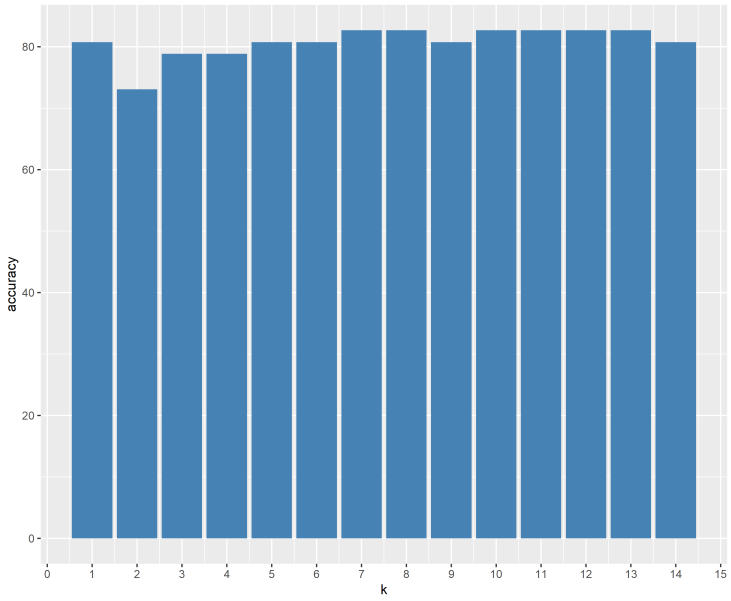
```
normalize <- normalize <- function(x) {
  return ((x - min(x)) / (max(x) - min(x)))
}
```

Normalizacja danych:

```
data.norm <- as.data.frame(lapply(data[1:13], normalize))
data.norm$num = data$num
data.norm.train <- data.norm[ind==1,]
data.norm.test <- data.norm[ind==2,]
```

Dla klasyfikatora `kNN` możemy dobrać różne k. Postanowiłem sprawdzić dla jakiego k od 1 do 14, uzyskam najwyższa dokładność.

```
knn.list <- data.frame(k = numeric(), accuracy = numeric())
for (i in 1:14) {
  temp_knn <- knn(data.norm.train[1:13], data.norm.test[1:13], cl=data.norm.train$nu
  knn.list[nrow(knn.list)+1,] <- c(i, mean(real==temp_knn))
}
# Szukanie k z najwieksza dokladnoscia
knn.kWithMaxAccuracy = knn.list[which.max(knn.list$accuracy),1]
```



Rysunek 2: Wykres zależności dokładności od parametru k

Wyniki

Predicted \ Real	Yes	No
Yes	19	7
No	2	24

Tabela 3: Macierz błędu dla kNN

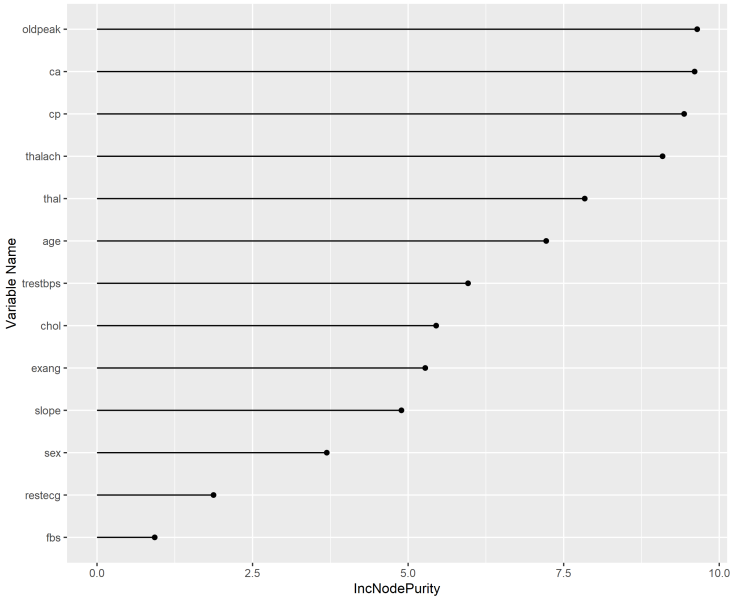
Dokładność wyniosła 82.69 %

Random Forest

Kod odpowiedzialny za uruchomienie klasyfikatora Random Forest:

```
rf <- randomForest(num ~ ., data=data.train, mtry=1, importance = TRUE)
rf.predicted <- predict(rf, data.test)
rf.confMat <- table(rf.predicted, real)[2:1, 2:1]
rf.accuracy <- mean(real == rf.predicted)
rf.tpr <- get_tpr(rf.confMat)
rf.fpr <- get_fpr(rf.confMat)
var_importance <- varImpPlot(rf)
```

Klasyfikator pozwala nam narysować wykres, przedstawiająca jak ważna jest każda z kolumn. Im większy wynik, tym gorsze byłby wyniki przewidywania bez danej kolumny.



