Dokumentacja projektu - PADR

Joanna Rancew

Heart Disease Dataset

Dane analizowane w projekcie pochodza z ogolnodostepnej bazy danych i dotycza aspektow zwiazanych z choroba serca.

Cele projektu

Celem projektu byla analiza oraz prezentacja danych przedstawionych w zbiorze "Heart Disease Dataset". Podczas pracy nad programem wielokrotnie mialam okazje nabyc nowe informacje i wykorzystac zdobyta wiedze w praktyce.

Z zalozenia program mial spelniac ponizsze funkcje:

- poprawne wczytywanie i obrobka danych
- polaczenie zbiorow danych
- wyswietlenie danych w postaci histogramu w zaleznosci od wybranych parametrow
- przewidywanie diagnozy z uzyciem drzew decyzyjnych
- prezentacja danych w postaci boxplotu
- prezentacja tekstowa przewidywanej diagnozy po wypelnieniu testu
- korelogram - prezentacja korelacji wynikow badan i diagnozy
- zastosowanie mozliwosci xgBoost do analizy i klasyfikacji

Zrodla danych

- Hungarian Institute of Cardiology. Budapest: Andras Janosi, M.D.
- University Hospital, Zurich, Switzerland: William Steinbrunn, M.D.
- University Hospital, Basel, Switzerland: Matthias Pfisterer, M.D.

Dane zostaly odpowiednio dostosowane do uzytku w programie

Informacje podstawowe

Baza danych zawiera 76 zmiennych. W oparciu o nia przeprowadzaone byly eksperymenty. Do eksperymentow brane byly pod uwage ponizsze zmienne z bazy danych.

Zmienne

1. age - wiek w liczbach
2. sex - plec, 1 - mezczyzna, 0 - kobieta
3. cp - typ bolu w klatce (1- typowa dusznosc, 2 - nietypowa, 3 - inny rodzaj bolu, 4 - bez objawow)
4. trestbps - cisnienie spoczynkowe krwi w mm Hg
5. chol - poziom cholesterolu w mg/dl
6. fbs - cukier we krwi (1- powyzej 120 mg/dl, 0 - ponizej 120 mg/dl)
7. restecg - 0 w normie, 1 - odchylki od normalnosci na odcinku ST-T, 2 - przerost lewej komory wg kryteriow Estesa
8. thalach - maksymalne tetno podczas wysilku
9. exang - czy wystapila dusznosc, dlawica wysilkowa
10. diag - diagnoza (0- zdrowy, 1,2,3,4- rozny stopien zaawansowania choroby)
11. painloc - lokalizacja bolu (podsrodkowa - 1, inna - 0)
12. painexer - powod bolu (prowokowany wysilkiem - 1, inny - 0)
13. relrest - czy bol ustaje po odpoczynku
14. smoke - czy pacjent jest palacy
15. cigs - liczba papierosow dziennie
16. years - liczba lat palenia

Efekt pracy, opisywane w dokumentacji wykresy i program dostepne sa pod adresem: Moja aplikacja shiny

Stosowane biblioteki

```
library(shiny)
library(dplyr)
library(ggplot2)
library(readxl)
library("stringr")
library(DT)
library(ggpubr)
library(tidyr)
library(corrplot)
library(rpart)
library(rpart.plot)
library(ggcorrplot)
library(party)
library(tree)
library(rattle)
```

Dzialanie programu

Wprowadzanie danych

Wprowadzanie danych realizowane jest w kilku etapach. Dane poczatkowo byly przechowywane w nieuporzadkowanym pliku *.data*, ktory wymaga odpowiedniego przygotowania. Na poczatku zczytywane sa kolejne linie i segregowane po zmiennej "name", ktora jest ostatnio zmienna kazdego wiersza w oryginale.

```
data <- readLines("data/data_hungarian.data") #wczytuje linjki z pliku po kolei
#wylapuje indeksy kiedy pojawia sie "name"
ind <- grep("name",data)
data_proc <- vector("character", length(ind))
for(i in ind){

  data_proc[i/10] <- paste(data[(i-9)],data[i-8],data[i-7],data[i-6],data[i-5],data[i-4],
                           data[i-3],data[i-2],data[i-1],data[i])


  data_proc[i/10]<-gsub("name","NA",data_proc[i/10])
  data_proc[i/10]<-gsub("-9. ", "NA ",data_proc[i/10])
  data_proc[i/10]<-gsub("-9 ", "NA ",data_proc[i/10])
  data_proc[i/10]<-gsub(" ", ",",data_proc[i/10])
  #-9 oraz 9 oznaczaly nieprawidlowe lub brakujace wartosci
}
```

