Razonamiento bajo incertidumbre

Tarea 4. Detector de piel a base de distibuciones gaussianas multivariables

Ángel García Báez

2024-09-24

Índice

1 Instrucciones: **2**

2 Definiciones necesarias antes de comenzar **3**
2.1 Vector de medias: . . . 3
2.2 Matriz de Varianzas y Covarianzas . . . 3
2.3 Distribución de probabilidad normal multivariante . . . 3

3 Introducción **4**
3.1 Modelo de contingencia de datos sobre el mundo (Modelo generativo) . . . 4

4 ¿Como modelar la función $P(x|w)$? **5**

5 Imagenes a utilizar y forma de abordar el problema **6**

6 Resultados individuales **8**
6.1 Imagen 1: Denisse Guerrero de Belanova . . . 8
6.2 Imagen 2: Selena Quintanilla . . . 9
6.3 Imagen 3: Ana Sofía . . . 10

7 Resultados de un modelo con todos los pixeles. **11**
7.1 Imagen 1 con el modelo general: Denisse Guerrero de Belanova . . . 12
7.2 Imagen 2 con el modelo general: Selena Quintanilla . . . 13
7.3 Imagen 3 con el modelo general: Ana Sofía . . . 14

8 Conclusiones **15**

9 Bibliografía **16**

10 Anexos **17**
10.1 Codigo del algoritmo en lenguaje Julia. . . . 17
10.2 Codigo del desarrollo del ejercicio en R. . . . 18

1 Instrucciones:

Dada una imagen digital que va a ser denotada por $\vec{x}$ que contenga "piel", un rostro, o piel de un cuerpo:

1.- Definir la distribución de densidad del color de piel tal que: $P(\vec{x}|w = piel) = N_x(\vec{\mu}_{piel}, \Sigma_{piel})$ a partir de almenos 10 muestras de pixeles de piel.

2.- Definir la distribución de densidad de los pixeles del fondo tal que: $P(\vec{x}|w = fondo) = N_x(\vec{\mu}_{fondo}, \Sigma_{fondo})$ a partir de almenos 10 muestras de pixeles de fondo.

3.- Se deja a criterio libre la elección del parámetro λ para definir $p(w)$.

4.- Barrer todos los pixeles de la imagen de entrada y clarificarlos con el teorema de Bayes y generar una imagen de salida en donde el pixel tiene el color de la imagen si es piel y negro (o blanco) si es fondo.

2 Definiciones necesarias antes de comenzar

2.1 Vector de medias:

El vector de medias para una matriz sera definido como un vector fila de tamaño $1 \times P$ donde P es la cantidad de columnas que tenga la matriz de la que se quiere obtener.

$$\bar{x} = [\bar{x}_1 + \bar{x}_2 + \cdots + \bar{x}_p]$$

2.2 Matriz de Varianzas y Covarianzas

La forma de calcular la matriz de varianzas y covarianzas de la matriz de datos, puede resumirse en la siguiente expresión:

$$\Sigma = \frac{1}{n-1}(X^T X - n\bar{x}^T \bar{x})$$

Donde:

Σ = Matriz de varianzas y covarianzas
X = Matriz de datos
X^T = Matriz de datos transpuesta
n = Filas o casos de la matriz
$\bar{x}$ = Vector fila de las medias
$\bar{x}^T$ = Vector fila de las medias transpuesto

2.3 Distribución de probabilidad normal multivariante

$$f(\mathbf{x}) = \frac{1}{(2\pi)^{P/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\bar{x})^T \Sigma^{-1}(x-\bar{x})\right)$$

Donde:

P = Cantidad de variables.
X = Vector de datos de tamaño $1 \times P$.
$\bar{x}$ = Vector de medias de tamapo $1 \times P$.
Σ = Matriz de varianzas y covarianzas.
Σ^{-1} = Inversa de la matriz de varianzas y covarianzas.
$|\Sigma|$ = Determinante de la matriz de varianzas y covarianzas.

3 Introducción

Para el presente reporte se pidió seguir los lineamientos expresados al inicio, antes de poder llevarlos a cabo, es necesario abordar las matemáticas que hay detrás para poder hacer esto posible.

