Maquinas de Soporte Vectorial

Maisy Samai Vázquez Sánchez

2022-05-29

Capítulo I

Introducción

Las máquinas de soporte vectorial sirven para la clasificación óptima entre clases. La función de las máquinas de soporte vectorial en primera instancia es mapear los datos en un espacio mayor a los que originalmente pertenecen, seguido de esto busca un hyperplano que logre separar y maximizar el margen entre las distintas clases. El hyperplano más óptimo se obtiene mediante los kernels. En las máquinas de soporte vectorial se presentan dos tipos de casos.

A) Linealmente separable: En este caso cada dato de entrenamiento pertenece a una de las dos clases, la mayor parte de las veces suele ser difícil encontrar un hyperplano para el uso práctico, es por esto que se propone mapear el espacio de entrada a dimensiones mayores y así encontrar el hyperplano más óptimo.

B) Linealmente no separable: Para los casos no separables se pueden introducir variables no negativas. El parámetro C (constante), sirve como un parámetro de regularización y puede ayudar al balance de la maximización del margen y la violación de la clasificación.

Truco Kernel.

Cuando Φ no es conocido se aplica la función de kernel, la cual calcula el producto de punto de los datos datos de entrada en el espacio de características.

Descripción de la matriz de datos

Los datos provienen del Instituto Nacional de Diabetes y Enfermedades Digestivas y Renales. Los datos sirven para predecir si un paciente padece o no diabetes, la mayor parte de los pacientes son mujeres de al menos 21 años y de descendecia indígena pima.

La base de datos tiene variables predictoras médicas y una variable latente. La variables clasificadas como predictoras contienen datos de edad, nivel de insulina, su IMC y numero de embarazos que ha tenido la paciente.

Objetivo: Encontrar un modelo que logre clasificar y discriminar de manera optima a individuos de dos condiciones.

Se cargan las librerias necesarias para el análisis

```
library(ggplot2)
library(ggpubr)
library(MVN)
library(e1071)
library(MASS)
library(caTools)
library(scatterplot3d)
```

Se cargan los datos desde un archivo csv

```
datos=read.csv("C:\\Users\\nitzi\\OneDrive\\One Drive 2\\OneDrive\\Documentos\\SAMAI\\diabetes.csv")
```

Exploración de la matriz

1.Dimensión

```
dim(datos)
```

```
## [1] 768   9
```

La base de datos cuenta con **768** observaciones y **9** variables.

2.Nombre de las variables

```
colnames(datos)
```

```
## [1] "Pregnancies"              "Glucose"
## [3] "BloodPressure"            "SkinThickness"
## [5] "Insulin"                  "BMI"
## [7] "DiabetesPedigreeFunction" "Age"
## [9] "Outcome"
```

```
str(datos)
```

```
## 'data.frame':    768 obs. of  9 variables:
##  $ Pregnancies             : int  6 1 8 1 0 5 3 10 2 8 ...
##  $ Glucose                 : int  148 85 183 89 137 116 78 115 197 125 ...
##  $ BloodPressure           : int  72 66 64 66 40 74 50 0 70 96 ...
##  $ SkinThickness           : int  35 29 0 23 35 0 32 0 45 0 ...
##  $ Insulin                 : int  0 0 0 94 168 0 88 0 543 0 ...
##  $ BMI                     : num  33.6 26.6 23.3 28.1 43.1 25.6 31 35.3 30.5 0 ...
##  $ DiabetesPedigreeFunction: num  0.627 0.351 0.672 0.167 2.288 ...
##  $ Age                     : int  50 31 32 21 33 30 26 29 53 54 ...
##  $ Outcome                 : int  1 0 1 0 1 0 1 0 1 1 ...
```

Como se mencionó anteriormente existen 9 variables en la base, de las cuales podemos ver que todas son numéricas, sin embargo, dos de las variables son continuas.

