Przedmiot: Sztuczna Inteligencja

TEMAT: Użycie i analiza różnych metod klasyfikacji danych

Imię Nazwisko Grupa : Iga Tupin L6

Rozdział 1

Wprowadzenie: zbiór danych

1.1 Opis zbioru danych

Do pracy nad projektem użyto zbioru "Contraceptive Method Choice Data Set" - który powstał w ramach badania metod antykoncepcji wśród indonezyjskich par i możliwym wpływie wybranych czynników na ich wybór.
Za wyborem zbioru zadecydował odpowiadający zadaniu format danych, optymalna dla wygodnej pracy ilość atrybutów oraz relatywnie duża ilość instancji.

1.2 Struktura zbioru danych

Ilość atrybutów: 9
Ilość instancji: 1473
Lista atrybutów:

1. Wiek żony
2. Wykształcenie żony (1=brak,2=podstawowe,3=średnie,4=wyższe)
3. Wykształcenie męża (1=brak,2=podstawowe,3=średnie,4=wyższe)
4. Ilość dzieci urodzonych w rodzinie
5. Religia żony (0=nie islam, 1=islam)
6. Czy żona pracuje? (0=tak, 1=nie)
7. Okupacja męża (kategoria od 1 do 4)

8. Standard życia (kategoria od 1 do 4)

9. Ekspozycja na media (0=wysoka, 1=niska)

10. Metoda antykoncpecji (1=brak, 2=długoterminowa, 3=krótkoterminowa)

Rozdział 2

Metoda KNN

2.1 Opis metody

Metoda KNN - czyli metoda najbliższych sąsiadów (ang. k-Nearest Neighbors) polega na przedstawieniu danych treningowych w postaci wektorów (Xn,Y) zmiennych objaśniających (Xn) oraz wartości zmiennej objaśnianej (Y). W dalszym kroku przyporządkowuje Y dla danych testowych na podstawie Xn porównując je do k-danych testowych o najbliższych wartościach Xn (sąsiadów). W celu zminimalizowania napotkanych problemów, zmienna k powinna być zawsze liczbą nieparzystą.

2.2 Klasyfikacja metodą KNN

Poniższy kod przedstawia 100-krotną klasyfikację danych metodą KNN dla proporcji TEST/TRAIN 20%/80% zestawu danych oraz liczby k=97. Wyświetla średnią trafność dla wszystkich prób oraz odchylenie standardowe.

```
1  library(class)
2
3  data = read.csv("D:\\szkulka\\semestr V\\AI\\projekt\\cmc.data")
4
5  colnames(data) <- c("wife.age",
6                      "wife.education",
7                      "husband.education",
8                      "NO.children",
9                      "wife.religion",
10                     "wife.work",
11                     "husband.occupation",
```

```
                    "living.standard",
                    "media.exposure",
                    "CM")
#deklaracja i definicja mnożnika proporcji
p <- 0.8

#deklaracja zmiennych potrzebnych do późniejszej analizy trafnośc
total_accuracy = 0
accuracy_array = matrix(data=NA, nrow=1, ncol=100)

set.seed(1)

for(i in 1:100){

  train.index <- sample.int(nrow(data),nrow(data)*p)
  data.train <- data[train.index,]
  data.test <- data[-train.index,]

  pred_knn = knn(data.train[,1:9],
                 data.test[,1:9],
                 data.train$CM,
                 k = 97)

  #confusion matrix
  cf <- table(data.test$CM,pred_knn)
  #pobieranie danych do obliczenia odchylenia
  accuracy_array[1,i] <- sum(diag(cf))/sum(cf)
  #obliczenia do sredniej ze 100 prob
  total_accuracy <- total_accuracy + sum(diag(cf))/sum(cf)

}

#obliczanie średniej trafności dla stu prób (total_accuracy)
total_accuracy <- total_accuracy/100
total_accuracy

```

```
48 #obliczanie odchylenia standardowego (sd)
49 tmp = 0
50 for(i in 1:100){
51   tmp <- tmp + (accuracy_array[1,i] - total_accuracy)
52 }
53 sd <- sqrt((tmp * tmp)/100)
54 sd
```

Rys. 2.1: KNN

Dane wyjściowe:

```
> total_accuracy
[1] 0.5359661
> sd
[1] 4.870537e-15
```

Confusion Matrix dla jednej próby:

```
> cf
   pred_knn
     1  2  3
  1 77  6 30
  2 23 24 26
  3 39 12 58
```

2.3 Analiza zmian dla różnych proporcji TEST/TRAIN

Spróbujmy podnieść jakość klasyfikatora zmianiając proporcje TEST/TRAIN (zmienną p w przedstawionym kodzie).