Odpowiednie przeksztalcanie danych

Otrzymane dane wymagaly przeksztalcenia, czyli:

- rozdzielenia elementow
- odpowiedniego zastapienia wartosci *NA*
- zmiany na typ *numeric*

```
df <- matrix(nrow=length(ind),ncol=76)
ind <- ind/10
for(i in ind){
  vector_test<-str_split(data_proc[i], ",")
 vector_test<- (unlist(vector_test))
 vector_test <- replace(vector_test, vector_test == "NA", NA)
 vector_test <- replace(vector_test, vector_test == "NA ", NA)
 vector_test<- as.numeric((vector_test))
  df[i,] <-vector_test
}
```

Przeksztalcenie na ramke danych i nazwanie kolumn

```
df <- as.data.frame(df)
df <- cbind(df$V3,df$V4,df$V9,df$V10,df$V12,df$V16,
            df$V19,df$V32,df$V38,df$V40,df$V41,df$V51,
            df$V58, df$V5,df$V6,df$V7,df$V13,df$V14,
            df$V15,df$V17,df$V18)

Hungarian_data <- as.data.frame(df)
colnames(Hungarian_data) <- c("age","sex","cp","trestbps",
                 "chol","fbs","restecg","thalach","exang",
                 "oldpeak","slope","thal","diag","painloc",
                 "painexer","relrest","smoke","cigs","years",
                 "dm","famhist")
```

Dokladnie takie same kroki zostaly wykonane dla plikow *data_va.data* oraz *data_switzerland.data*. Ze wzgledu na nieduza uzytecznosc dane z Cleveland zostaly odrzucone na etapie wprowadzenia.

Dostosowanie danych

Po wstepnym przetworzeniu danych, wymagaly one dodatkowej obrobki. Na tym etapie dolaczylam dodatkowa kolumne *diag2*, ktora okreslala konkretnie, czy dany pacjent jest zdrowy chory (0-zdrowy, 1-chory). Zastapilam tez niektore dane brakujace srednia wartoscia pozostalych danych, co zaprezentowane zostalo ponizej:

```
heart_data$diag2 <- heart_data$diag
heart_data$diag2[heart_data$diag2>0] = rep(1,length(heart_data$diag2[heart_data$diag2>0]))



heart_data$chol[heart_data$chol==0] <- mean(heart_data$chol, na.rm=TRUE)
heart_data$chol[is.na(heart_data$chol)] <- mean(heart_data$chol, na.rm=TRUE)

heart_data$trestbps[heart_data$trestbps==0] <- mean(heart_data$trestbps, na.rm=TRUE)

heart_data$fbs[is.na(heart_data$fbs)] <- rep(0,
               length(heart_data$fbs[is.na(heart_data$fbs)]))
heart_data$exang[is.na(heart_data$exang)] <- rep(0,
             length(heart_data$exang[is.na(heart_data$exang)]))

heart_data$relrest[is.na(heart_data$relrest)] <- rep(0,
              length(heart_data$relrest[is.na(heart_data$relrest)]))


heart_data = subset(heart_data, select = -c(slope,oldpeak,thal,dm,famhist))
```

Sekcja serwera

Po wprowadzeniu danych, zajelam sie odpowiednia obsluga ich w aplikacji. Korzystalam z dwoch plikow: ui.R oraz server.R, co pozwolilo na czytelne rozdzielenie interfejsu graficznego od dzialania serwera.

```
 result <- heart_data
  #modyfikacja danych obslugiwanych
  dane <- reactive(dplyr::filter(result, sex == input$jaka_plec &
                                   age >= input$jaki_wiek[1] & age <= input$jaki_wiek[2]))
  dane_palacz <- reactive(dplyr::filter(result, sex == 1 & smoke == input$czy_pali &
                                          age >= input$jaki_wiek[1] & age <= input$jaki_wiek[2]))
  #gdy jest jako reactive - to traktowane jest jako funkcja
```