3.1 Modelo de contingencia de datos sobre el mundo (Modelo generativo)

Dentro de lo que se explicaba en la clase donde se reviso el material de la presentación 6 sobre aprendizaje e inferencia de los modelos, se hablo sobre la existencia de 2 tipos de modelos, los modelos generativos y los modelos discriminativos.

Los modelos discriminativos se avocan a modelar la frontera de decisión entre las clases de los datos (estados del mundo) sin tomar en cuenta como es que dichos datos fueron generados.

Los modelos generativos por otro lado, buscan entender como se generan los datos para cada clase (estado del mundo). Para ello es que se entrena un modelo mediante la regla de decisión de Bayes para que, dado un vector de parametros θ que proviene de un conjunto de entrenamiento y una función de probabilidad para las clases (estados del mundo), pueda tomarse la decisión respecto a donde es más probable que pertenezca un nuevo punto dado el conocimiento adquirido por el algortimo. Lo anterior se expresa del siguiente modo:

$$P(w|x) = \frac{P(x|w)P(w)}{\int P(x|w)P(w)}$$

El problema recae ahora en definir las funciones de probabilidad adecuadas para los datos, regresando al contexto del procesamiento de imágenes y bajo las condiciones de detectar piel y fondo dentro de diferentes fotografías, puede re-escribirse la expresión anterior como sigue:

$$P(w = Piel|x) = \frac{P(x|w = Piel)P(w = Piel)}{(P(x|w = Piel)P(w = Piel) + P(x|w = Fondo)P(w = Fondo))}$$

Donde:

$x =$ El vector que contiene la información RGB de 1 pixel.
$P(w = Piel) =$ La distribución Bernoulli de los pixeles que son piel.
$P(w = Fondo) =$ La distribución Bernoulli de los pixeles que son fondo.
$P(x|w = Piel) =$ Función verosímil de la información RGB de los pixeles que son de piel del conjunto de entrenamiento.
$P(x|w = Piel) =$ Función verosímil de la información RGB de los pixeles que son de fondo del conjunto de entrenamiento.

4 ¿Como modelar la función $P(x|w)$?

Para modelar todo lo anterior, se procede de la siguiente manera:

1.- Se escoge la distribución apropiada para $P(x)$, la cual sera una distribución normal multivariante.

2.- Los parámetros van a ser una función de la clase discreta (W), dado que para el ejercicio solo hay 2 clases, se puede usar una distribución Bernoulli, si fueran 3 clases o más tendría que usarse la distribución categórica. A modo que se puede expresar como

$$P(x|w) = Norm_x[\bar{x}_w, \Sigma_w]$$

3.- Los parámetros que puede tomar la función son:

$$\bar{x}_{w=Piel} \quad \Sigma_{w=Piel} \quad \bar{x}_{w=Fondo} \quad \Sigma_{w=Piel}$$

Dichos parámetros provienen previamente del conjunto de entrenamiento para calcular dichos parametros por máxima verosimilitud, un conjunto de entrenamiento etiquetados de pixeles de piel y otro conjunto de pixeles etiquetados como fondo.

5 Imagenes a utilizar y forma de abordar el problema

A continuación se muestra una selección de 3 imágenes donde aparecen 3 personas de distintas características mostrando partes de su piel como el rostro y los brazos.

Imagen 1	**Imagen 2**	**Imagen 3**
		
768x1152 = 884,736 px	266x320 = 85,120 px	768x1024 = 786,432 px

Table 1: Imagenes para el desarrollo del ejercicio

Las 3 imágenes tienen características completamente distintas, tanto en composición de colores como en resolución (tamaño), el objetivo es lograr hacer un modelo generativo capaz de distinguir los pixeles que son piel, de los que no son piel.

El desarrollo de este ejercicio se llevara a cabo en el lenguaje R con ayuda del lenguaje Julia por motivos que se explicaran más adelante en los resultados, por ahora, el plan para abordar el problema es el siguiente:

Paso 1: Cargar una imagen en lenguaje R y descomponerla en 3 vectores (1 por cada canal de color R,G y B) que posteriormente serán unidos en un solo dataframe, al cual ademas se le hará el etiquetado de las coordenadas de cada pixel (fila del dataframe) para su posterior reconstrucción y manipulación.