Proporcja: 50%/50%

```
> total_accuracy
[1] 0.5166304
> sd
[1] 2.547962e-15
> cf
   pred_knn
      1   2   3
  1 169  20 128
  2  41  38  92
  3  69  27 152
```

Proporcja: 40%/60%

```
> total_accuracy
[1] 0.529966
> sd
[1] 1.010303e-15
> cf
   pred_knn
      1   2   3
  1 161  21  60
  2  27  45  57
  3  60  33 125
```

Proporcja: 30%/70%

```
> total_accuracy
[1] 0.5375792
> sd
[1] 2.209344e-15
> cf
   pred_knn
      1   2   3
  1 118  10  47
  2  23  35  36
  3  60  20  93
```

Najlepszy wyniki uzyskano dla proporcji 30%/70%. Trafność jest wyższa niż w przypadku proporcji 20%/80%, a rozrzut (odchylenie standardowe) niższy.

2.4 Analiza zmian dla różnych wartości parametru K

Spróbujmy ustalić bardziej optymalną wielkość parametru K, pamiętając, że powinien on być liczbą nieparzystą.

k=3

```
> total_accuracy
[1] 0.5038462
> sd
[1] 1.44329e-16
> cf
   pred_knn
     1  2  3
  1 90 25 52
  2 27 44 25
  3 56 29 94
```

k=13

```
> total_accuracy
[1] 0.5397059
> sd
[1] 5.268008e-15
> cf
   pred_knn
      1   2   3
  1 100  19  60
  2  21  40  43
  3  40  27  92
```

k=23

```
> total_accuracy
[1] 0.5382127
> sd
[1] 8.359979e-15
> cf
   pred_knn
      1   2   3
  1 112  23  66
  2  24  27  49
  3  27  25  89
```

k=47

```
> total_accuracy
[1] 0.5391629
> sd
[1] 2.153833e-15
> cf
   pred_knn
     1  2  3
  1 94 19 56
  2 32 35 45
  3 44 23 94
```

k=67

```
> total_accuracy
[1] 0.5365837
> sd
[1] 3.574918e-15
> cf
   pred_knn
      1   2   3
  1 119  13  63
  2  28  40  35
  3  42  17  85
```

k=87

```
> total_accuracy
[1] 0.5323529
> sd
[1] 2.214895e-15
> cf
   pred_knn
      1   2   3
  1  95  12  83
  2  15  18  67
  3  27  12 113
```

Co prawda dla najniższego podanego parametru (k=3) można zauważyć pogorszenie trafności, jednak dla wartości powyżej 10 różnice są nieznaczące.

Rozdział 3

Metoda Naive-Bayes

3.1 Opis metody

Naiwny klasyfikator Bayes'a to relatywnie prosty klasyfikator probabilistyczny. Jego działanie jest oparte na założeniu prawdopodobieństwa, że dana cecha należy do konkretnego obiektu. Nie rozważa jednak powiązań między tymi cechami, stąd określenie "naiwny". Pomimo tego, metoda Bayes'a jest uznawana za osiągającą wysoki poziom dokładności. Zaletą tej metody jest to, że potrzebuje względnie niewielkiej liczby danych treningowych do oszacowania parametrów niezbędnych do klasyfikacji.

