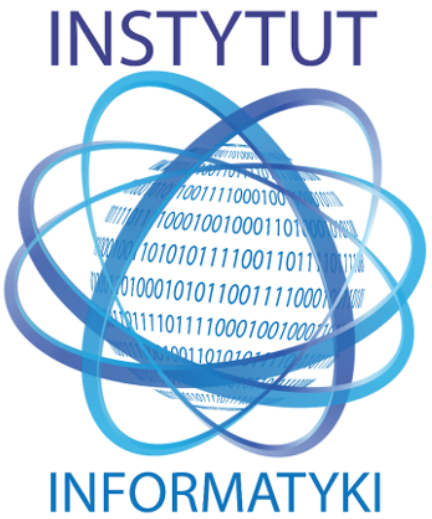
Uniwersytet w Białymstoku
Instytut informatyki



Projekt Sztuczna inteligencja
2021/2022

Wykonał: Dawid Metelski

Podstawowe informacje

Sprzęt:

- Procesor: Intel Core i5-9400F 2.9GHz
- Pamięć RAM: 16GB 3000MHz
- Karta graficzna: MSI GeForce GTX 1660
- System operacyjny: Windows 10 64bit

Wykorzystane środowisko:

- RStudio

Wykorzystane biblioteki:

- randomForest
- neuralnet
- kernlab
- naivebayes
- cluster
- ClusterR

Wykorzystane zbiory danych

1. Haberman's Survival Data Set

Źródło: https://archive.ics.uci.edu/ml/datasets.php
Ilość obiektów: 306
Ilość atrybutów: 4
Brakujące wartości: 0
Lista atrybutów:

- Age of patient at time of operation : numerical
- Patient's year of operation : numerical
- Number of positive axillary nodes detected : numerical
- Survival status - atrybut decyzyjny (2 klasy) : numerical (1 or 2)

Wczytanie zbioru i przypisanie nazwy:

```
haberman <- read.table("https://archive.ics.uci.edu/ml/machine-learning-databases/haberman/haberman.data",header = F, sep = ',')
View(haberman)
# https://archive.ics.uci.edu/ml/datasets/Haberman%27s+Survival
```

2. South German Credit Data Set

Źródło: https://archive.ics.uci.edu/ml/dataset
Ilość obiektów: 1000
Ilość atrybutów: 21
Brakujące wartości: 0
Lista atrybutów:

- laufkont - atrybut decyzyjny (4 klasy) : numerical
- laufzeit : numerical
- moral : numerical
- verw : numerical
- hoehe : numerical
- sparkont : numerical
- beszeit : numerical
- rate : numerical
- famges : numerical
- buerge : numerical
- wohnzeit : numerical
- verm : numerical
- alter : numerical
- weitkred : numerical
- wohn : numerical
- bishkred : numerical
- beruf : numerical
- pers : numerical
- telef : numerical
- gastarb : numerical
- kredit : numerical

Wczytanie zbioru i przypisanie nazwy:

```
southGerman <- read.table("SouthGermanCredit.asc", header = TRUE)
# https://archive.ics.uci.edu/ml/datasets/South+German+Credit
```

Metody klasyfikacji

Wybrane metody:

- Random Forest
- K najbliższych sąsiadów
- Naive Bayes’a
- Neuralnet

Metoda Random Forest

Las losowy, losowy las decyzyjny – metoda uczenia maszynowego dla klasyfikacji lub regresji, która polega na konstruowaniu wielu drzew decyzyjnych w czasie uczenia i generowaniu klasy, która jest dominantą klas (klasyfikacja) lub przewidywaną średnią (regresja) poszczególnych drzew.
Do zaimplementowania metody wykorzystano bibliotekę randomForest, ilość drzew (ntree) wynosiła 2000, a parametr do.trace został ustawiony na 100. Podział zbiorów: 80% zbiór treningowy, 20% zbiór testujący.

Zbiór Haberman's Survival Data Set

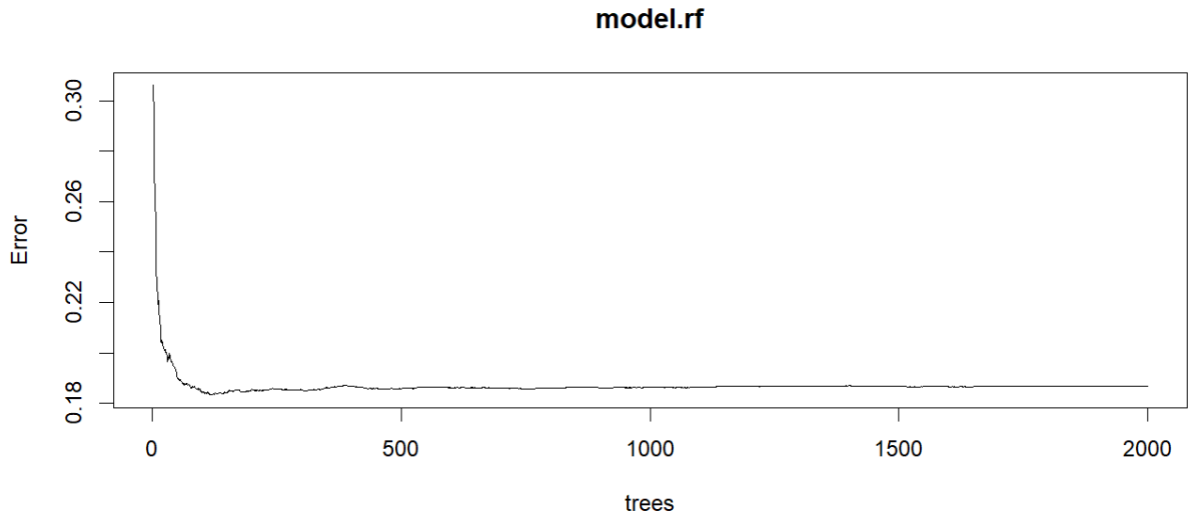
Użyty kod:

```
# zbior 1

haberman <- read.table("https://archive.ics.uci.edu/ml/machine-learning-databases/
                        haberman/haberman.data",header = F, sep = ',')

folds <- cut(seq(1, nrow(haberman)), breaks = 10, labels = FALSE)
rows <- which(folds == 10, arr.ind=TRUE)
haberman.train <- haberman[-rows,]
haberman.test <- haberman[rows,]
model.rf <- randomForest(x = haberman.train[,-4], y = haberman.train[,4], ntree = 2000, do.trace = 100)

rf.result <- predict(model.rf, newdata = haberman.test[,-4])
plot(model.rf)
```