Paso 2: Se definió al ojo la proporción de piel que hay en la imagen (λ) para poder usar ese valor como probabilidad a priori de los datos.

$$P(w = Piel) = \lambda, \quad P(w = Fondo) = 1 - \lambda$$

Paso 3: Crear 2 subconjuntos de datos con distintas regiones de la imagen. Un subconjunto que contenga la región de la cara o donde se vea más piel y otro subconjunto para todo lo que no sea piel como el fondo y la ropa. Se le puso una etiqueta a cada uno de los datos, a los del subconjunto de piel, se les puso la etiqueta "piel" y a los del subconjunto de fondo, se les puso la etiqueta "fondo".

$$Piel = \begin{bmatrix} R & G & B & Etiqueta \\ \vdots & \vdots & \vdots & \vdots \\ Ri & G_i & B_i & Piel \end{bmatrix} \quad Fondo = \begin{bmatrix} R & G & B & Etiqueta \\ \vdots & \vdots & \vdots & \vdots \\ R_i & G_i & B_i & Fondo \end{bmatrix}$$

Paso 4: Seleccionar al azar 10 pixeles de cada subconjunto de datos.

$$Piel = \begin{bmatrix} R & G & B & Etiqueta \\ \vdots & \vdots & \vdots & \vdots \\ R10 & G_{10} & B_{10} & Piel \end{bmatrix} \quad Fondo = \begin{bmatrix} R & G & B & Etiqueta \\ \vdots & \vdots & \vdots & \vdots \\ R_{10} & G_{10} & B_{10} & Fondo \end{bmatrix}$$

Paso 5: Calcular el vector de medias y la matriz de covarianzas para cada uno de los subconjuntos de los pixeles etiquetados seleccionados al azar.

$$\bar{x}_{w=Piel} \quad \Sigma_{w=Piel} \quad \bar{x}_{w=Fondo} \quad \Sigma_{w=Piel}$$

Paso 6: Crear una distribución normal multivariante para los pixeles de la piel con el vector de medias y matriz de covarianzas calculados a partir de la muestra de 10 pixeles de piel y lo mismo para los 10 pixeles de fondo.

$$P(x|w = Piel) = Normal_3(\bar{x}_{w=Piel}, \Sigma_{w=Piel})$$

$$P(x|w = Fondo) = Normal_3(\bar{x}_{w=Fondo}, \Sigma_{w=Fondo})$$

Paso 7: Se evalúa el pixel $i - ésimo$ en la distribución normal de piel y se multiplica por la probabilidad a priori de que sea piel, despues se hace lo mismo con el pixel $i - ésimo$ en la distribución normal de fondo y se multiplica por la probabilidad a priori de que sea fondo.

$$P(x_i|w = Piel) * P(w = Piel)$$

$$P(x_i|w = Fondo) * P(w = Fondo)$$

Paso 8: Se suman los valores obtenidos en el paso 7:

$$Suma = P(x_i|w = Piel) * P(w = Piel) + P(x_i|w = Fondo) * P(w = Fondo)$$

Paso 9: Se evalúa la probabilidad de que el pixel pertenezca a la piel y si es mayor que la probabilidad de que sea fondo, se asigna como piel, en caso contrario se asigna como fondo.

$$\frac{P(x_i|w = Piel) * P(w = Piel)}{suma} > \frac{P(x_i|w = Fondo) * P(w = Fondo)}{suma} = Piel$$
$$\frac{P(x_i|w = Piel) * P(w = Piel)}{suma} < \frac{P(x_i|w = Fondo) * P(w = Fondo)}{suma} = Fondo$$

Paso 10: Se repiten los pasos del 7 al 9 hasta etiquetar todos los pixeles del conjunto de datos.

Los pasos descritos anteriormente permiten crear un modelo generativo individual por cada una de las imagenes. Cabe mencionar que para crear todos los subconjuntos de entrenamiento se extrajeron 10 pixeles de piel y 10 de fondo, esta extracción se hizo al azar fijando una semilla (12) para la replicabilidad de los resultados.