Wykresy

W aplikacji, ktora odpowiada za obrobke, analize i prezentacje danych, kluczowym elementem sa wykresy. Zgodnie z zalozeniami projektu, prezentowalam dane o pacjentach w rozny sposob. Miedzy innymi uzywajac *boxplota, histogramu i wykresu gestosci wystepowania danej wartosci.*

Prezentacja danych dotyczaca palacych lub niepalacych mezczyzn: Dane zaprezentowaly zostaly przy uzyciu funkcji *geom_boxplot*, ktora pozwolila dobrze zaprezentowac roznice w spoczynkowym cisnieniu krwi zdrowych i chorych, w zaleznosci od tego, czy sa palaczami czy nie.

```
ggplot(dane_palacz(),aes(y=trestbps, x=as.factor(diag2)))+ geom_boxplot(na.rm=TRUE)+
   scale_x_discrete(labels=c("zdrowy", "chory"))+
      labs(y="Spoczynkowe cisnienie krwi",x="Wystepowanie choroby")+
      theme(legend.position="top")+theme(legend.title = element_blank())
```

Prezentacja danych dotyczacych poziomu cholesterolu Dane zaprezentowane zostaly z uzyciem funkci *geom_density*, ktora podobnie jak histogram, pozwala rzetelnie zaprezentowac czestosc wystepowania danych wartosci.

```
ggplot(dane(),aes( x=chol, color=as.factor(diag2)))+ geom_density(na.rm = TRUE)+
     scale_color_discrete(labels=c("zdrowy", "chory"))+
       labs(x="poziom cholesterolu",y="gestosc wystepowania")+
       theme(legend.position="top")+theme(legend.title = element_blank())
```

Prezentacja danych dotyczacych tetna - histogram Dane dotyczace tetna zdrowych i chorych pacjentow, zostaly zaprezentowane dzieki funkcji *geom_histogram*, ktora pozwolila zaprezentowac zaleznosc tego od stanu zdrowia pacjenta.

Korelacja

Przed analiza i wykorzystaniem posiadanego zbioru danych do przewidywania diagnozy, kluczowa kwestia jest ocena korelacji danych. W kolumnie *diag2* oraz w kolumnie *diag* umieszczone zostaly diagnozy. Korelacja innych danych z nimi, pozwala ocenic, jak wyniki badan i styl zycia wplywaja na ryzyko choroby. Stosowane funkcje *cor* oraz *ggcorrplot* pozwolily w przejrzysty sposob zaprezentowac korelacje miedzy danymi.

```
corr <- round(cor(result),1)
     ggcorrplot(corr)
```

Sekcja interfejsu

Zaprezentowane powyzej fragmenty programu pozwalaja wykonac dzialania serwera i obrobke danych. Za prezentacje danych w aplikacji shiny odpowiada kod programu *ui.R*. Skladaja sie na niego ponizsze elementy:

Layout

Za odpowiednie ulozenie elementow w aplikacji odpowiada *fluidPage* oraz *sidebarLayout*.

Wprowadzanie danych:

Dane stosowane do prezentacji i analizy wynikow pacjenta wprowadzane sa roznymi metodami. Ponizej przedstawie ich dzialanie:

```
sliderInput("jaki_wiek", "Wiek:",
                      min = 28, max = 77,
                      value = c(30,75)),
```

SliderInput

```
 selectInput("jaka_plec","Plec", choices = c("Kobieta"=0,"Mezczyzna"=1),multiple=FALSE)
```

selectInput

Dane do testu Dane to testu wprowadzane sa przy uzyciu *radioButtons* i *numericInput*.

```
radioButtons(
              "test_sex",
              "Plec",
              inline = TRUE,
              choiceNames = c("Mezczyzna","Kobieta"),
              choiceValues = c("1","0")
            ),
            numericInput(
              "test_age",
              "Wiek",
              min=0,
              value=40

            ),
```