Variables

- ***Pregnancies:*** Número de veces que a estado embarazada.
- ***Glucose:*** Concentración de glucosa plasmática a las 2 horas en una prueba de tolerancia oral a la glucosa.
- ***BloodPressure:*** Presión arterial diastólica (mm Hg).
- ***SkinThickness:*** Grosor del pliegue cutáneo del tríceps (mm).
- ***Insulin:*** Insulina sérica de 2 horas (mu U/ml).
- ***BMI:*** Índice de masa corporal (peso en kg/(altura en m)ˆ2).
- ***DiabetesPedigreeFunction:*** Función de pedigrí de diabetes.
- ***Age:*** Años de edad
- ***Outcome:*** Variable de clase (0 o 1) 268 de 768 son 1, los demás son 0.

3.Presencia de Na's

```
anyNA(datos)
```

```
## [1] FALSE
```

No hay valores faltantes en la base de datos.

Capítulo II

Tratamiento de la matriz

1.Asignación del data frame

```
datos1=datos
```

2.Convertir los valores de 0 a 1 en etiquetas

```
datos1[datos1$Outcome==0,9]="Sin diabetes"
datos1[datos1$Outcome==1,9]="Diabetes"
datos1$Outcome=as.factor(datos1$Outcome)
summary(datos1)
```

```
##   Pregnancies        Glucose      BloodPressure    SkinThickness
##  Min.   : 0.000   Min.   :  0.0   Min.   :  0.00   Min.   : 0.00
##  1st Qu.: 1.000   1st Qu.: 99.0   1st Qu.: 62.00   1st Qu.: 0.00
##  Median : 3.000   Median :117.0   Median : 72.00   Median :23.00
##  Mean   : 3.845   Mean   :120.9   Mean   : 69.11   Mean   :20.54
##  3rd Qu.: 6.000   3rd Qu.:140.2   3rd Qu.: 80.00   3rd Qu.:32.00
##  Max.   :17.000   Max.   :199.0   Max.   :122.00   Max.   :99.00
##     Insulin           BMI        DiabetesPedigreeFunction      Age
##  Min.   :  0.0   Min.   : 0.00   Min.   :0.0780           Min.   :21.00
```

```
##  1st Qu.:  0.0   1st Qu.:27.30   1st Qu.:0.2437          1st Qu.:24.00
##  Median : 30.5   Median :32.00   Median :0.3725          Median :29.00
##  Mean   : 79.8   Mean   :31.99   Mean   :0.4719          Mean   :33.24
##  3rd Qu.:127.2   3rd Qu.:36.60   3rd Qu.:0.6262          3rd Qu.:41.00
##  Max.   :846.0   Max.   :67.10   Max.   :2.4200          Max.   :81.00
##          Outcome
##  Diabetes    :268
##  Sin diabetes:500
##
##
##
##
```

Se puede observar que se cuenta con **268** sujetos **con diabetes** y **500 sin diabetes**.

3.Fijar semilla

```
set.seed(1234)
```

Utilizó una semilla con el fin de obtener los mismos resultados sin importar las veces que se ejecute el código.

Generación los datos de prueba y de entrenamiento

```
split <- sample.split(datos1$Outcome , SplitRatio = 0.80)
entrenamiento <- subset(datos1, split == TRUE)
prueba <- subset(datos1, split == FALSE)
```

Se toma el 80% para generar el modelo y con el 20% restante se prueba el mismo.

4.Proporción de los datos de entrenamiento

```
table(entrenamiento$Outcome)
```

```
##
##     Diabetes Sin diabetes
##          214          400
```

5.Proporción de los datos de prueba

```
table(prueba$Outcome)
```

```
##
##     Diabetes Sin diabetes
##           54          100
```

Capítulo III

Metodología de análisis

Como se mencionó anteriormente se desea encontrar un modelo que sea capaz de clasificar correctamente a los sujetos de 2 condiciones (Sin diabetes y Diabetes).