Se utilizara el lenguaje Julia para poder implementar el algoritmo previamente descrito debido a que tiene mayor velocidad de procesamiento que el lenguaje R, el cual durante pruebas internas resulto ser extremadamente lento para computar las enormes cantidades de pixeles que hay en las imágenes (20 minutos para computar la asignación de 200,000 pixeles). Por lo que toda la manipulación y reconstrucción de los datos se hará mediante R y el proceso de entrenamiento y etiquetado de los pixeles se corrió en Julia pero desde dentro del mismo R.

6 Resultados individuales

6.1 Imagen 1: Denisse Guerrero de Belanova

Para la primera foto que tiene unas dimensiones de 768 X 1152 (884,736 pixeles) se asumió una distribución a priori de que el 40% de la imagen es piel ($\lambda = 0.4$), se muestra a continuación la región de la piel de la cara extraída junto con la distribución de los pixeles seleccionados para ser piel y fondo.

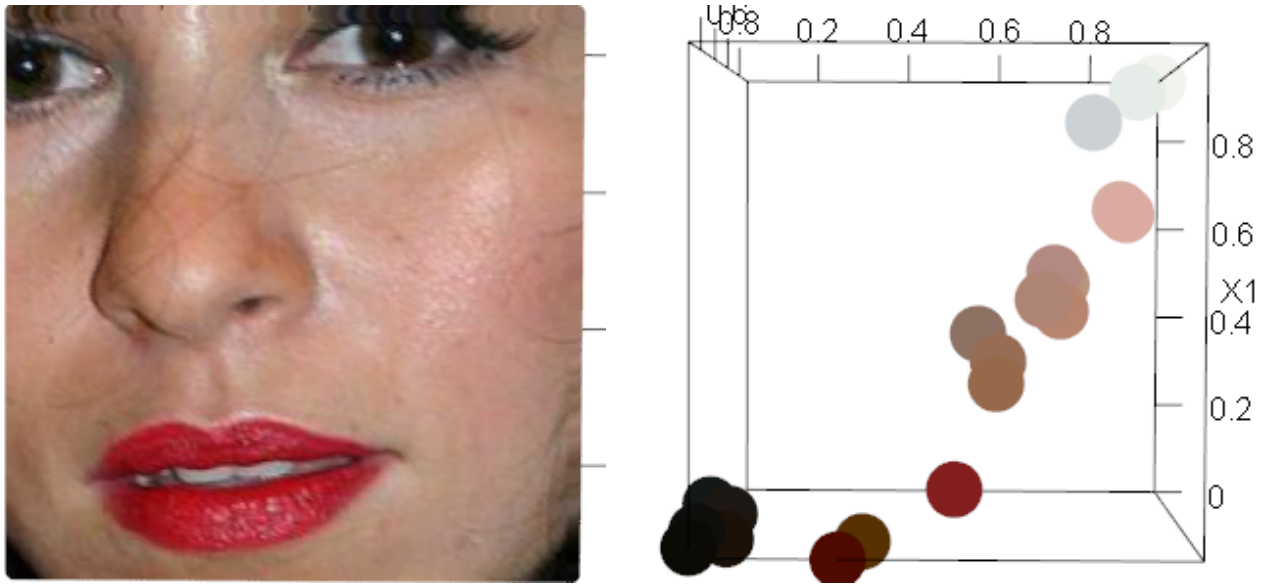

Table 2: Área de la piel y pixeles de entrenamiento para la foto 1.

A continuación se muestran los resultados de aplicar el algoritmo solo con los datos de la imagen 1:

Table 3: Resultados de la imagen 1 para detección de piel y fondo.

6.2 Imagen 2: Selena Quintanilla

Para la segunda foto que tiene unas dimensiones de 320 X 266 (85,120 pixeles) se asumió una distribución a priori de que el 40% de la imagen es piel ($\lambda = 0.4$), se muestra a continuación la región de la piel de la cara extraída junto con la distribución de los pixeles seleccionados para ser piel y fondo.