3.2 Klasyfikacja metodą Naive-Bayes

Poniższy kod przedstawia 100-krotną klasyfikację danych metodą Naive_Bayes dla proporcji TEST/TRAIN 20%/80% zestawu danych. Wyświetla średnią trafność dla wszystkich prób oraz odchylenie standardowe.

```
1  library(naivebayes)
2
3  data = read.csv("D:\\szkulka\\semestr V\\AI\\projekt\\cmc.data")
4
5  colnames(data) <- c("wife.age",
6                      "wife.education",
7                      "husband.education",
8                      "NO.children",
9                      "wife.religion",
10                     "wife.work",
11                     "husband.occupation",
```

```
                      "living.standard",
                      "media.exposure",
                      "CM")

data$CM <- as.factor(data$CM)

#deklaracja i definicja mnożnika proporcji
p <- 0.8

#deklaracja zmiennych potrzebnych do późniejszej analizy trafnośc
total_accuracy = 0
accuracy_array = matrix(data=NA, nrow=1, ncol=100)


for(i in 1:100){
  train.index <- sample.int(nrow(data),nrow(data)*p)
  data.train <- data[train.index,]
  data.test <- data[-train.index,]

  model=naive_bayes(CM ~ ., data=data.train, usekernel=T)
  plot(model)

  pred=predict(model, data.test)

  #confusion matrix
  cf <- table(data.test$CM,pred)
  #pobieranie danych do obliczenia odchylenia
  accuracy_array[1,i] <- sum(diag(cf))/sum(cf)
  #obliczenia do sredniej ze 100 prob
  total_accuracy <- total_accuracy + sum(diag(cf))/sum(cf)
}

#obliczanie średniej trafności dla stu prób (total_accuracy)
total_accuracy <- total_accuracy/100

#obliczanie odchylenia standardowego (sd)
```

```
48 tmp = 0
49 for(i in 1:100){
50   tmp <- tmp + (accuracy_array[1,i] - total_accuracy)
51 }
52
53 odchylenie <- sqrt((tmp * tmp)/100)
54
55 total_accuracy
56 odchylenie
57 cf
```

Rys. 3.1: Naive Bayes

Dane wyjściowe:

```
> total_accuracy
[1] 0.5016271
> odchylenie
[1] 3.164136e-16
```

Confusion Matrix dla jednej próby:

```
> cf
   pred
     1  2  3
  1 51 18 65
  2  8 26 31
  3  7 20 69
```

3.3 Analiza zmian dla różnych proporcji TEST/TRAIN

Spróbujmy podnieść jakość klasyfikatora zmianiając proporcje TEST/TRAIN (zmienną p w przedstawionym kodzie).

Proporcja: 50%/50%

```
> total_accuracy
[1] 0.5052853
> odchylenie
[1] 2.353673e-15
> cf
   pred
      1   2   3
  1 123  93 103
  2  17  89  45
  3  34  85 147
```

Proporcja: 40%/60%

```
> total_accuracy
[1] 0.5034975
> odchylenie
[1] 2.942091e-16
> cf
   pred
      1   2   3
  1 113  53  77
  2  18  60  63
  3  36  57 112
```

Proporcja: 30%/70%

```
> total_accuracy
[1] 0.5063122
> odchylenie
[1] 6.605827e-16
> cf
   pred
     1  2  3
  1 93 37 59
  2 17 53 17
  3 20 57 89
```

W przypadku metody Naive Bayes zmiana proporcji TEST/TRAIN nie czyni widocznych różnic.

Rozdział 4

Metoda Random Forest

4.1 Opis metody

Metoda klasyfikacji Random Forest to rozwinięcie metody drzewa decyzyjnego - stąd nazwa lasu, ponieważ składa się na nią wiele ”drzew”. Algorytm na podstawie danych treningowych tworzy wiele drzew decyzyjnych używając losowo dobranych atrybutów. Przy przewidywaniu klasy z użyciem danych testowych używa utworzonych drzew i przypisuje danym klasę, która wynika z większości z nich.