Rysunek 3: Wykres pokazujący ważność wszystkich kolumn

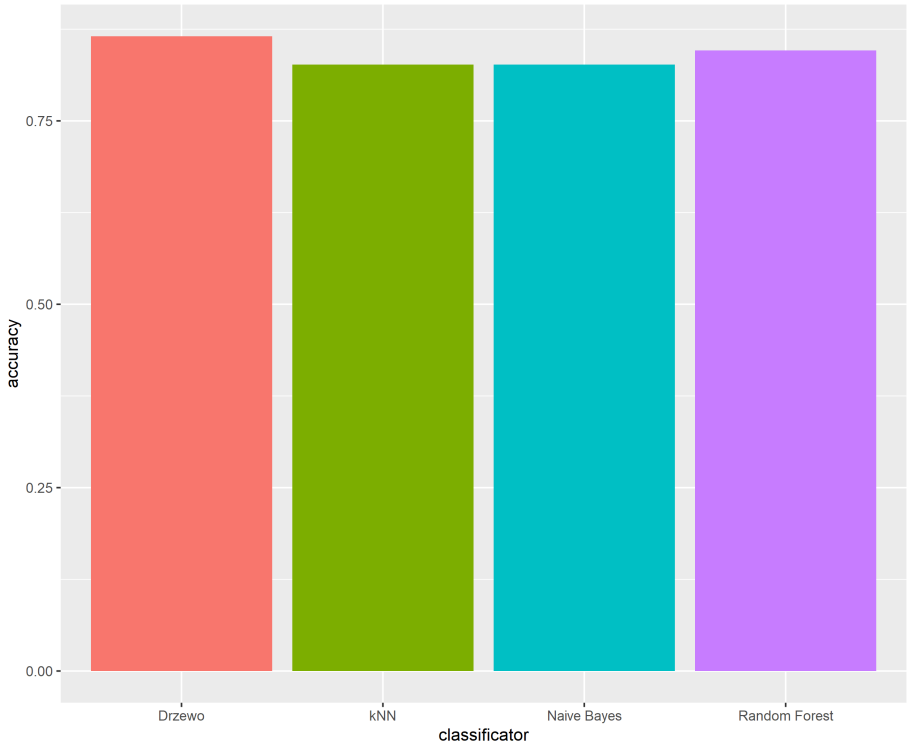
Wyniki

Predicted \ Real	Yes	No
Yes	17	4
No	4	27

Tabela 4: Macierz błędu dla Random Forest

Dokładność wyniosła 84.61 %

Porównanie dokładności wszystkich klasyfikatorów



Rysunek 4: Wykres pokazujący dokładność klasyfikatorów

Klasyfikator `C4.5/ID3 Drzewo` dla naszej bazy danych daje najlepsze rezultaty. `kNN` i `NaiveBayes` daję te same, aczkolwiek najsłabsze rezultaty. `RandomForest` zajmuje drugie miejsce, jeśli chodzi o dokładność.

Jak dokładniej ewaluować klasyfikatory

Co oznaczają u mnie wartości TP, FP, TN, FN?

TP czyli człowiek chory, który został zdiagnozowany jako chory. **FP** czyli człowiek zdrowy, który został zdiagnozowany jako chory. **TN** czyli człowiek zdrowy, który został zdiagnozowany jako zdrowy. **FN** czyli człowiek chory, który został zdiagnozowany jako zdrowy.

Dla każdego klasyfikatora obliczyć TPR (recall, sensitivity) i FPR (fall-out, false alarm).

Napisałem funkcję, które obliczają mi te dwa składniki, później po stworzeniu matrycy błędów, obliczam je dla każdego klasyfikatora

```
get_tpr <- function(t){
  # t - confusion matrix, TP/TP+FN
  tpr = t[1]/(t[1]+t[2])
  return(tpr)
}
get_fpr <- function(t) {
  tpr = t[3]/(t[3]+t[4])
  return(tpr)
}
```

Klasyfikator	TPR	FPR
Drzewo	0.7619048	0.0645161
NaiveBayes	0.8571429	0.1935484
kNN	0.9047619	0.2258065
RandomForest	0.8095238	0.1290323