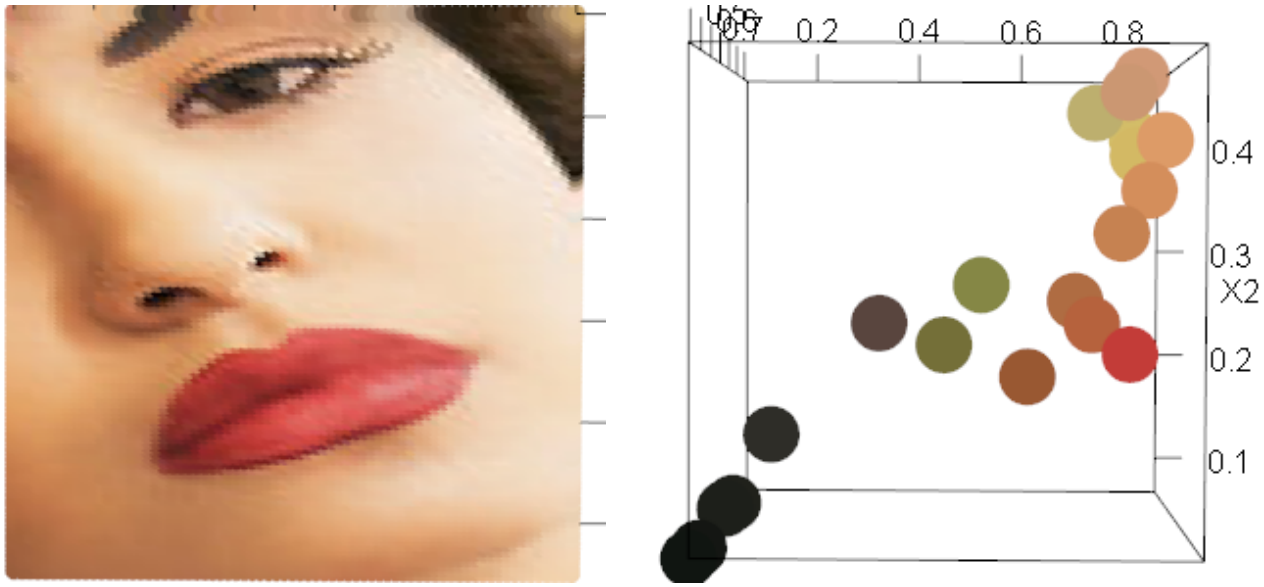

Table 4: Área de la piel y pixeles de entrenamiento para la foto 2.

A continuación se muestran los resultados de aplicar el algoritmo solo con los datos de la imagen 2:

Table 5: Resultados de la imagen 2 para detección de piel y fondo.

6.3 Imagen 3: Ana Sofía

Para la tercera foto que tiene unas dimensiones de 1024 X 768 (786,432 pixeles) se asumió una distribución a priori de que el 30% de la imagen es piel ($\lambda = 0.3$), se muestra a continuación la región de la piel de la cara extraída junto con la distribución de los pixeles seleccionados para ser piel y fondo.

