hm4

Robert Benke

29 marca 2019

Homework IV

Zadanie 1.1

Wybrane zmienne:

- time_from_rel_to_cohab - czas pomiedzy poznaniem a rozpoczęciem relacji
- hcm2017q24_college - poznali sie na uniwersytecie

- hcm2017q24_bar_restaurant - poznali się w barze/restauracji/itp.
- partner_yrsed - liczba lat jaką pertner spędził na edukcji

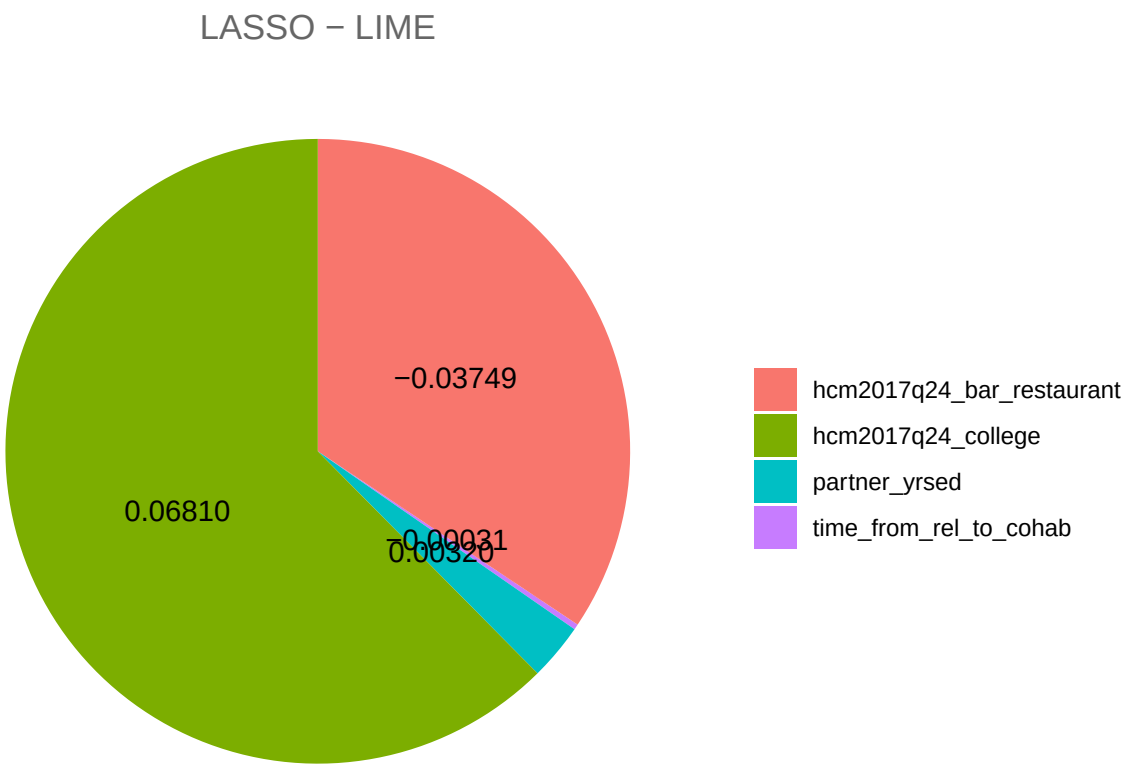
Zadanie 1.2 - random forest

Zadanie 4

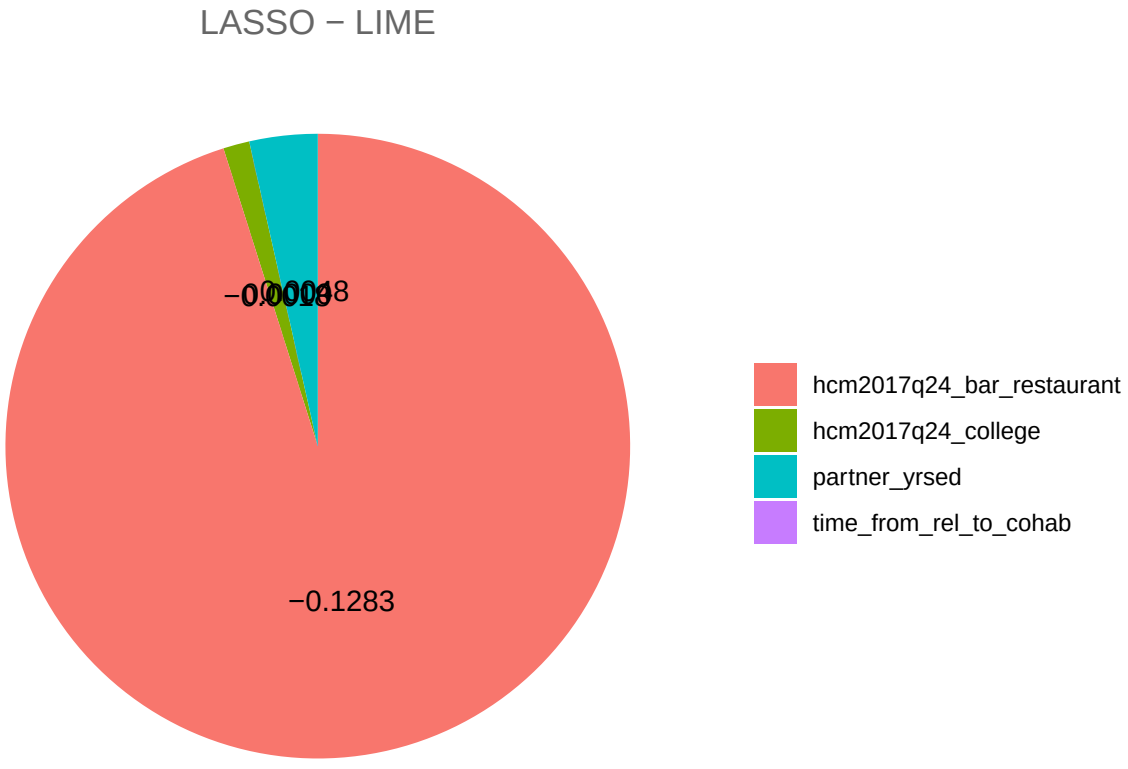
```
closest_in_dataset <- function(vec,neighbourhood){
  vec <- unique(vec)
  sapply(neighbourhood,FUN= function(x){
    vec[which.min(abs(vec - x))]
  })
}

LIME <- function(model ,lrn ,data,obs,target = "S1",sigma = 0.5,n_obs = 100,lambda= 0.01,expo){
  #data preprocesing
  data_lime <- obs[rep(1,n_obs),]
  data_lime <- select(data_lime,-target) %>% apply(MARGIN = 2,FUN = as.numeric) %>% as.data.frame()
  data <- select(data,-target) %>% apply(MARGIN = 2,FUN = as.numeric) %>% as.data.frame()
  names <- apply(data, MARGIN = 2, FUN = function(x) length(unique(x)))
  obs <- select(obs,-target)

  # spliting the attributes (discrite, continous)
  names_continous <- names(names[names>sqrt(nrow(data))])
  names_discrete <- names(obs)[!(names(obs) %in% names_continous)]
  names_continous_NonNeg <-names_continous[!names_continous %in% names(data[data<0])]
  names_continous_With_Neg <- names(names_continous)[!(names(names_continous) %in% names_continous_NonN
  #generate new values, with mean from the new observation, and with sigma
  # from all data
  lapply(names_discrete, FUN=function(name){
    sd <- data[,name] %>% sd()
    neighbourhood <- rnorm(n_obs,as.numeric(obs[name]),sigma*sd)
    vec <- data[,name]