Tabela 5: Wskaźniki TPR i FPR dla każdego z klasyfikatorów

Rozszyfrować w jakiej zależności od TPR i FPR są miary FNR i TNR. Podać te wzory w sprawozdaniu.

$$FNR = 1 - TPR$$

$$TNR = 1 - FPR$$

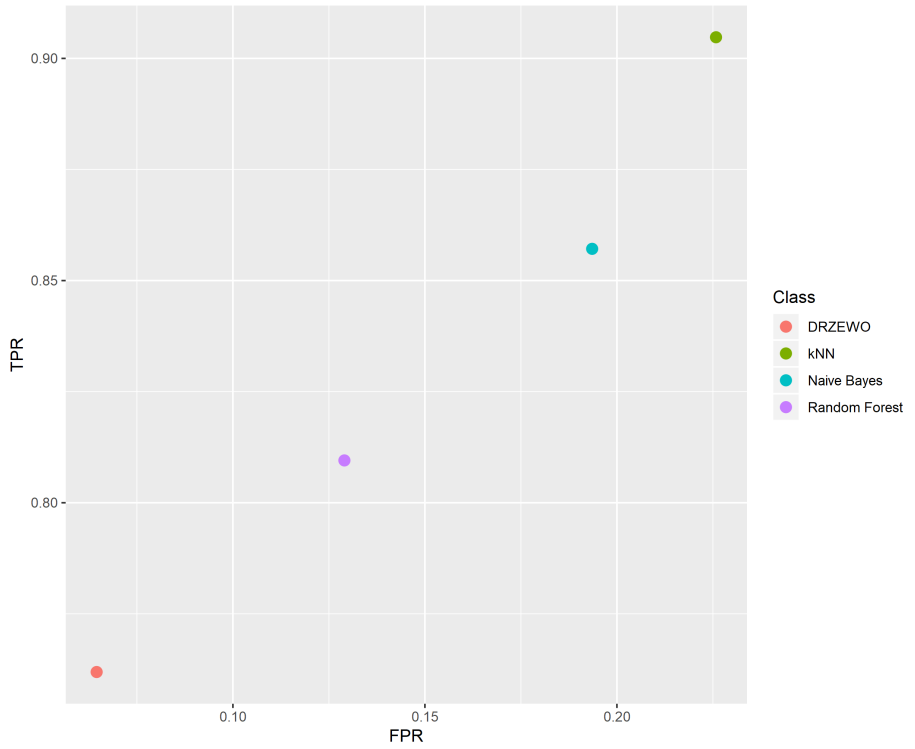
Udzielić odpowiedzi czym jest u Państwa błąd pierwszego i drugiego rodzaju. Jak mają się oba rodzaje błędów do TPR, FPR, TNR, FNR. Im więcej błędów pierwszego rodzaju tym większe jest co? Co z błędami drugiego rodzaju?

Błąd pierwszego rodzaju to ludzie zdrowi, błędnie zdiagnozowania jako chorzy. Błąd drugiego rzędu to ludzie chorzy, błędnie zdiagnozowania jako zdrowi. Im więcej błędów pierwszego rodzaju tym mniejszy TNR, a tym większy FPR. Im więcej błędów drugiego rodzaju tym mniejszy TPR, a tym większy FNR.

Pytanie filozoficzne, na które również proszę udzielić odpowiedzi: który z błędów w Państwa bazie jest gorszy do popełnienia: pierwszego czy drugiego rodzaju? Odpowiedź uzasadnić.

Uważam, że błąd drugiego rodzaju jest gorszy do popełnienia. Na przykładzie wykorzystanej przeze mnie bazy: ludzie, którzy na prawdę są chorzy zostaną zakwalifikowani jako zdrowi i nie będą mieli szans na leczenie, co może spowodować zgon. Przy błędzie pierwszego rodzaju, zdrowy pacjent po dokładnych badaniach zostanie odesłany do domu.

Dla każdego z czterech klasyfikatorów obliczyć parę (FPR,TPR) i zaznaczyć jako punkt na wykresie