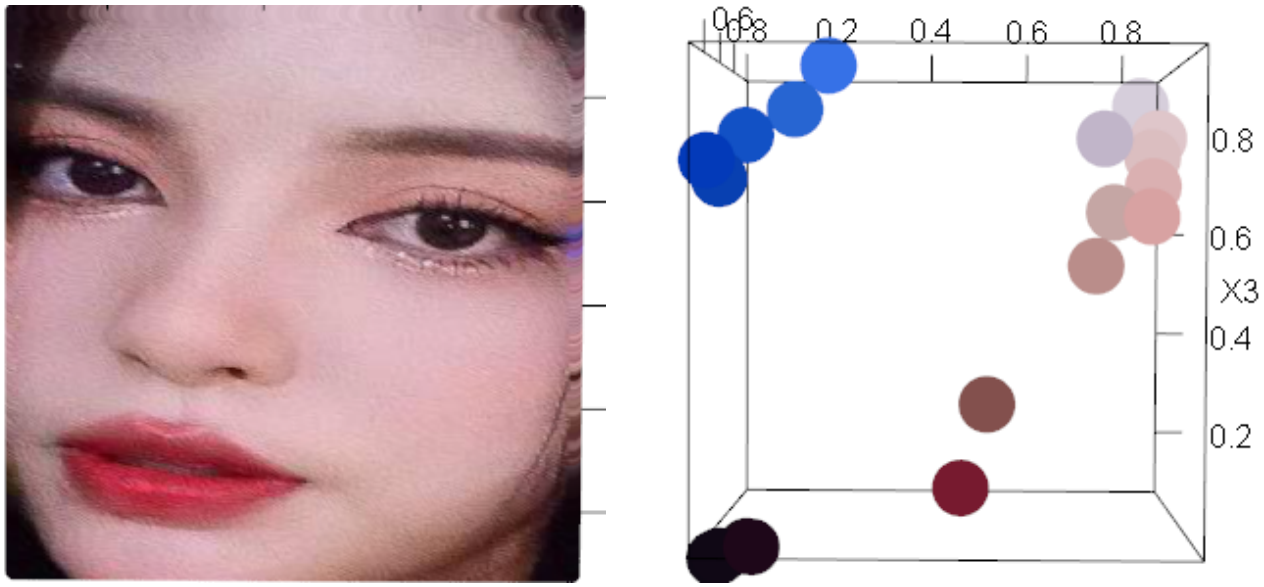

Table 6: Área de la piel y pixeles de entrenamiento para la foto 3.

A continuación se muestran los resultados de aplicar el algoritmo solo con los datos de la imagen 3:

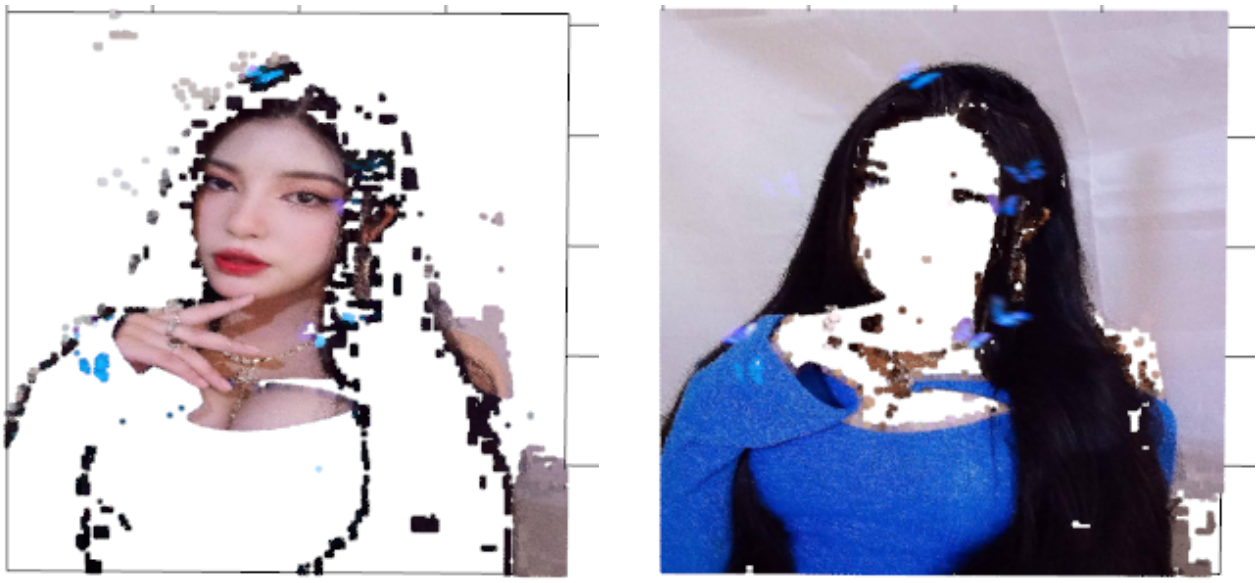

Table 7: Resultados de la imagen 3 para detección de piel y fondo.

