Support Vector Machine: Uma aplicação R

Kim Silva

20/05/2022

Vamos aplicar o modelo SVM no conjunto de dados sobre diabetes disponível no `R` com o comando `data("diabetes392")` do pacote `HTLR`. Este conjunto de dados é originalmente do Instituto Nacional de Diabetes e Doenças Digestivas e Renais. O objetivo do conjunto de dados é prever diagnosticamente se um paciente tem ou não diabetes, com base em certas medidas de diagnóstico incluídas no conjunto de dados. Várias restrições foram colocadas na seleção dessas instâncias de um banco de dados maior. Em particular, todos os pacientes aqui são mulheres com pelo menos 21 anos de ascendência indígena Pima. Diferente da versão original do UCI, o conjunto de dados foi pré-processado para que as linhas com valores ausentes sejam removidas e os recursos sejam dimensionados.

Para mais informações sobre a base de dados consultar no link: https://search.r-project.org/CRAN/refmans/HTLR/html/diabetes392.html

```
#install.packages('HTLR')
library(HTLR)
```

```
## Warning: package 'HTLR' was built under R version 4.1.3
```

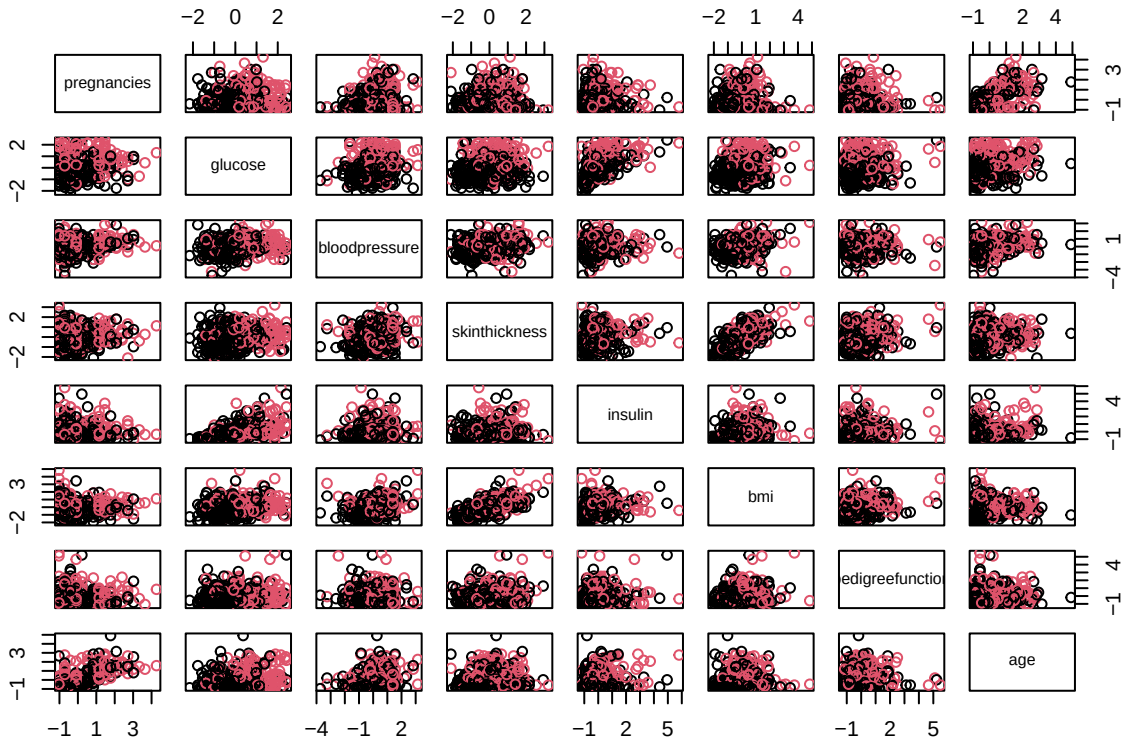
```
data("diabetes392")
dados = as.data.frame(cbind(diabetes392$X,diabetes392$y))
dados$V9 <- as.factor(dados$V9)
summary(dados)
```

```
##   pregnancies         glucose        bloodpressure      skinthickness
##  Min.   :-1.0279   Min.   :-2.1590   Min.   :-3.73423   Min.   :-2.10579
##  1st Qu.:-0.7165   1st Qu.:-0.7656   1st Qu.:-0.69328   1st Qu.:-0.77454
##  Median :-0.4051   Median :-0.1175   Median :-0.05308   Median :-0.01383
##  Mean   : 0.0000   Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.00000
##  3rd Qu.: 0.5290   3rd Qu.: 0.6601   3rd Qu.: 0.58712   3rd Qu.: 0.74689
##  Max.   : 4.2657   Max.   : 2.4423   Max.   : 3.14792   Max.   : 3.21921
##     insulin             bmi           pedigreefunction       age
##  Min.   :-1.1953   Min.   :-2.11823   Min.   :-1.2679   Min.   :-0.9671
##  1st Qu.:-0.6673   1st Qu.:-0.66683   1st Qu.:-0.7332   1st Qu.:-0.7710
##  Median :-0.2571   Median : 0.01619   Median :-0.2129   Median :-0.3789
##  Mean   : 0.0000   Mean   : 0.00000   Mean   : 0.0000   Mean   : 0.0000
##  3rd Qu.: 0.2856   3rd Qu.: 0.57114   3rd Qu.: 0.4746   3rd Qu.: 0.5034
##  Max.   : 5.8056   Max.   : 4.83999   Max.   : 5.4906   Max.   : 4.9148
##  V9
##  1:262
##  2:130
##
##
```