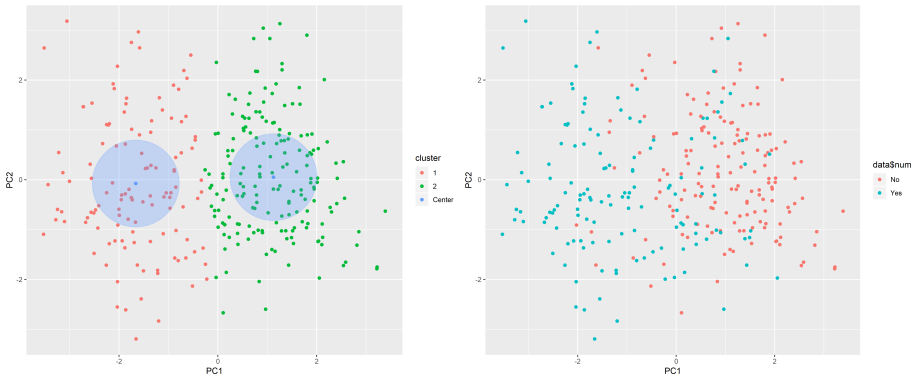
Rysunek 5: ROC Space

Najlepszy klasyfikator znajdowałby się w górnym lewym rogu. Drzewo wydaję się być dobrym wyborem, aczkolwiek, mała wartość TPR może oznaczać, że jest trochę błędów drugiego rzędu, który chcieli byśmy unikać. Najmniej błędów gorszego rzędu ma kNN.

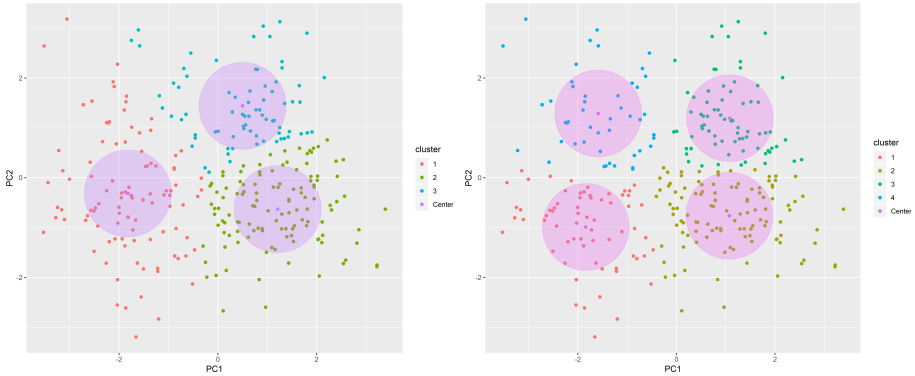
4 Grupowanie metodą k-śrendich

Cały kod odpowiedzialny za przygotowanie danych od uruchomienia algorytmu, utworzenie wykresów etc.

```
data.log <- log(data[,c(1:13)])

replace_inf <- function(x) {
  x[is.infinite(x)] <- NA
  replace_value <- mean(x, na.rm = TRUE)
  x[is.na(x)] <- replace_value
}

replace_means <- sapply(data.log, replace_inf)
replace_means <- as.data.frame(replace_means)

data.log$oldpeak[is.infinite(data.log$oldpeak)] <- replace_means["oldpeak",]
data.log$restecg[is.infinite(data.log$restecg)] <- replace_means["restecg",]
data.log$ca[is.infinite(data.log$ca)] <- replace_means["ca",]

data.log$sex = data$sex
data.log$fbs = data$fbs
data.log$exang = data$exang

data.stand <- scale(data.log, center=TRUE)
data.pca <- prcomp(data.stand)
data.final <- predict(data.pca)[,1:2]
data.final <- as.data.frame(data.final)

kmeans <- kmeans(data.final, 2)
data.final$cluster = factor(kmeans$cluster)
centers = as.data.frame(kmeans$centers)


kmeans.pred.plot <- ggplot(data.final, aes(x=PC1, y=PC2, color=cluster)) +
  geom_point() +
  geom_point(data=centers, aes(x=PC1,y=PC2, color='Center')) +
  geom_point(data=centers, aes(x=PC1,y=PC2, color='Center'), size=52, alpha=.3, sho

kmeans.real.plot <- ggplot(data.final, aes(x=PC1, y=PC2, color=data$num)) +
  geom_point()
```



Rysunek 6: Podział na dwa klastry, pórównanie prawdziwych wyników z algorytmem



Rysunek 7: Podział na trzy klastry i cztery klastry

Uważam, że przy podziale na dwa klastry algorytm prawidłowo oddzielił jedną klasę od drugiej. Oczywiście, nie wszystkie rekordy się zgadzają, ale poprawność jest na wysokim poziomie. Zdołał podzielić na trzy i cztery klastry, prawdopodobnie odpowiada to stopniom zachorowania na choroby serca w początkowej bazie danych.

5 Reguły asocjacyjne

Fragment kodu odpowiedzialny za przekształcenie danych umożliwiające uruchomienie algorytmu apriori, zawiera również ustawienia dla algorytmu apriori.