Otrzymane wyniki:

	Out-of-bag	
Tree	MSE	%Var(y)
100	0.1846	94.89
200	0.1856	95.39
300	0.1854	95.29
400	0.1868	96.00
500	0.1858	95.49
600	0.1861	95.68
700	0.1859	95.55
800	0.186	95.63
900	0.1862	95.69
1000	0.1864	95.82
1100	0.1865	95.86
1200	0.1866	95.92
1300	0.1869	96.06
1400	0.187	96.11
1500	0.1867	95.99
1600	0.1866	95.90
1700	0.1868	96.01
1800	0.1867	96.00
1900	0.1867	95.95
2000	0.1868	96.01

Tabela przedstawia wyniki MSE (czyli błąd średniokwadratowy) i % dokładności, dla poszczególnej ilości drzew. Średnia wartość błędu średniokwadratowego wyniosła 0,186305, a średnia dokładność 95,762%.



Wykres przedstawiający zestawienie wyników błędu (oś y) dla określonej ilości drzew (oś x).

Zbiór South German Credit Data Set

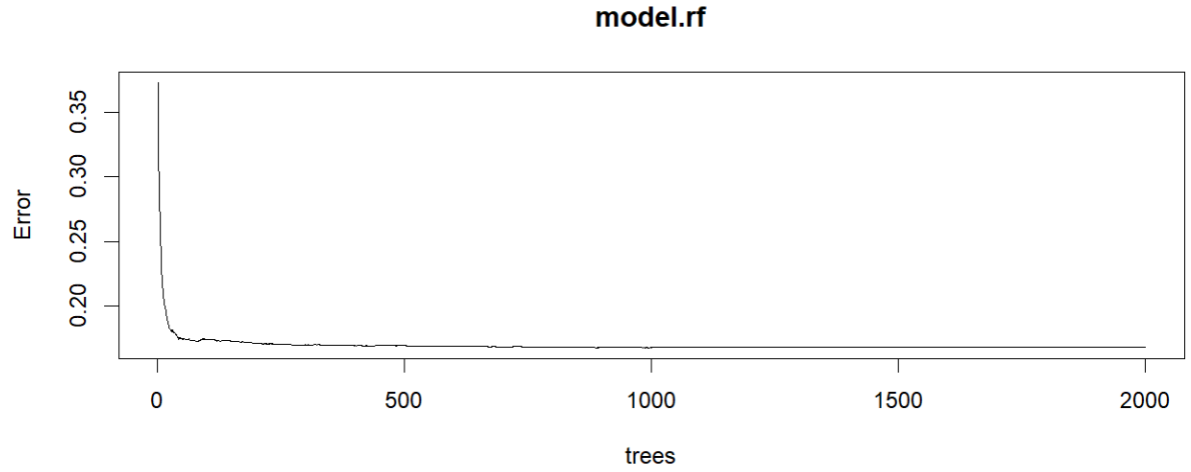
Użyty kod:

```r
# zbior 2

southGerman <- read.table("SouthGermanCredit.asc", header = TRUE)

folds <- cut(seq(1, nrow(southGerman)), breaks = 10, labels = FALSE)
rows <- which(folds == 10, arr.ind = TRUE)
southGerman.train <- southGerman[-rows,]
southGerman.test <- southGerman[rows,]
model.rf <- randomForest(x = southGerman.train[,-21], y = southGerman.train[,21], ntree = 2000, do.trace = 100)

rf.result <- predict(model.rf, newdata = southGerman.test[,-21])
plot(model.rf)
```

Otrzymane wyniki:

	Out-of-bag	
Tree	MSE	%Var(y)
100	0.1741	78.89
200	0.171	77.46
300	0.1699	76.97
400	0.1692	76.64
500	0.1693	76.69
600	0.1688	76.47
700	0.1683	76.26
800	0.1682	76.21
900	0.1678	76.02
1000	0.1677	75.96
1100	0.168	76.10
1200	0.1682	76.20
1300	0.1681	76.15
1400	0.1681	76.18
1500	0.1681	76.15
1600	0.1682	76.19
1700	0.1682	76.20
1800	0.1681	76.15
1900	0.1679	76.06
2000	0.1677	75.97

Tabela przedstawia wyniki MSE (czyli błąd średniokwadratowy) i % dokładności, dla poszczególnej ilości drzew. Średnia wartość błędu średniokwadratowego wyniosła 0,168745, a średnia dokładność 76,446%.