##
##

Como podemos notar há um desbalanceamento na variável de interesse. A base de dados, conta com 130 pacientes diabéticas e 262 pacientes não diabéticas. Desse modo, o indicado seria utilizar algum método para correção do desbalanceamento das classes.

```
library(graphics)
pairs(dados[,-9], col = dados$V9)
```



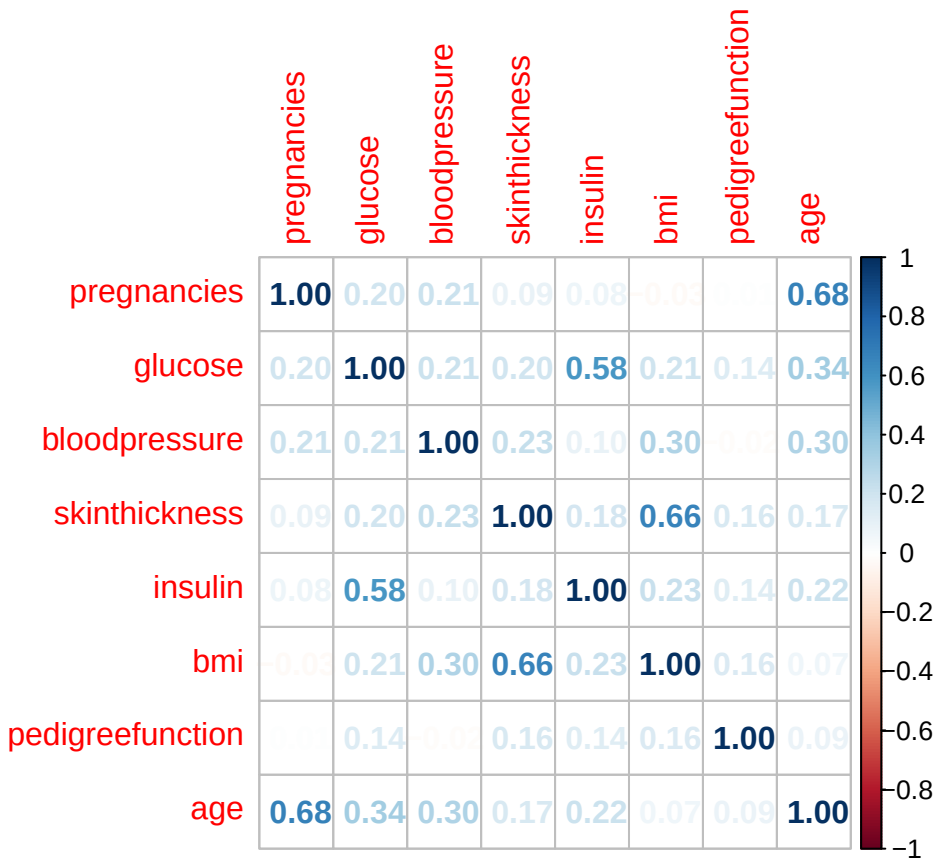
Vemos na imagem acima, o gráfico de dispersão. As observações das pacientes, estão coloridas de acordo com a condição diabética ou não diabética. Podemos perceber, de acordo com o comportamento dos dados, que não conseguimos ver uma distinção clara para classificarmos diabéticas e não diabéticas. Nesse contexto, o modelo SVM pode ver uma boa opção de classificador.

```
library(corrplot)
```

```
## Warning: package 'corrplot' was built under R version 4.1.1
```

```
## corrplot 0.90 loaded
```

```
corrplot(cor(dados[,-9]), method = "number")
```



A correlação entre as variáveis `age` e `pregnancies`, `skinthicknesse` `bmi` apresentam correlação maior que 60%, o que pode causar problemas de autocorrelação Desse modo, iremos construir os modelos SVM e Regressão Logística com todas as variáveis e sem as variáveis `age` e `skinthickness` pois apresenta alta correlação com as variáveis `pregnancies` e `bmi`.