    data_lime <<- data_lime %>% mutate(!!sym(name):= closest_in_dataset(vec,neighbourhood))
    })
  #gamma is non-negative
```

```
  lapply(names_continous_NonNeg, FUN=function(name){
    sd <- data[,name] %>% sd()
    vec = rgamma(n_obs,shape = (obs[,name]+0.1)^2/(sigma*sd) , scale = (sigma*sd)/(obs[,name]+0.1))
    data_lime <<- mutate(data_lime, !!sym(name) := vec)
    })

  lapply(names_continous_With_Neg, FUN=function(name){
    vec <- rnorm(n_obs,obs[,name],sigma*sd(data[,name]))
    data_lime <<- mutate(data_lime, !!sym(name) := vec)
  })


  prediction <- predictLearner(lrn,model,data_lime)[,1]
  # calculate weights as 1/(euclidian distance from orginal)
  weights <- apply(data_lime, MARGIN = 1, FUN = function(x) norm(as.numeric(x)-as.numeric(obs),type="2")
  weights <-  1/(weights+0.01)^expo
  weights <- weights/sum(weights)
  lasso <- glmnet(as.matrix(data_lime) , family = "gaussian", weights = weights,
                  prediction, alpha = 1, lambda = lambda)
# plot results
  results <-  lasso$beta %>% as.matrix() %>%  as.data.frame() %>%
    cbind(names(data_lime)) %>% setNames(c("importance","variable")) %>%
    arrange(desc(abs(importance)))

  ggplot(results,aes(x="",y=importance,fill=variable))+ geom_bar(width = 1, stat = "identity") +
    coord_polar("y", start=0) + labs(x = NULL, y = NULL, fill = NULL, title = "LASSO - LIME") +
    geom_text(aes(label = (format(importance,digits=2 ))), position = position_stack(vjust = 0.5))+
    theme_classic() + theme(axis.line = element_blank(),
          axis.text = element_blank(),
          axis.ticks = element_blank(),
          plot.title = element_text(hjust = 0.5, color = "#666666"))
  }
```