Wykres przedstawiający zestawienie wyników błędu (oś y) dla określonej ilości drzew (oś x).

Wnioski:

Metoda las losowy uzyskała dużo lepsze wyniki dla zbioru Haberman’s Survival Data Set. Różnica średnich dokładności między zbiorami wyniosła 19,316%. Może być to spowodowane mniejszą ilością obserwacji lub atrybutów przeważającego zbioru.

Metoda k najbliższych sąsiadów

Metoda K-najbliższych sąsiadów stosuje pamięciowy (memory-based) model zdefiniowany przez zestaw "przykładowych" przypadków o znanych wartościach zmiennej wyjściowej (i zmiennych wejściowych). Zarówno zmienne zależne jak i niezależne mogą być ciągłe i skategoryzowane.
Do implementacji metody wykorzystano bibliotekę class. Metoda dla każdego zbioru była wywoływana dla 10 różnych wartości k (ilości sąsiadów). Podział zbiorów: 80% zbiór treningowy, 20% zbiór testujący.

Zbiór Haberman's Survival Data Set

Użyty kod:

```
# zbior 1

haberman <- read.table("https://archive.ics.uci.edu/ml/machine-learning-databases/
haberman/haberman.data",header = F, sep = ',')

rows.haberman <- sample.int(nrow(haberman), size = round(nrow(haberman) / 2), replace = F)
haberman.train <- haberman[-rows.haberman, -4]
haberman.test <- haberman[rows.haberman, -4]
cl.train.haberman <- haberman[-rows.haberman, 4]
cl.test.haberman <- haberman[rows.haberman, 4]
for(i in 1:10){
  knn.result.haberman <- knn(train = haberman.train, test = haberman.test, cl =
                             cl.train.haberman, k = i) # tu ilosc sasiadow
  error.haberman <- sum(cl.test.haberman != knn.result.haberman) / length(knn.result.haberman)
  print(paste("Dla k =", i, " Blad:", round(error.haberman, 2), "% Dokladnosc: ",
              1 - round(error.haberman, 2), "%"))
}
```

Otrzymane wyniki:

```
[1] "Dla k = 1  Blad: 0.27 % Dokladnosc:  0.73 %"
[1] "Dla k = 2  Blad: 0.35 % Dokladnosc:  0.65 %"
[1] "Dla k = 3  Blad: 0.3 % Dokladnosc:  0.7 %"
[1] "Dla k = 4  Blad: 0.33 % Dokladnosc:  0.67 %"
[1] "Dla k = 5  Blad: 0.29 % Dokladnosc:  0.71 %"
[1] "Dla k = 6  Blad: 0.29 % Dokladnosc:  0.71 %"
[1] "Dla k = 7  Blad: 0.29 % Dokladnosc:  0.71 %"
[1] "Dla k = 8  Blad: 0.29 % Dokladnosc:  0.71 %"
[1] "Dla k = 9  Blad: 0.29 % Dokladnosc:  0.71 %"
[1] "Dla k = 10  Blad: 0.3 % Dokladnosc:  0.7 %"
```

Wynik przedstawia wartość błędu i dokładności, dla każdego k. Średnia dokładność wyniosła 70%, a błąd 30%.

Zbiór South German Credit Data Set

Użyty kod:

```
# zbior 2
southGerman <- read.table("SouthGermanCredit.asc", header = TRUE)

rows.southGerman <- sample.int(nrow(southGerman), size = round(nrow(southGerman) / 2), replace = F)
train.set <- southGerman[-rows.southGerman, -21]
test.set <- southGerman[rows.southGerman, -21]
cl.train.southGerman <- southGerman[-rows.southGerman, 21]
cl.test.southGerman <- southGerman[rows.southGerman, 21]
for(i in 1:15){
  knn.result.southGerman <- knn(train = train.set, test = test.set, cl =
                                  cl.train.southGerman, k = i) # tu ilosc sasiadow
  error.southGerman <- sum(cl.test.southGerman != knn.result.southGerman) /
    length(knn.result.southGerman)
  print(paste("Dla k =", i, " Blad:", round(error.southGerman, 2), "% Dokladnosc:",
              1 - round(error.southGerman, 2), "%"))
}
```

Otrzymane wyniki:

```
[1] "Dla k = 1  Blad: 0.42 % Dokladnosc: 0.58 %"
[1] "Dla k = 2  Blad: 0.43 % Dokladnosc: 0.57 %"
[1] "Dla k = 3  Blad: 0.41 % Dokladnosc: 0.59 %"
[1] "Dla k = 4  Blad: 0.38 % Dokladnosc: 0.62 %"
[1] "Dla k = 5  Blad: 0.38 % Dokladnosc: 0.62 %"
[1] "Dla k = 6  Blad: 0.38 % Dokladnosc: 0.62 %"
[1] "Dla k = 7  Blad: 0.36 % Dokladnosc: 0.64 %"
[1] "Dla k = 8  Blad: 0.35 % Dokladnosc: 0.65 %"
[1] "Dla k = 9  Blad: 0.34 % Dokladnosc: 0.66 %"
[1] "Dla k = 10  Blad: 0.36 % Dokladnosc: 0.64 %"
[1] "Dla k = 11  Blad: 0.36 % Dokladnosc: 0.64 %"
[1] "Dla k = 12  Blad: 0.36 % Dokladnosc: 0.64 %"
[1] "Dla k = 13  Blad: 0.34 % Dokladnosc: 0.66 %"
[1] "Dla k = 14  Blad: 0.33 % Dokladnosc: 0.67 %"
[1] "Dla k = 15  Blad: 0.32 % Dokladnosc: 0.68 %"
```

Wynik przedstawia wartość błędu i dokładności, dla każdego k. Średnia dokładność wyniosła 63.2%, a błąd 36.8%.