Support Vector Machine

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.1.1
```

```
tune.out = tune(svm ,V9~.,data=dados,
                ranges=list(kernel=c("linear", "polynomial",
                                     "radial", "sigmoid")))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  kernel
```

```
##  linear
##
## - best performance: 0.2269231
##
## - Detailed performance results:
##       kernel     error dispersion
## 1     linear 0.2269231 0.05790559
## 2 polynomial 0.2475641 0.08050186
## 3     radial 0.2499359 0.06781315
## 4    sigmoid 0.2805769 0.06813931
```

```
tune.out=tune(svm ,V9~.,data=dados, kernel = 'linear',
              ranges=list(cost=c(0.001, 0.01, 0.1, 1,5,10,100)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##     5
##
## - best performance: 0.2195513
##
## - Detailed performance results:
##    cost     error dispersion
## 1 1e-03 0.3319872 0.07084772
## 2 1e-02 0.2196795 0.07139825
## 3 1e-01 0.2271795 0.05770671
## 4 1e+00 0.2220513 0.05580172
## 5 5e+00 0.2195513 0.05863214
## 6 1e+01 0.2195513 0.05863214
## 7 1e+02 0.2221154 0.05722927
```

A função `tune` realiza o método 10-fold cross validation e como resultado o melhor kernel foi o `linear`. O mesmo método realizado para escolha do parâmetro `cost` resultou no melhor `cost` igual a 5.

Iremos utilizar o método 10-fold cross validation para mensurar o desempenho dos modelos.

```
# 10-fold cross validation:
ACC=numeric(10)
MCC=numeric(10)
SEN=numeric(10)
VPP=numeric(10)
F1=numeric(10)
BS=numeric(10)

set.seed(42553)
df = dados[sample(1:nrow(dados)),]
a = c(1,round(0.1*length(dados$V9)), round(0.2*length(dados$V9)),
      round(0.3*length(dados$V9)), round(0.4*length(dados$V9)),
```

```
      round(0.5*length(dados$V9)), round(0.6*length(dados$V9)),
      round(0.7*length(dados$V9)), round(0.8*length(dados$V9)),
      round(0.9*length(dados$V9)), length(dados$V9))
for(i in 1:10){
    # Divisao em treinamento (70%) e teste (30%):
    dados.trein= df[a[i]:a[i+1], ]
    dados.test=df[-(a[i]:a[i+1]),]
    # SVM:
    mod = svm(V9~.,data = dados.trein, kernel = "linear", cost = 5)
    # Desempenho preditivo (teste):
    pred = predict(mod, dados.test)
    mc=table(predict = pred, truth = dados.test$V9)
    VN=mc[1,1]; FP=mc[2,1]; FN=mc[1,2]; VP=mc[2,2]
    ACC[i]=(VP+VN)/(VP+VN+FN+FP)
    MCC[i]=(VP*VN-FP*FN)/(sqrt(VP+FP)*sqrt(VP+FN)*sqrt(VN+FP)*sqrt(VN+FN))
    SEN[i]=VP/(VP+FN)
    VPP[i]=VP/(VP+FP)
    F1[i]=2*VPP[i]*SEN[i]/(VPP[i]+SEN[i])
}
rsvm = c(mean(ACC), mean(MCC), mean(SEN), mean(VPP), mean(F1))
rsvm
```

```
## [1] 0.7254747 0.3589180 0.5105625 0.5968940 0.5347491
```

```
# 10-fold cross validation:
ACC=numeric(10)
MCC=numeric(10)
SEN=numeric(10)
VPP=numeric(10)
F1=numeric(10)
BS=numeric(10)

for(i in 1:10){
    # Divisao em treinamento (70%) e teste (30%):
    dados.trein = df[a[i]:a[i+1],-c(4,8) ]
    dados.test = df[-(a[i]:a[i+1]),-c(4,8)]
    # SVM:
    mod = svm(V9~.,data = dados.trein, kernel = "linear", cost = 5)
    # Desempenho preditivo (teste):
    pred = predict(mod, dados.test)
    mc=table(predict = pred, truth = dados.test$V9)
    VN=mc[1,1]; FP=mc[2,1]; FN=mc[1,2]; VP=mc[2,2]
    ACC[i]=(VP+VN)/(VP+VN+FN+FP)
    MCC[i]=(VP*VN-FP*FN)/(sqrt(VP+FP)*sqrt(VP+FN)*sqrt(VN+FP)*sqrt(VN+FN))
    SEN[i]=VP/(VP+FN)
    VPP[i]=VP/(VP+FP)
    F1[i]=2*VPP[i]*SEN[i]/(VPP[i]+SEN[i])
}
rsvm1 = c(mean(ACC), mean(MCC), mean(SEN), mean(VPP), mean(F1))
rsvm1
```

```
## [1] 0.7388311 0.3845624 0.4740185 0.6605530 0.5293173
```

```
medidas <- data.frame(Medidas = c('ACC', 'MCC', 'SEN', 'VPP', 'F1'),
SVM = rsvm, SVM_2 = rsvm1)

library(knitr)
kable(medidas)
```

Medidas	SVM	SVM_2
ACC	0.7254747	0.7388311
MCC	0.3589180	0.3845624
SEN	0.5105625	0.4740185
VPP	0.5968940	0.6605530
F1	0.5347491	0.5293173

A tabela acima mostra o desempenho dos modelos. O modelo `SVM_2`, que remove as variáveis `age` e `skinthickness`, foi o modelo com maiores Acurácia (ACC), Coeficiente de Correlação de Matthews (MCC) e Valor Preditivo Positivo (VPP). Por outro lado, o modelo ajustado com todas as variáveis tem uma maior sensibilidade (SEN).