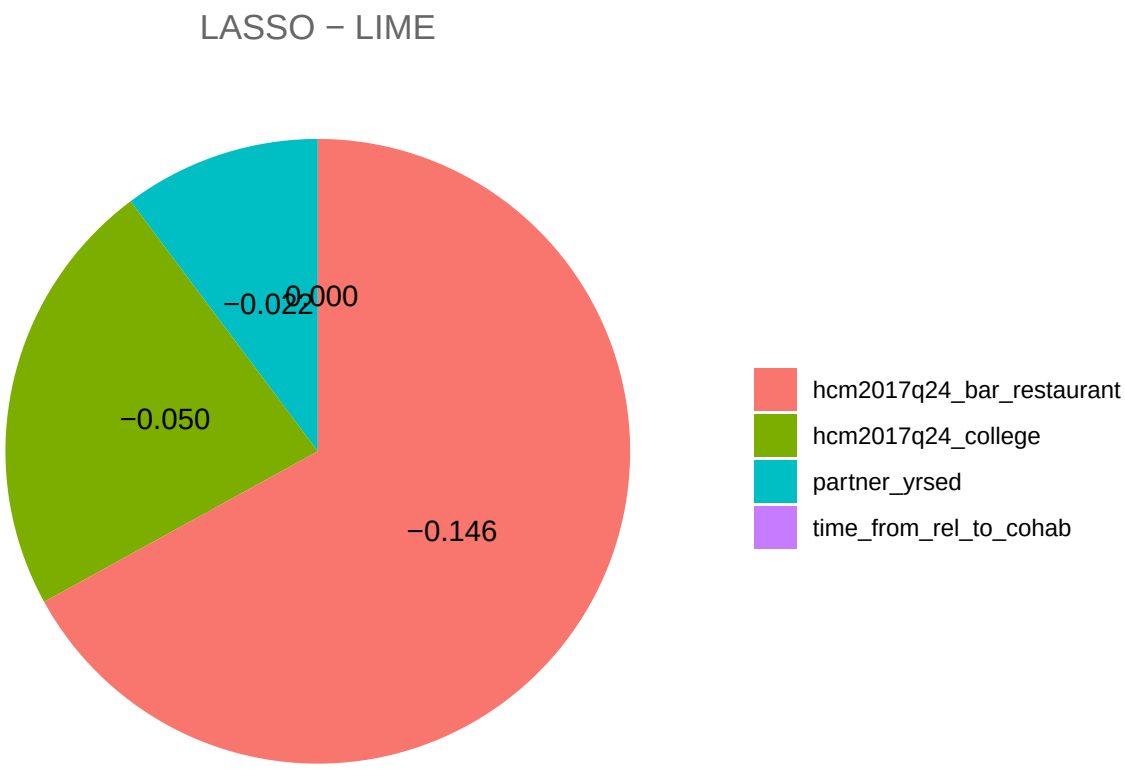
```
LIME(model ,lrn ,data,obs = data[123,],target = "S1",sigma = 3,n_obs = 10000,lambda= 0.001,expo=0.2)
```