Wnioski:

Metoda uzyskała dość słabe, zbliżone do siebie wyniki. Oznaczać to może, że różnica obserwacji i atrybutów zbiorów nie spowodowała dużej różnicy wyników.

Metoda Naive Bayes’a

Naiwny klasyfikator bayesowski, naiwny klasyfikator Bayesa – prosty klasyfikator probabilistyczny. Naiwne klasyfikatory bayesowskie są oparte na założeniu o wzajemnej niezależności predyktorów (zmiennych niezależnych). Często nie mają one żadnego związku z rzeczywistością i właśnie z tego powodu nazywa się je naiwnymi. Bardziej opisowe jest określenie – „model cech niezależnych”. Ponadto model prawdopodobieństwa można wyprowadzić korzystając z twierdzenia Bayesa.
Do zaimplementowania metody wykorzystano bibliotekę naivebayes. Podział zbiorów: 80% zbiór treningowy, 20% zbiór testujący.

Zbiór Haberman's Survival Data Set

Użyty kod:

```
# zbior 1
haberman <- read.table("https://archive.ics.uci.edu/ml/machine-learning-databases/
haberman/haberman.data",header = F, sep = ',')

folds <- haberman$V3
rows <- which(folds == 2, arr.ind = TRUE)
train.set <- haberman[-rows,]
test.set <- haberman[rows,]
model.bayes <- naiveBayes(x = train.set[,-4], y = train.set[,4])
bayes.result <- predict(model.bayes, test.set[,-4])
error.bayes <- 1 - sum(bayes.result == test.set[,4]) / length(test.set[,4])
print(paste("Blad:", round(error.bayes * 100, 2), "%"))
print(paste("Dokladnosc: ", (sum(bayes.result == test.set[,4]) / length(test.set[,4])) * 100, "%"))
```

Otrzymane wyniki:

```
[1] "Blad: 25 %"
[1] "Dokladnosc:  75 %"
```

Zbiór South German Credit Data Set

Użyty kod:

```r
# zbior 2
southGerman <- read.table("SouthGermanCredit.asc", header = TRUE)

folds <- cut(seq(1,nrow(southGerman)), breaks=5, labels=FALSE)
rows <- which(folds==5, arr.ind=TRUE)
southGerman.train <- southGerman[-rows,]
test.set <- southGerman[rows,]
model.bayes <- naiveBayes(x = southGerman.train[,-21], y = southGerman.train[,21])
bayes.result <- predict(model.bayes, test.set[,-21])
error.bayes <- 1-sum(bayes.result == test.set[,21])/length(test.set[,21])
print(paste("Blad:", round(error.bayes * 100, 2), "%"))
print(paste("Dokladnosc: ",(sum(bayes.result == test.set[,21]) / length(test.set[,21])) * 100, "%"))
```

Otrzymane wyniki:

```
[1] "Blad: 3 %"
[1] "Dokladnosc:  97 %"
```

Wnioski:

Metoda Bayes’a uzyskała dużo lepszy wynik dla zbioru South German Credit Data Set. Dokładność wyniosła 97% i jest o 22% wyższa od dokładności dla zbioru Haberman’s Survival Data Set (75%).

Metoda Neuralnet

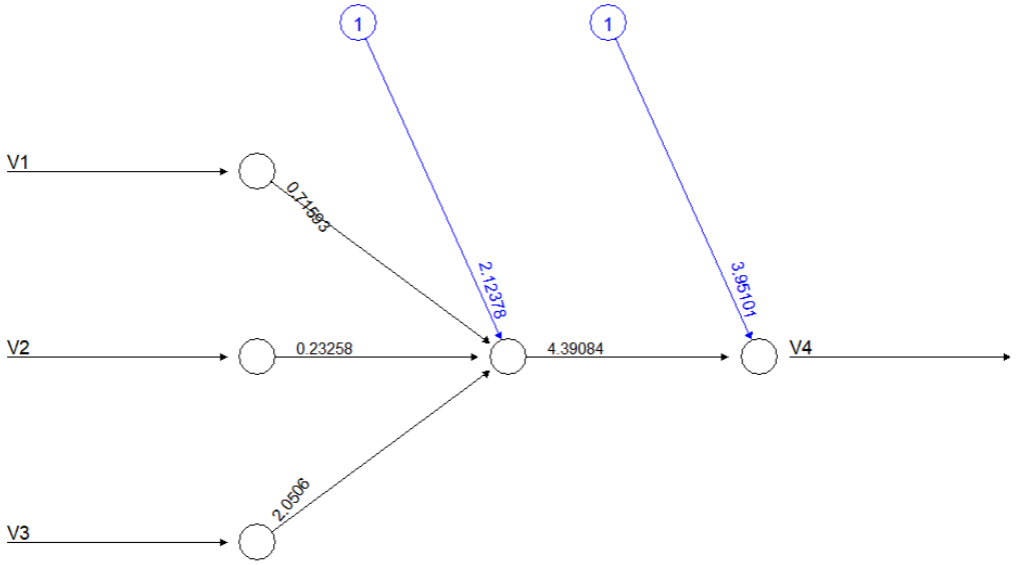
Metoda sieci neuronowej wewnętrznie rozdziela rekordy między podzbiór budowania modelu oraz zbiór zabezpieczający przed “przeuczeniem”, który jest niezależnym zbiorem rekordów danych używanym do śledzenia błędów podczas uczenia i zapobiegania modelowaniu przez metodę zmienności prawdopodobieństwa w danych.
Do zaimplementowania metody wykorzystano biblioteki neuralnet i kernlab. Podział zbiorów: 80% zbiór treningowy, 20% zbiór testujący.