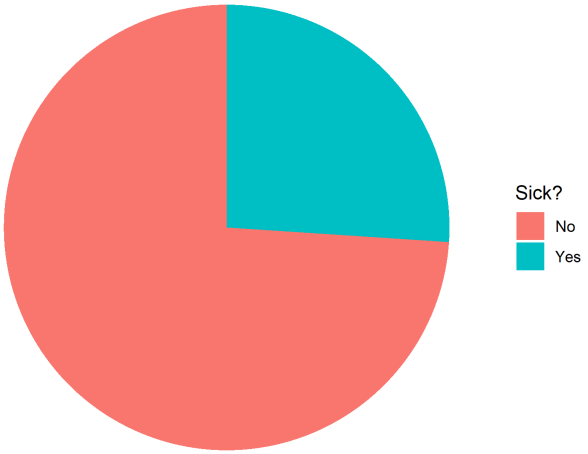
```
data.rules <- data.table(data)
data.rules$age = as.factor(data.rules$age)
data.rules$sex = ifelse(data.rules$sex == 1, "Male", "Female")
data.rules$sex = as.factor(data.rules$sex)
data.rules$cp = as.factor(data.rules$cp)
data.rules$trestbps = as.factor(data.rules$trestbps)
data.rules$chol = as.factor(data.rules$chol)
data.rules$fbs = data.rules$fbs == 1
data.rules$restecg = as.factor(data.rules$restecg)
```

```
data.rules$thalach = as.factor(data.rules$thalach)
data.rules$exang = data.rules$exang == 1
data.rules$oldpeak = as.factor(data.rules$oldpeak)
data.rules$slope = as.factor(data.rules$slope)
data.rules$ca = as.factor(data.rules$ca)
data.rules$thal = as.factor(data.rules$thal)

data.rules.disc <- discretizeDF(data.rules)

data.rules.disc <- as(data.rules.disc, 'transactions')


rules <- apriori(data.rules.disc,
                 parameter = list(minlen=2, supp=0.1, conf=0.8),
                 appearance = list(rhs=c("num=No", "num=Yes"), default="lhs"),
                 control = list(verbose=F)
                 )
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]
```

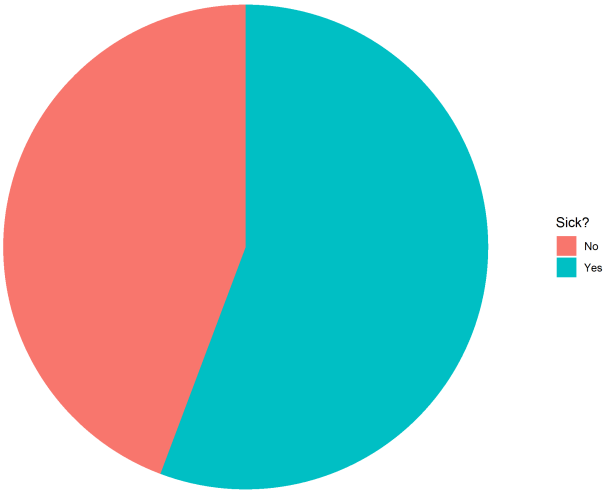
```
     lhs                                    rhs       support
confidence lift     count
[1]  {cp=4,restecg=2,thal=4}           => {num=Yes} 0.1313131 0.9750000
2.113686  39
[22] {sex=Female,cp=3,thal=2}          => {num=No}  0.1043771 1.0000000
1.856250  31
[23] {exang,slope=2}                   => {num=Yes} 0.1784512 0.8548387
1.853190  53
```

[1] Dużą szansę na zachorowalność mają pacjenci z następujacymi cechami: bezobjawowym bólem klatki piersiowej, przerostem lewej komory, zaburzeniem syntezy hemoglobiny. [22]Każda pacjentka, która nie ma bólu dławicowego klatki piersiowej i nie ma zaburzen syntezy hemoglobiny będzie zdrowa. [23]Jeśli ktoś ma dusznicę wywołaną wysiłkiem i płaski szczytowy odcinek ST, jest duża szansa, że będzie chory.

6 Zachorowalność w zależności od płci



Rysunek 8: Zachorowalność kobiet



Rysunek 9: Zachorowalność mężczyzn

7 Podsumowanie

Z klasyfikatorów największą dokładność wykazało się drzewo. Jednakże, należy jasno określić, że dokładność na poziomie 86% nie jest do końca zadowalająca. Przy pracy z danymi medycznymi na pewno ciekawym rozwiązaniem są reguły asocjacyjne. Dzięki nim, możemy się dowiedzieć jakie objawy charakteryzują daną chorobę.