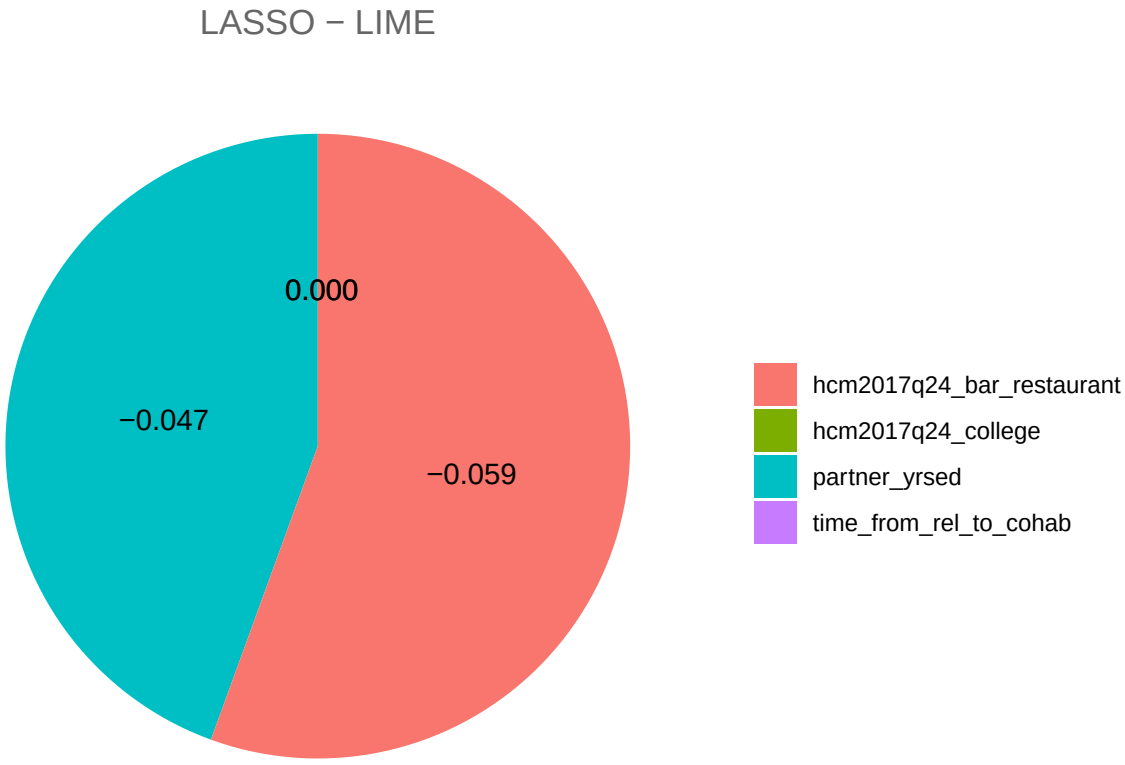

```
LIME(model ,lrn ,data,obs = data[123,],target = "S1",sigma = 3,n_obs = 10000,lambda= 0.001,expo=0.6)
```




```
LIME(model ,lrn ,data,obs = data[123,],target = "S1",sigma = 1,n_obs = 10000,lambda= 0.001,expo=1)
```




```
LIME(model ,lrn ,data,obs = data[123,],target = "S1",sigma = 0.8,n_obs = 10000,lambda= 0.001,expo=1.5)
```




```
library(iml)
data2 <- apply(data,MARGIN = 2, as.numeric) %>% as.data.frame()
X <- data2[which(names(data) != "S1")]
predictor = Predictor$new(model, data = X, y = data2$S1)
```

```
lime.explain = LocalModel$new(predictor, x.interest = X[123,], k =4)
```