Zbiór Haberman's Survival Data Set

Użyty kod:

```
# zbior 1
haberman <- read.table("https://archive.ics.uci.edu/ml/machine-learning-databases/
haberman/haberman.data",header = F, sep = ',')

rows.haberman <- sample.int(nrow(haberman), size = round(nrow(haberman)/2), replace = F)
haberman.train <- haberman[-rows.haberman,]
haberman.test <- haberman[rows.haberman,]
model <- neuralnet(formula = V2 ~., data = haberman.train, linear.output = F)
result <- predict(model, haberman.test)
print(model$result.matrix)
plot(model)
```

Otrzymane wyniki:



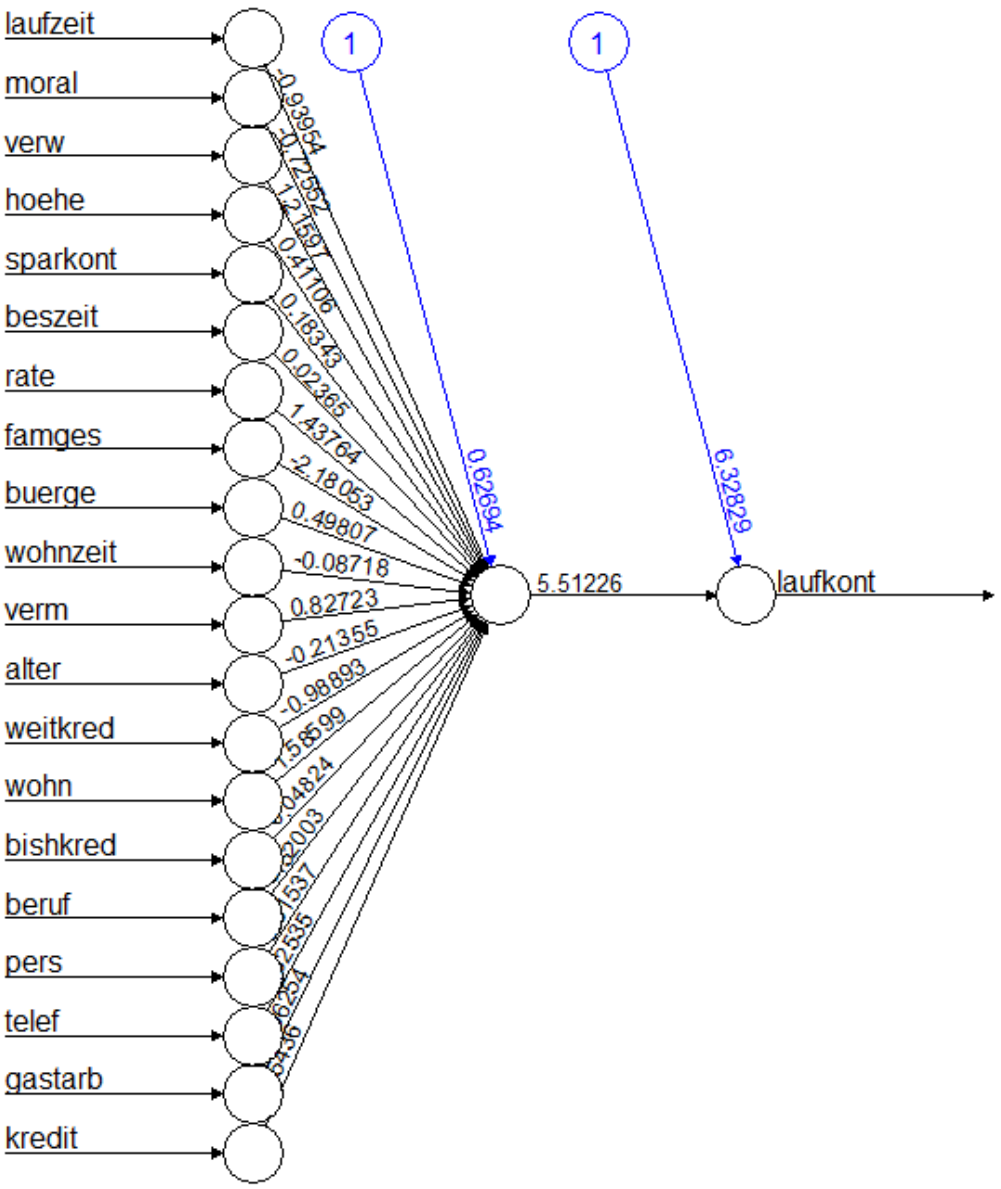
Wykres ukazuje trzy neurony w warstwie ukrytej, z wagami: 0.71593, 0.23258, 2.0506 i jeden w warstwie wejściowej o wadze 4.39084. Błąd wyniósł w przybliżeniu 19.51%.

Zbiór South German Credit Data Set

Użyty kod:

```
# zbior 2
southGerman <- read.table("SouthGermanCredit.asc", header = TRUE)

rows.data2 <- sample.int(nrow(southGerman), size = round(nrow(southGerman)/5), replace = F)
train.set <- southGerman[-rows.data2,]
test.set <- southGerman[rows.data2,]
model <- neuralnet(formula = hoehe~., data = train.set, hidden=c(3,1),linear.output = F)
result <- predict(model, test.set)
plot(model)
model$result.matrix
```

Otrzymane wyniki:



Wykres przedstawia 20 neuronów w warstwie ukrytej i jeden w warstwie wyjściowej o wadze 5.51226. Błąd wyniósł w przybliżeniu 15%.