Para ello se empleó un modelo del tipo *"Maquina de soporte vectorial"* para llevar a cabo esta tarea se usará la misma depuración de los datos y las mismas muestras que se usan para el análisis discriminante lineal con la finalidad de comparar los resultados obtenidos.

Se realizó el ajuste del modelo con un kernel radial (Gaussiano), kernel lineal, polynomial y sigmoid a un costo computacional de 10 y se obtuvieron diferentes posibles vectores de soporte para clasificación en cada modelo.

Posteriormente se probará que tan bueno es el modelo discriminando a los sujetos que lo constituyen.

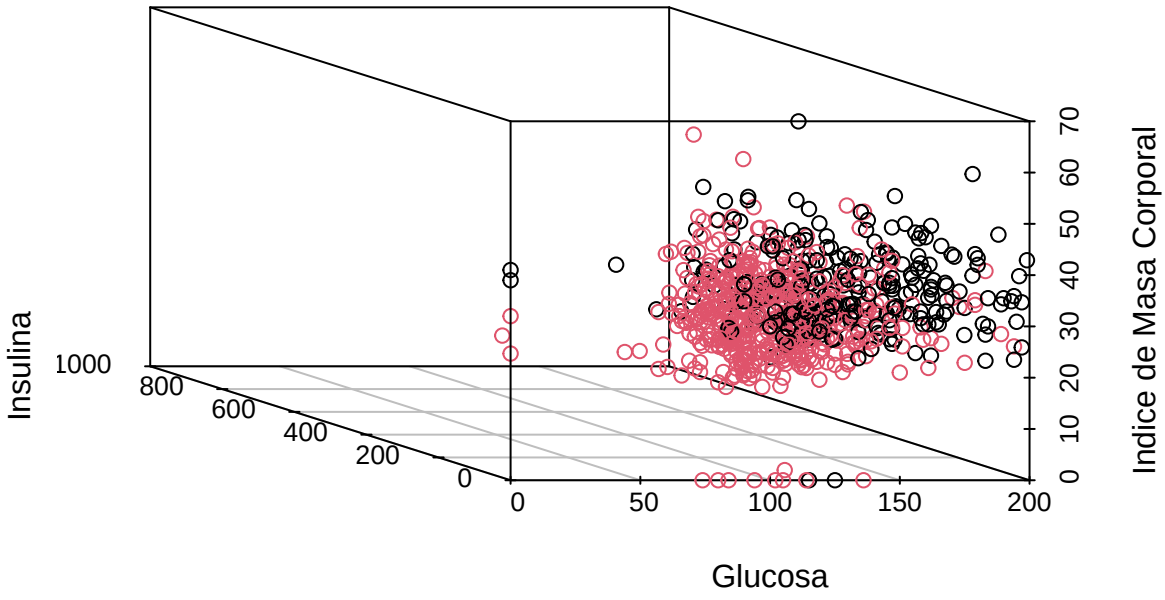
Capítulo IV

Resultados