Prezentacja danych

Prezentacja danych odbywa sie w panelu glownym przy uzyciu *tabsetPanel* oraz *tabPanel*, co porzadkuje widoki.

```
tabPanel("Cholesterol a stan zdrowia", plotOutput("plot_density")),
              tabPanel("Tetno a stan zdrowia",
                       plotOutput("plot_hist_chory"),
                       plotOutput("plot_hist_zdrowy")
                       ),



              tabPanel("Tetno - dla mezczyzn palacych i niepalacych", selectInput("czy_pali","Wybierz
                       plotOutput("distPlot"))
```

Diagnoza prognozowana po wypelnieniu testu wyswietlana jest jako tekst przy uzyciu *textOutput*.

```
textOutput("text"),
```

Skrypty analityczne - analiza i przewidywanie danych

Do analizy danych wykorzystywane byly *drzewa decyzyjne*. Drzewo tworzone bylo za pomoca funkcji *rpart* na danych treningowych *training*. Nastepnie testowane na danych testowych *test*. Wyniki nie byly satysfakcjonujace, lecz po przetestowaniu kilku metod, zdecydowalam sie na pozostanie przy tej funkcji.

W trakcie tworzenia projektu badalam takze rozwiazanie z wykorzystaniem funkcji *tree* oraz *ctree* po wyswietleniu schematu drzew i wynikow, moglam stwierdzic, ze stosowanie rpart dla mojego zbioru danych jest wystarczajace. Juz sam korelogram pozwolil stwierdzic, ze dane nie sa od siebie silnie zalezne i postawienie diagnozy moze byc utrudnione.

Podczas pracy nad dodatkowym zagadnieniem (drzewa decyzyjne, xgBoost) korzystalam z zasobow forum internetowych na temat programowania w R, prac na temat drzew decyzyjnych oraz dokumentacji. Zrezygnowalam z wykorzystania xgBoost w realizowanym projekcie ze wzgledu na ograniczony czas i zlozonosc programu.

```
  training <- heart_data[1:300,]
      test <- heart_data[301:617,]
      test$diag2 <- as.factor(test$diag2)
      training$diag2 <- as.factor(training$diag2)
      tree <- rpart(diag2~., training)
      predicted_Classes <- predict(tree, test, type="class")
```

Dla danych pobranych w ramach wykonywanego testu (our_test), przeprowadzilismy weryfikacje i prognoze diagnozy:

```
      our_test$age=input$test_age
      our_test$sex=as.integer(input$test_sex)
      our_test$cp=as.integer(input$test_cp)
      our_test$painloc=as.integer(input$test_painloc)
      our_test$trestbps=input$test_trestbps
      our_test$chol=input$test_chol
      our_test$fbs=as.integer(input$test_fbs)
      our_test$thalach=input$test_thalach
      our_test$exang=as.integer(input$test_exang)
      our_test$painexer=as.integer(input$test_painexer)
      our_test$relrest=as.integer(input$test_relrest)

 our_result <- predict(tree, our_test, type ="class")

      if(our_result[1]==0)
        diagnosis = "Wynik: Masz mniejsze ryzyko zachorowania"
      else if(our_result[1]==1)
        diagnosis="Wynik: masz duze ryzyko zachorowania"
      else diagnosis="Zapraszam do wypelnienia testu"

#diagnosis - wyswietlane jako tekst w aplikacji
```

Diagnoza stawiana jest poprawnie, jednak samo wyswietlanie jej nie dziala poprawnie.

Podsumowanie

Praca nad projektem podzielona byla na rozne etapy. Najtrudniejszym z nich byl, ku mojemu zaskoczeniu, etap wprowadzania danych, ktore byly zapisane w niezbyt wygodny sposob w pliku. Po wprowadzeniu danych, przeanalizowalam je i odeszlam nieco od schematu proponowanego na stronie - sama oceniajac, ktore dane sa w moim programie potrzebne i uzyteczne.

Nastepnym trudnym etapem byl wybor danych do prezentacji i analizy. Sam korelogram pokazuje, ze dane nie sa od siebie mocno zalezne, wiec histogram czy inne wizualizacje nie do konca wskazywaly na to, jakie czynniki wplywaja na chorobe.

Ostatni etap - etap klasyfikacji i wykorzystanie drzew decyzyjnych, pozwolil mi poznac metody i rozszerzyc wiedze, ktora jest bardzo dobrym wstepem do dalszego rozwoju w pisaniu skryptow analitycznych.