Wnioski:

Metoda uzyskała gorszą dokładność dla zbioru Haberman’s Survival Data Set, wyniosła ona 80,49%. Dokładność dla drugiego zbioru była równa 85%.

Metody grupowania

Wybrana metoda:

- k średnich (k-means)

Metoda k - średnich

Algorytm k-średnich (z ang. k-means) inaczej zwany również algorytmem centroidów, służy do podziału danych wejściowych na z góry założoną liczbę klas. Jest to jeden z algorytmów stosowany w klasteryzacji (grupowaniu) .
Do zaimplementowania metody wykorzystano biblioteki cluster i ClusteR. Ilość grup, na jakie podzielone zostały zbiory, zależna była od ilości klas atrybutu decyzyjnego.

Zbiór Haberman's Survival Data Set

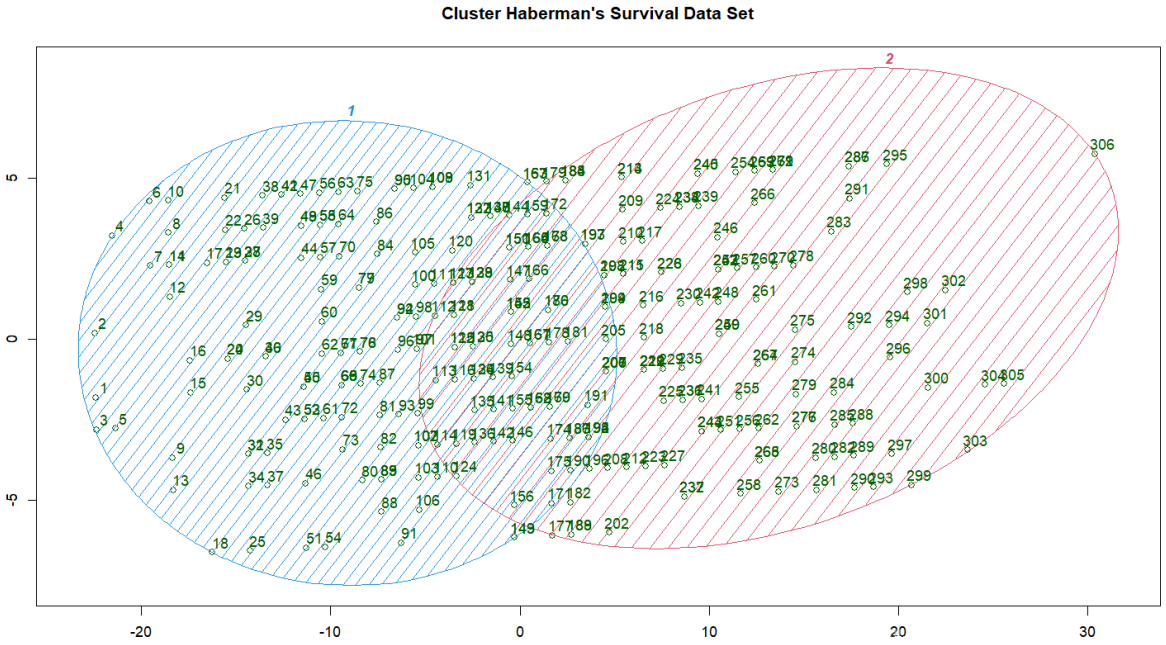
Użyty kod:

```r
# Fitting K-Means clustering Model
# to training dataset
set.seed(240) # Setting seed
kmeans.re <- kmeans(haberman, centers = 2, nstart = 20)
kmeans.re

# Confusion Matrix
cm <- table(haberman$V4, kmeans.re$cluster)
cm

# Model Evaluation and visualization
plot(haberman[c("V1", "V2")],
     col = kmeans.re$cluster,
     main = "K-means with 2 clusters")

## Visualizing clusters
y_kmeans <- kmeans.re$cluster
clusplot(haberman[, c("V1", "V2")],
         y_kmeans,
         lines = 0,
         shade = TRUE,
         color = TRUE,
         labels = 2,
         plotchar = FALSE,
         span = TRUE,
         main = paste("Cluster Haberman's Survival Data Set"),
         xlab = 'V1',
         ylab = 'V2')
```

Otrzymane wyniki:

```
K-means clustering with 3 clusters of sizes 152, 30, 124

Cluster means:
       V1       V2       V3
1 44.31579 62.57237  2.289474
2 51.50000 63.06667 21.666667
3 62.66935 63.14516  1.887097

Clustering vector:
  [1] 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1
 [58] 1 1 2 1 1 2 1 1 1 2 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 2 1 1 1 1 1
[115] 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 1 1 1 1
[172] 3 3 3 2 3 3 2 3 3 3 2 3 3 3 2 3 3 2 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 2 3 3
[229] 3 3 3 3 3 3 3 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 2 3 2 3 3 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
[286] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3

Within cluster sum of squares by cluster:
[1] 8823.309 4700.033 7641.250
 (between_SS / total_SS =  61.2 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss" "betweenss"    "size"
[8] "iter"         "ifault"
```

Wynik przedstawia model kmeans. Widzimy, że suma kwadratów reszt wyniosła 61.2%.