Gráfico para ver la relación entre las variables

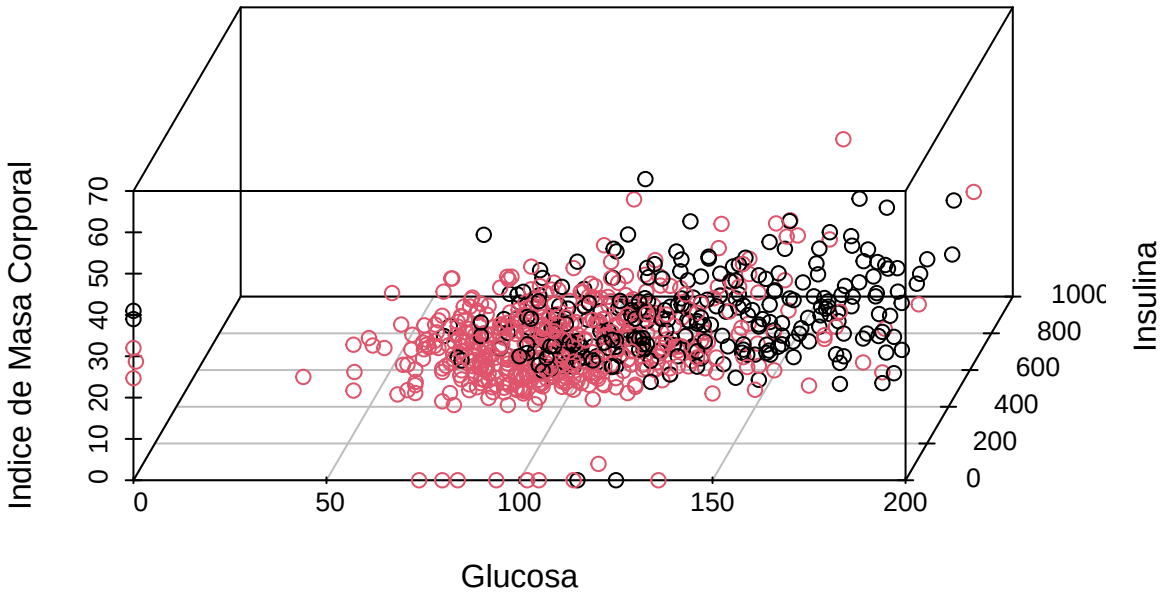
Ángulo de 500

```
scatterplot3d(x=datos1$Glucose,y=datos1$Insulin,
              z=datos1$BMI,
              color =as.numeric(datos1$Outcome),
              xlab="Glucosa",
              ylab = "Insulina",
              zlab = "Indice de Masa Corporal",
              angle = 500)
```



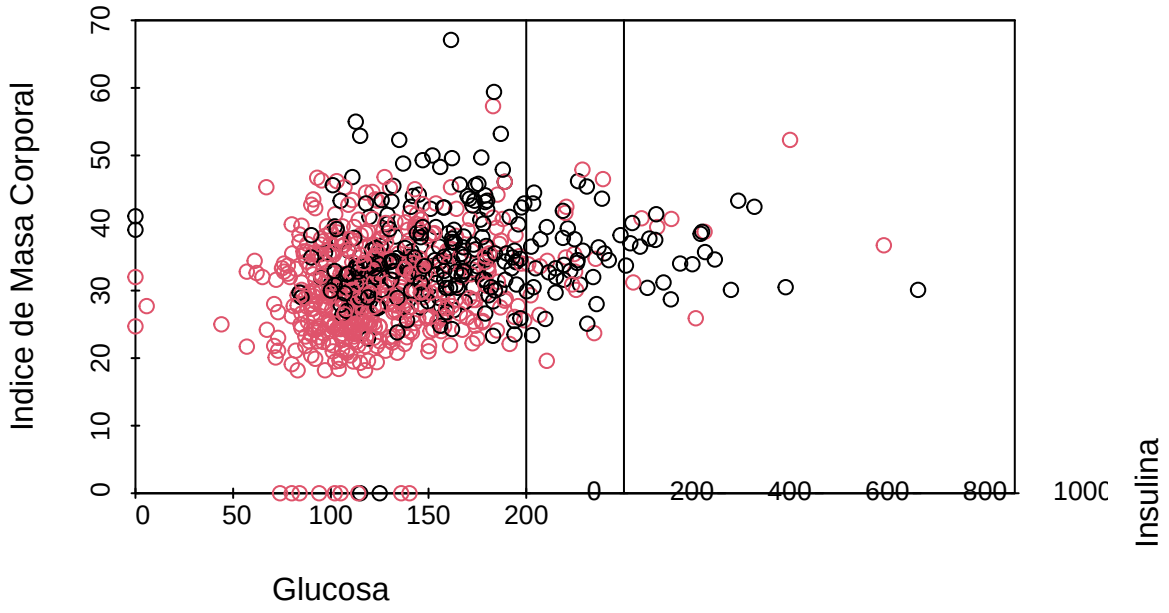
Ángulo de 800

```
scatterplot3d(x=datos1$Glucose,y=datos1$Insulin,
              z=datos1$BMI,
              color =as.numeric(datos1$Outcome),
              xlab="Glucosa",
              ylab = "Insulina",
              zlab = "Indice de Masa Corporal",
              angle = 800)
```