4.2 Klasyfikacja metodą Random Forest

Poniższy kod przedstawia 100-krotną klasyfikację danych metodą Random Forest. Wyświetla średnią trafność dla wszystkich prób oraz odchylenie standardowe.
Uwaga: proporcja TEST/TRAIN jest w tym przypadku mniej dostępna, ponieważ jest ustalana już wewnątrz funkcji randomForest(). Na ogół przy tej metodzie dane treningowe stanowią około 2/3 danych.

```
1  library(ggplot2)
2  library(cowplot)
3  library(randomForest)
4
5  data = read.csv("D:\\szkulka\\semestr V\\AI\\projekt\\cmc.data")
6
7  colnames(data) <- c("wife.age",
8                      "wife.education",
9                      "husband.education",
10                     "NO.children",
```

```
                      "wife.religion",
                      "wife.work",
                      "husband.occupation",
                      "living.standard",
                      "media.exposure",
                      "CM")
data$CM <- as.factor(data$CM)

#deklaracja zmiennych potrzebnych do obliczenia średniej i odchyl
total_accuracy = 0
accuracy_array = matrix(data=NA, nrow=1, ncol=100)


set.seed(3)


for(i in 1:100){

  model <- randomForest(CM~., data=data, proximity=TRUE)

  #confusion matrix wchodzi w skład funkcji randomForest
  accuracy_array[1,i] <- sum(diag(model$confusion))/sum(model$con
  total_accuracy <- total_accuracy + sum(diag(model$confusion))/s
}

#obliczanie średniej trafności dla stu prób (total_accuracy)
total_accuracy <- total_accuracy/100

#obliczanie odchylenia standardowego (sd)
tmp = 0
for(i in 1:100){
  tmp <- tmp + (accuracy_array[1,i] - total_accuracy)
}

odchylenie <- sqrt((tmp * tmp)/100)

```

```
47 plot(model)
48
49 total_accuracy
50 odchylenie
51 model
```

Rys. 4.1: RandomForest

Dane wyjściowe:

```
> total_accuracy
[1] 0.5374769
> odchylenie
[1] 1.472877e-15
```

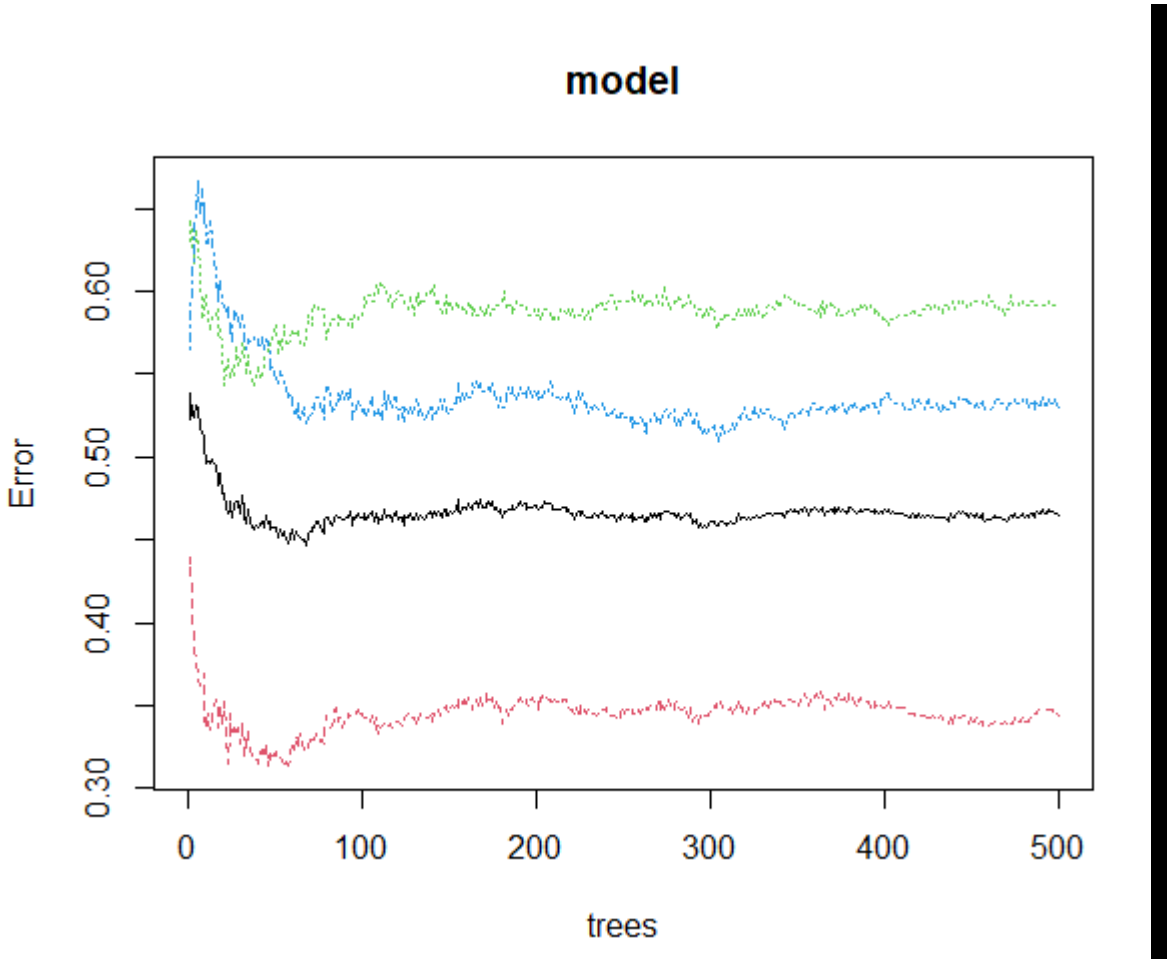
Confusion Matrix dla jednej próby:

```
Confusion matrix:
    1   2   3 class.error
1 412  63 153   0.3439490
2  97 136 100   0.5915916
3 165 106 240   0.5303327
```