Obraz przedstawia dane zbioru Haberman’s Survival Data Set podzielone na dwie grupy. Grupowanie zostało przeprowadzone dla dwóch grup, ponieważ atrybut decyzyjny zawiera dwie klasy.

Zbiór South German Credit Data Set

Użyty kod:

```r
# zbior 2

# Fitting K-Means clustering Model
# to training dataset
set.seed(240) # Setting seed
kmeans.re <- kmeans(southGerman, centers = 4, nstart = 20)
kmeans.re

# Confusion Matrix
cm <- table(southGerman$laufkont, kmeans.re$cluster)
cm

# Model Evaluation and visualization
plot(southGerman[c("laufzeit", "rate")],
     col = kmeans.re$cluster,
     main = "K-means with 3 clusters")

## Visualizing clusters
y_kmeans <- kmeans.re$cluster
clusplot(southGerman[, c("laufzeit", "rate")],
         y_kmeans,
         lines = 0,
         shade = TRUE,
         color = TRUE,
         labels = 2,
         plotchar = FALSE,
         span = TRUE,
         main = paste("Cluster South German Credit Data Set"),
         xlab = 'laufzeit',
         ylab = 'rate')
```

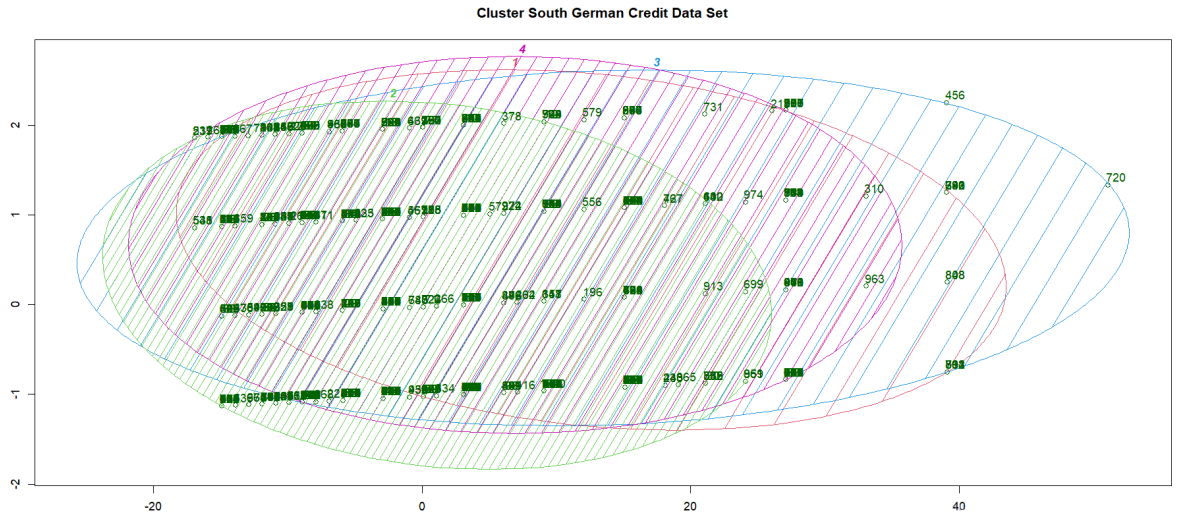
Otrzymane wyniki:

```
K-means clustering with 3 clusters of sizes 56, 728, 216

Cluster means:
  laufkont laufzeit    moral
1 2.517857 39.66071 2.303571
2 2.612637 16.71841 2.561813
3 2.472222 30.14352 2.550926
      verw     hoehe sparkont
1 3.214286 11695.589 2.392857
2 2.715659  1890.062 2.070055
3 3.106481  5742.269 2.148148
   beszeit     rate   famges
1 3.339286 2.285714 2.642857
2 3.401099 3.126374 2.682692
3 3.337963 2.634259 2.689815
    buerge wohnzeit     verm
1 1.107143 2.839286 3.250000
2 1.157967 2.821429 2.190934
3 1.111111 2.925926 2.689815
     alter weitkred     wohn
1 36.05357 2.535714 2.196429
2 35.31868 2.692308 1.884615
3 36.16204 2.652778 2.004630
  bishkred    beruf     pers
1 1.375000 3.339286 1.839286
2 1.383242 2.813187 1.851648
3 1.495370 3.097222 1.824074
     telef  gastarb
1 1.821429 1.964286
2 1.335165 1.954670
3 1.527778 1.990741

Clustering vector:
   [1] 2 2 2 2 2 2 2 2 2 2 3 3
  [13] 2 3 2 2 3 2 2 3 2 2 2 2
  [25] 3 3 2 2 2 3 2 2 3 2 2 2
  [37] 2 2 2 3 2 2 2 2 2 2 2 2
  [49] 2 2 2 3 2 2 3 2 2 2 2 2
  [61] 2 2 2 1 2 2 2 2 2 2 2 2
  [73] 2 2 2 2 2 2 2 2 2 2 3 2
  [85] 2 2 2 3 2 2 2 2 3 2 3 3
  [97] 2 3 2 3 2 2 3 3 2 2 2 2
 [109] 2 2 3 2 2 2 3 3 2 2 2 2
 [121] 3 3 2 2 2 3 2 2 2 2 2 2
 [133] 2 2 2 2 2 2 2 2 3 3 3 2
 [145] 2 2 2 2 2 2 2 2 1 2 2 2
 [157] 2 3 3 2 2 2 3 2 2 2 3 3
 [169] 2 2 3 2 2 2 2 2 3 2 2 1
 [181] 2 2 3 2 2 3 2 2 2 2 2 2
```

```
[193]  3  2  2  3  3  2  2  2  2  3  2  3
[205]  2  3  2  2  1  2  2  2  2  2  2  3
[217]  2  2  1  2  2  2  3  3  2  2  3  2
[229]  2  2  2  2  3  3  2  2  2  2  2  2
[241]  2  1  2  2  2  2  2  3  2  3  2  3
[253]  2  2  3  2  2  2  2  2  2  2  2  2
[265]  2  2  2  2  2  2  2  3  2  2  2  2
[277]  2  2  2  2  2  2  3  2  2  2  2  2
[289]  2  2  2  2  2  2  2  3  2  2  2  2