Ángulo de 360

```
scatterplot3d(x=datos1$Glucose,y=datos1$Insulin,
              z=datos1$BMI,
              color =as.numeric(datos1$Outcome),
              xlab="Glucosa",
              ylab = "Insulina",
              zlab = "Indice de Masa Corporal",
              angle = 360)
```



Nota: Se puede jugar con el número de los ángulos para apreciarlo desde diferentes puntos.

Se puede apreciar como los valores están traslapados y no se pueden separar, se observa dos conjuntos de datos (rosa y negro) bastante cercanos, algunos valores sobre salen del conjunto que podrían tener un comportamiento diferente. Desde este gráfico exploratorio especulamos que los valores no podrían clasificarse y discriminarse de manera adecuada.

Armar y hacer ajustes a la máquina de soporte vectorial

Kernel Lineal

1.Se aplica el kernel **lineal**

```
MSV = svm(Outcome~.,data = entrenamiento,kernel = "linear", cost = 10,scale = F)
print(MSV)
```

```
##
## Call:
## svm(formula = Outcome ~ ., data = entrenamiento, kernel = "linear",
##     cost = 10, scale = F)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  linear
```

```
##        cost:  10
##
## Number of Support Vectors:  325
```

Con el kernel lineal se obtuvieron *325* posibles vectores.

2.Se prueba el poder de discriminación.

```
prediclas = predict(object = MSV, newdata = entrenamiento[,-9])
discriminacion = table(entrenamiento$Outcome ,
                       prediclas ,
                       dnn=c("Clase real","Clase predicha"))
discriminacion
```

```
##               Clase predicha
## Clase real     Diabetes Sin diabetes
##   Diabetes          136           78
##   Sin diabetes       65          335
```

Se puede observar que hay 143 sujetos que el modelo NO logra discriminar correctamente. Se muestra que hay 78 personas que padecen diabetes y el modelo los discrimina de la forma contraria, por otro lado hay 65 sin padecer diabetes y el modelo discrimina como si los sujetos SI padecieran diabetes.

ERROR DE DISCRIMINACIÓN

```
(discriminacion[1,2] + discriminacion[2,1])/sum(discriminacion)*100
```

```
## [1] 23.2899
```

El modelo se equivoca al discriminar en un ***23%***

3.Se prueba el poder de clasificación

```
preds = predict(object = MSV, newdata = prueba[,-9])
clasi = table(prueba$Outcome,
              preds,
              dnn=c("Clase real","Clase predicha"))
clasi
```

```
##               Clase predicha
## Clase real     Diabetes Sin diabetes
##   Diabetes           31           23
##   Sin diabetes        8           92
```

Al clasificar el modelo clasifica a 23 personas de manera errónea puesto que si padecen diabetes y los manda al grupo de "sin diabetes", de igual manera hay 8 sujetos sin diabetes que que se clasificaron en "diabetes".