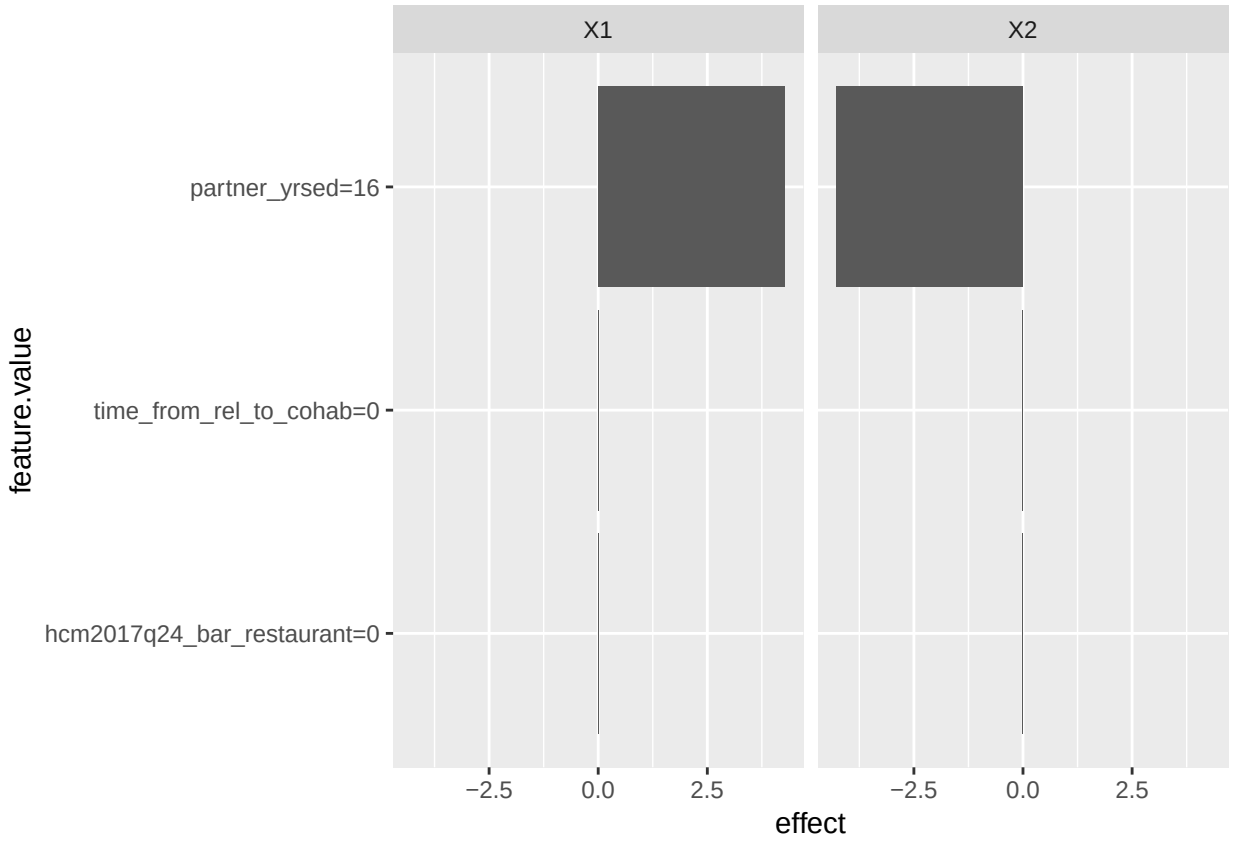
```
## Loading required package: gower
```

```
## Warning in private$aggregate(): Had to choose a smaller k
```

```
lime.explain$results
```

```
##         beta x.recoded    effect x.original                    feature
## 1 -0.0798626         0  0.000000          0    time_from_rel_to_cohab
## 2 -0.4784807         0  0.000000          0 hcm2017q24_bar_restaurant
## 3  0.2670028        16  4.272045         16             partner_yrsed
## 4  0.0798626         0  0.000000          0    time_from_rel_to_cohab
## 5  0.4784807         0  0.000000          0 hcm2017q24_bar_restaurant
## 6 -0.2670028        16 -4.272045         16             partner_yrsed
##                  feature.value .class
## 1    time_from_rel_to_cohab=0     X1
## 2 hcm2017q24_bar_restaurant=0     X1
## 3            partner_yrsed=16     X1
## 4    time_from_rel_to_cohab=0     X2
## 5 hcm2017q24_bar_restaurant=0     X2
## 6            partner_yrsed=16     X2
```

```
plot(lime.explain)
```