[301]  2  3  2  2  2  2  2  2  2  1  2  2
[313]  2  2  2  2  2  2  2  2  2  2  2  2
[325]  2  2  2  2  2  3  2  2  2  2  2  2
[337]  2  2  2  2  2  2  2  2  2  2  2  2
[349]  2  2  2  2  3  2  2  1  2  2  2  3
[361]  2  2  2  2  2  2  2  2  2  2  3  2
[373]  3  2  2  2  2  3  2  3  2  2  1  2
[385]  3  2  2  3  2  2  2  2  2  2  3  2
[397]  3  2  2  3  2  2  2  2  3  3  3  2
[409]  2  3  2  2  2  2  2  2  2  3  2  2
[421]  2  2  2  2  2  2  1  2  2  2  3  2
[433]  2  2  2  2  2  2  2  2  2  3  1  2
[445]  2  2  2  2  3  2  2  2  3  3  3  3
[457]  2  2  2  2  3  3  2  3  3  2  2  3
[469]  2  2  2  2  3  2  3  2  2  2  2  3
[481]  3  2  3  3  2  2  2  2  2  2  2  2
[493]  2  2  2  2  2  3  2  2  2  2  2  2
[505]  3  2  2  3  3  2  2  2  3  3  2  2
[517]  3  2  3  3  2  1  2  2  2  1  2  2
[529]  2  2  2  2  2  2  2  2  2  2  2  2
[541]  2  3  2  2  2  2  2  2  2  2  2  2
[553]  2  2  2  2  3  2  3  2  2  3  3  2
[565]  3  2  1  2  2  1  2  1  3  2  3  1
[577]  2  2  3  3  2  2  2  2  3  2  3  2
[589]  2  2  2  2  3  2  3  3  2  2  2  3
[601]  3  2  2  2  2  2  2  2  3  2  2  2
[613]  2  2  3  2  1  2  2  2  1  3  2  2
[625]  3  2  2  2  2  2  2  3  3  3  2  2
[637]  2  2  3  2  2  3  3  3  2  1  2  3
[649]  2  2  3  2  2  2  3  2  2  2  2  2
[661]  2  2  2  2  2  2  2  2  2  2  2  2
[673]  2  2  2  2  2  2  2  2  2  2  3  2
[685]  2  2  2  3  2  2  1  2  2  2  2  1
[697]  2  2  3  2  3  3  3  2  2  2  2  2
[709]  3  3  2  2  3  2  2  2  3  3  2  3
[721]  3  2  2  1  3  2  2  2  1  1  1  3
[733]  2  2  3  2  2  2  2  3  2  2  2  3
[745]  3  2  2  3  2  2  2  2  2  3  1  2
[757]  2  2  2  2  2  2  3  3  2  2  3  2
[769]  2  2  2  2  2  1  2  2  3  2  2  3
```

```
 [781] 3 2 1 2 2 2 2 2 3 3 2 2
 [793] 3 2 2 2 2 2 2 2 2 2 2 2
 [805] 2 2 2 3 2 2 1 3 2 2 3 2
 [817] 2 2 2 2 1 3 2 2 2 2 3 3
 [829] 2 3 2 3 3 3 3 1 3 3 3 2
 [841] 2 2 2 1 1 2 2 1 2 1 2 2
 [853] 2 2 2 3 2 2 3 2 3 3 2 2
 [865] 3 2 2 1 2 2 2 3 2 1 2 3
 [877] 3 3 2 2 2 2 3 2 2 3 2 3
 [889] 2 1 1 1 2 2 2 1 2 2 3 2
 [901] 3 2 2 3 2 2 2 2 2 3 3 2
 [913] 3 2 3 2 2 2 3 3 2 2 2 3
 [925] 2 2 2 2 1 1 2 2 2 2 2 2
 [937] 2 2 2 3 1 2 2 1 2 3 2 3
 [949] 2 2 3 2 2 1 2 2 2 2 3 2
 [961] 3 2 1 2 2 1 3 2 2 2 3 3
 [973] 2 1 2 1 1 1 2 2 2 2 3 2
 [985] 3 2 1 2 2 2 2 1 3 3 3 2
 [997] 2 1 3 3

Within cluster sum of squares by cluster:
[1] 284497168 566458954
[3] 427282030
 (between_SS / total_SS =  83.9 %)

Available components:

[1] "cluster"
[2] "centers"
[3] "totss"
[4] "withinss"
[5] "tot.withinss"
[6] "betweenss"
[7] "size"
[8] "iter"
[9] "ifault"
```

Wynik przedstawia model kmeans. Widzimy, że suma kwadratów reszt wyniosła 83.9%.



Obraz przedstawia dane zbioru South German Credit Data Set podzielone na cztery grupy. Grupowanie zostało przeprowadzone dla czterech grup, ponieważ atrybut decyzyjny zawiera cztery klasy.