ERROR DE CLASIFICACIÓN

```
(clasi[1,2] + clasi[2,1])/sum(clasi)*100
```

```
## [1] 20.12987
```

Clasifica mal en un ***20%***

Kernel Gaussiano

1.Se aplica el kernel **gaussiano**

```
MSV = svm(Outcome~.,data = entrenamiento,kernel = "radial", cost = 10,scale = F)
print(MSV)
```

```
##
## Call:
## svm(formula = Outcome ~ ., data = entrenamiento, kernel = "radial",
##     cost = 10, scale = F)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  radial
##        cost:  10
##
## Number of Support Vectors:  614
```

Se obtuvieron *614* posibles vectores

2.Se prueba el poder de discriminación

```
prediclas = predict(object = MSV, newdata = entrenamiento[,-9])
discriminacion = table(entrenamiento$Outcome ,
                       prediclas ,
                       dnn=c("Clase real","Clase predicha"))
discriminacion
```

```
##               Clase predicha
## Clase real     Diabetes Sin diabetes
##   Diabetes          214            0
##   Sin diabetes        0          400
```

El kernel gaussiano discrimina de manera perfecta, hay 214 personas que padecen diabetes y 400 que no la padecen.

ERROR DE DISCRIMINACIÓN

```
(discriminacion[1,2] +discriminacion[2,1])/sum(discriminacion)*100
```

```
## [1] 0
```

Podemos observar que con el Kernel gaussiano el modelo no se equivoca al discriminar a los individuos.

3.Se prueba el poder de clasificación

```
preds = predict(object = MSV, newdata = prueba[,-9])
clasi = table(prueba$Outcome,
              preds,
              dnn=c("Clase real","Clase predicha"))
clasi
```

```
##               Clase predicha
## Clase real     Diabetes Sin diabetes
##   Diabetes            0           54
##   Sin diabetes        0          100
```

Aunque el modelo no se equivoque al discriminar no se garantiza que el modelo sea igual de perfecto para la clasificación. En este caso es posible notar que el modelo clasifica de manera errónea a 54 sujetos.

ERROR DE CLASIFICACIÓN

```
(clasi[1,2] + clasi[2,1])/sum(clasi)*100
```

```
## [1] 35.06494
```

El kernel gaussiano resultó ser muy bueno para discriminar, sin embargo, para clasificar nos da un valor de ***35%***

Kernel polynomial

1.Se aplica el kernel **polynomial**

```
MSV = svm(Outcome~.,data = entrenamiento,kernel = "polynomial", cost = 10,scale = F)
print(MSV)
```

```
##
## Call:
## svm(formula = Outcome ~ ., data = entrenamiento, kernel = "polynomial",
##     cost = 10, scale = F)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  polynomial
##        cost:  10
##      degree:  3
##      coef.0:  0
##
## Number of Support Vectors:  290
```

Con este modelo hay *290* posibles vectores.

2.Se prueba el poder de discriminación

```
prediclas = predict(object = MSV, newdata = entrenamiento[,-9])
discriminacion = table(entrenamiento$Outcome ,
                       prediclas ,
                       dnn=c("Clase real","Clase predicha"))
discriminacion
```

```
##               Clase predicha
## Clase real     Diabetes Sin diabetes
##   Diabetes           80          134
##   Sin diabetes       42          358
```

En el modelo de discriminación observamos que hay 176 sujetos en grupos que no pertenecen.

ERROR DE DISCRIMINACIÓN

```
(discriminacion[1,2] + discriminacion[2,1])/sum(discriminacion)*100
```

```
## [1] 28.6645
```

El kernel polynomial se equivoca casi en un ***29%*** al discriminar.

3.Se prueba el poder de clasificación

```
preds = predict(object = MSV, newdata = prueba[,-9])
clasi = table(prueba$Outcome,
              preds,
              dnn=c("Clase real","Clase predicha"))
clasi
```

```
##               Clase predicha
## Clase real     Diabetes Sin diabetes
##   Diabetes           23           31
##   Sin diabetes       12           88
```

Podemos ver que hay 43 sujetos en el grupo equivocado.