Wnioski:

- w porównaniu do poprzeniego ćwiczenia (wartość oczekiwana w różnych kolejnościach) otrzymałem inne wyniki niż poprzednio. Zmienne wyraźnie istotne wcześniej nie pojawiają się, lub pojawiają się jako mało znaczące obecnie.
- dużym problemem tej metody jest ustalenie wag. W przypadku gdy w zbiorze atrybutów znajdują się zarówno zmienne ciągłe jak i dyskretne, łatwo jest faworyzować któreś z nich. W szczególności jeśli wagi zbyt szybko spadają do zera to zmienne dyskretne nie mają szans bo posiadają minimalną odległość od nowej obserwacji. Zmienne ciągłe mogą za to być dowolnie blisko.
- jeśli tylko wagi maleją dostatecznie wolno, przestajemy mieć do czynienia z lokalnym wyjaśnieniem. Jednak nawet godząc się na badanie większego obszaru nie udało mi się uzyskać wyników podobnych do uzyskanych wcześniej. W skrajnym przypadku, gdy wagi byłyby równe, a dane losowane z rozkładu empirycznego, wartości współczynników przy zmiennych dychotomicznych powinny być zbliżone do średnich wartości z metody 'warunkowania zmiennych'.
- na koniec sprawdziłem jeszcze funkcje lime z pakietu iml. Metoda dała inne odpowiedzi niż moja. Dokumentacja tej metody wskazuje na szereg różnić w podejściu, w szczególności w dobieraniu wag.
- metoda ta wydaje się użyteczna, ale tylko wtedy, gdy wszystkie zmienne są tego samego typu (dla zdjęć - segmenty obrazu, dla tekstu - słowa, dla danych tabelarycznych - wszystkie ciągłe (znormalizowane), ALBO dychotomiczne)
- po chwili poszukiwania odpowiednich parametrów udało mi się uzyskać te same wyniki co funkcji z 'iml'. (plot poniżej)

```
LIME(model ,lrn ,data,obs = data[123,],target = "S1",sigma = 0.5,n_obs = 10000,lambda= 0.001,expo=2)
```