funkcja randomForest() zwraca również błędy obliczone dla każdej z klas osobno. To potwierdza pewną obserwację z poprzednich metod - klasyfikacja pierwszej z klas jest trafna dużo częściej niż klasyfikacja pozostałych. Podobną zależność można zauważyć analizując trafności wcześniej omówionych metod.

4.3 (

Analiza zmian dla różnej ilości drzew) Domyślna ilość drzew to 500. Wykres błędu dla takiej ilości w przypadku omawianego zbioru danych wygląda następująco.



Widocznym jest, że błąd szybko się stabilizuje i nie ma znaczenia ile setek drzew wybierzemy. Jednak moim zdaniem warto sprawdzić mniejszą ilość drzew, jako że z wykresu wynika, że wszystkie błędy są niższe przed dodaniem setnego drzewa.

```
> total_accuracy
[1] 0.5317574
> odchylenie
[1] 3.996803e-16
```

Sprawdźmy więc wyniki dla 80 drzew.

```
Confusion matrix:
    1   2   3 class.error
1 411  66 151   0.3455414
2  97 134 102   0.5975976
3 167 106 238   0.5342466
```

Mimo zmiany ilości drzew, w wynikach nie stało nic się wartego uwagi.

Rozdział 5

Podsumowanie i wnioski

Przy wybranym zbiorze danych klasyfikacja metodą Naive-Bayes okazała się przynieść najmniej zadowalające efekty. Metody KNN oraz Random Forest uzyskiwały stabilnie średnio prawie 54% trafności.

Taki poziom trafności sugeruje, że zbadane czynniki nie mają na tyle decydującego wpływu na wybory indonezyjskich par, aby z pewnością być w stanie je przewidzieć. Co nie znaczy, że tego wpływu nie ma wcale.
Algorytm musi ustalić jedną z trzech metod antykoncpecji, więc przy wybieraniu odpowiedzi losowo powinien uzyskać trafność bliską 33%. Ta trafność jednak była wyższa dla wszystkich metod klasyfikacji.
Ponadto, trafność klasowa przy wyborze pierwszym (brak antykoncpecji) była widocznie wyższa niż inne w każdym przypadku, co oznacza, że na podstawie tych danych dużo łatwiej jest ocenić czy antykoncpecja w ogóle jest, natomiast jej konkretny rodzaj widocznie nie ma dużego związku ze zbadanymi czynnikami.

Link do tego opracowania:

https://www.overleaf.com

Literatura

- en.wikipedia.org/wiki/k-nearest-neighbors-algorithm
- en.wikipedia.org/wiki/Naive-Bayes-classifier
- en.wikipedia.org/wiki/Random-forest
- www.youtube.com/@statquest