ERROR DE CLASIFICACIÓN

```
(clasi[1,2] + clasi[2,1])/sum(clasi)*100
```

```
## [1] 27.92208
```

El error de clasificación con este kernel es de un ***28%***

Kernel Sigmoid

1.Se aplica el kernel **sigmoid**

```
MSV = svm(Outcome~.,data = entrenamiento,kernel = "sigmoid", cost = 10,scale = F)
print(MSV)
```

```
##
## Call:
## svm(formula = Outcome ~ ., data = entrenamiento, kernel = "sigmoid",
##     cost = 10, scale = F)
##
##
## Parameters:
##    SVM-Type:  C-classification
##  SVM-Kernel:  sigmoid
##        cost:  10
##      coef.0:  0
##
## Number of Support Vectors:  428
```

Hay *428* posibles vectores.

2.Se prueba el poder de discriminación

```
prediclas = predict(object = MSV, newdata = entrenamiento[,-9])
discriminacion = table(entrenamiento$Outcome ,
                       prediclas ,
                       dnn=c("Clase real","Clase predicha"))
discriminacion
```

```
##               Clase predicha
## Clase real     Diabetes Sin diabetes
##   Diabetes            0          214
##   Sin diabetes        0          400
```

Discrimina de manera errónea a *214* sujetos.

ERROR DE DISCRIMINACIÓN

```
(discriminacion[1,2] + discriminacion[2,1])/sum(discriminacion)*100
```

```
## [1] 34.85342
```

El error de discriminación con el kernel sigmoid es de ***34.8%***

3.Se prueba el poder de clasificación

```
preds = predict(object = MSV, newdata = prueba[,-9])
clasi = table(prueba$Outcome,
              preds,
              dnn=c("Clase real","Clase predicha"))
```

En el modelo de clasificación hay 54 personas en el grupo equivocado.

ERROR DE CLASIFICACIÓN

```
(clasi[1,2] + clasi[2,1])/sum(clasi)*100
```

```
## [1] 35.06494
```

El error de clasificación es de un ***35%.***

Desempeño

En la siguiente tabla se puede apreciar toda la información obtenida anteriormente de manera más resumida.

Table 1: Desempeño de Kernel´s

Kernel	N° de vectores	Error de discriminación	Error de clasificación
Lineal	325	23%	20%
Gaussiano	614	0	35%
Polynomial	290	29%	28%
Sigmoid	428	35%	35%

Capítulo V

Conclusiones

Tras el análisis podemos deducir que no hay un modelo que se ajuste correctamente para la base de datos. No se logró llegar a lo esperado, es decir, encontrar un modelo que lograra discriminar y clasificar lo mejor posible. Dicho de otra manera tener un error mínimo de discriminación y de clasificación.

A pesar de todo decimos que para este caso el kernel lineal hizo un mejor trabajo con un error de 23% en discriminación y 20% en clasificación en comparación al resto (gaussiano, sigmoid y polynomial), ya que, desgraciadamente el porcentaje de error de los kernel ya mencionados era más alto (mayores al 30%). Es importante aclarar que se podrían obtener mejores resultados si se trabajara con variables que tuvieran un número menor de valores iguales a "0" los cuales son valores faltantes. Desgraciadamente a causa de esto estamos perdiendo información lo que ocasiona un problema para poder establecer un buen modelo. Como propuesta para encontrar un mejor modelo, realizar una selección de variables de manera minuciosa o recurrir a métodos de re-muestreo en conjunto con la aplicación de las máquinas de soporte vectorial.

Referencias

Montano, J.A. (2021) MÁQUINAS DE SOPORTE VECTORIAL ANALISIS SUPERVISADO [Diapositiva de PowerPoint]. Plataforma Microsoft Teams. https://uvmx.sharepoint.com/:p:/s/MineradeDatosyAprendizajeMquina/EQYae-ZriAJKpwqo8MU4WTUB62sISdpYA9fnk3qx-DLAng?e=KO9pG4

UCI Machine Learning. (2016). Pima Indians Diabetes Database [Data set]