7 Resultados de un modelo con todos los pixeles.

Después de observar los resultados que arroja el algoritmo cuando únicamente se utiliza a la misma imagen para entrenarlo, se plantea la idea de tratar de expandirlo hacia un concepto más general. Para ello se tomaron a todos los subconjuntos de entrenamiento de las imágenes y se convirtieron en un solo subconjunto con sus respectivas etiquetas para entrar al modelo. A continuación se ve la representación de los pixeles en el espacio de 3 dimensiones:

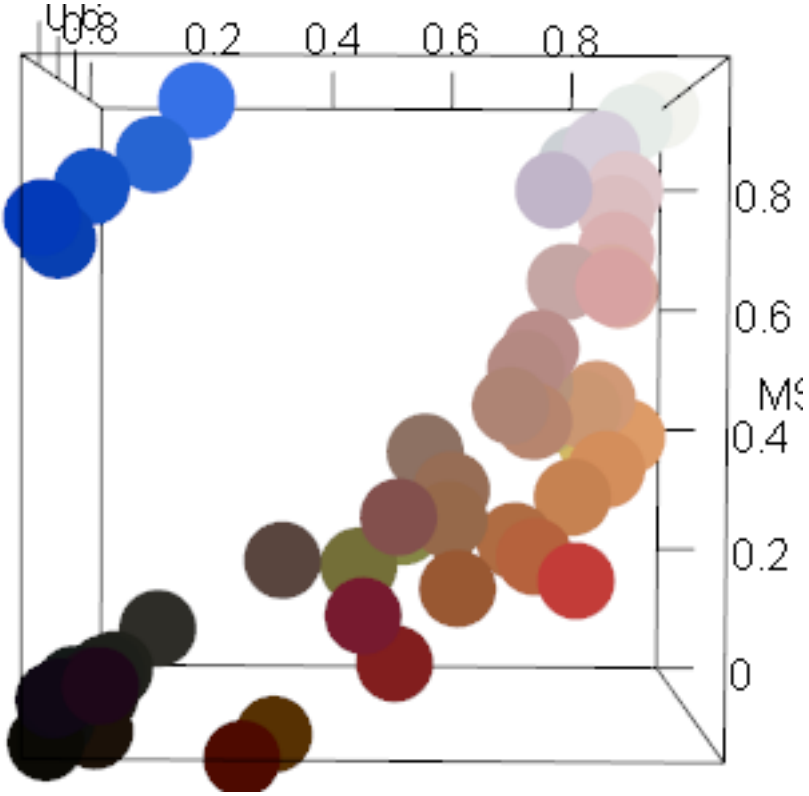

Table 8: Piexles de fondo y de piel de entrenamiento de las 3 imagenes.

Con dicho conjunto de entrenamiento se entreno un modelo para evaluar a todas las imágenes, a continuación se presentan los resultados de este modelo "general" entrenado por 20 pixeles de cada una de las 3 imágenes etiquetados como piel y fondo.

7.1 Imagen 1 con el modelo general: Denisse Guerrero de Belanova

A continuación se muestra el comparativo entre lo que logra el modelo entrenado solo con los datos de la imagen 1 contra los resultados del modelo general:

Resultados del modelo individual

Resultados del modelo general

Table 9: Comparación de detección de piel y fondo usando modelos individuales y generales en la imagen 1.

7.2 Imagen 2 con el modelo general: Selena Quintanilla

A continuación se muestra el comparativo entre lo que logra el modelo entrenado solo con los datos de la imagen 2 contra los resultados del modelo general:

Resultados del modelo individual

Resultados del modelo general

Table 10: Comparación de detección de piel y fondo usando modelos individuales y generales.

7.3 Imagen 3 con el modelo general: Ana Sofía

A continuación se muestra el comparativo entre lo que logra el modelo entrenado solo con los datos de la imagen 3 contra los resultados del modelo general:

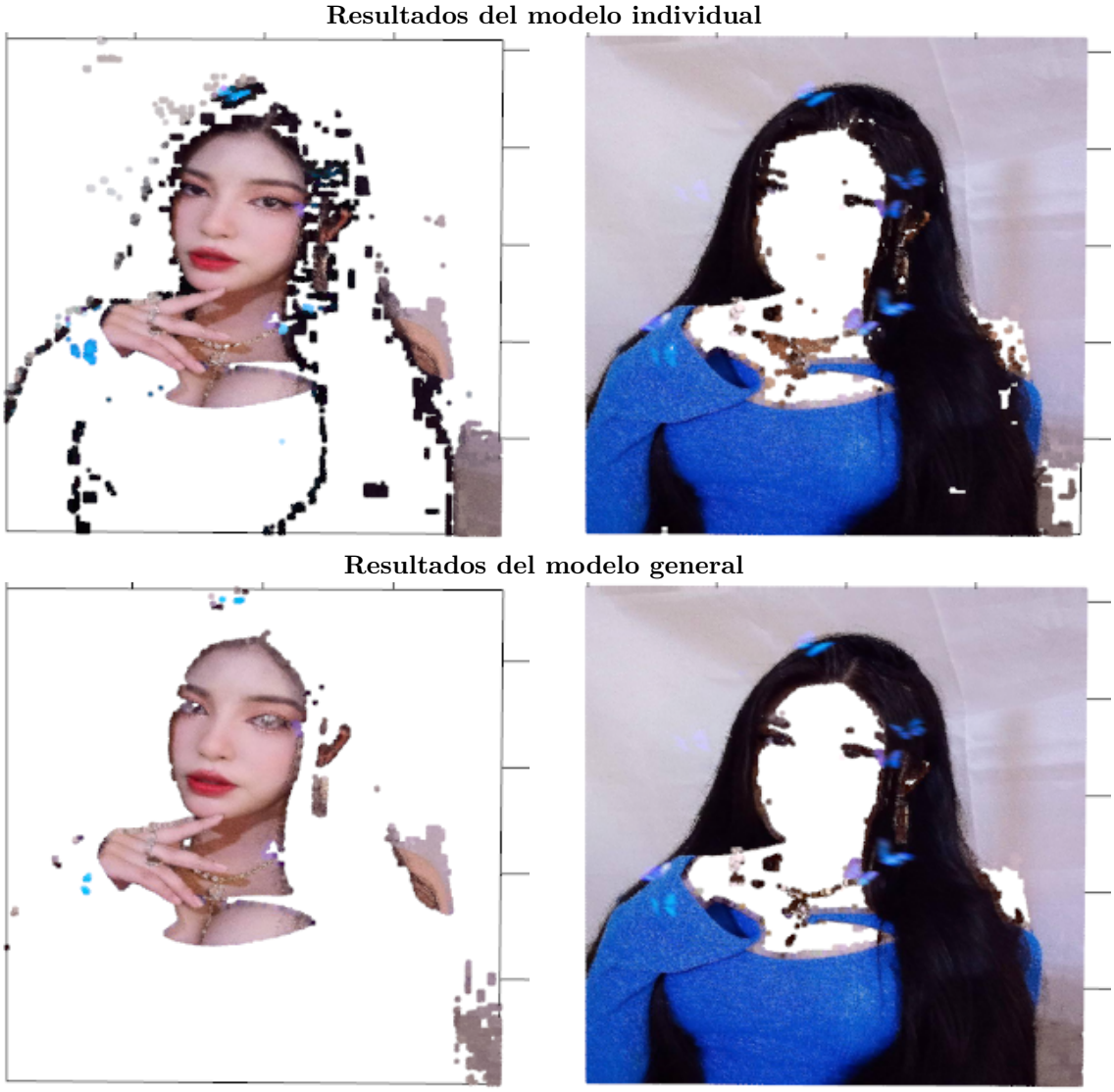

Table 11: Comparación de detección de piel y fondo usando modelos individuales y generales.

8 Conclusiones

Lo primera que cabe destacar el presente trabajo, fue la limitación en computo que tenia el lenguaje R y la razón por la que se opto hacer esta implementación híbrida con ayuda del lenguaje Julia. Se encontró que para el caso de la primera imagen, al correr el algoritmo en R, tardaba demasiado, se dejo correr la computadora por 20 minutos y al ver que no terminaba, se decidió finalizar la ejecución y se consulto el numero de pixeles que había evaluado, mostrando así que unicamente había logrado evaluar cerca de 200,000 pixeles en esos 20 minutos. La complejidad y lentitud de los cálculos fueron el motor para tomar la decisión de traducir el algoritmo de R a Julia y a encontrar una forma de que pudiera correr dicho algoritmo en Julia desde R.

Se mantuvo en todo momento a R para la manipulación de los datos y la construcción de las imágenes pero el algoritmo se mandaba a evaluar en Julia donde hacia el computo de los 884,736 pixeles en apenas 2 segundos.

Lo siguiente, es que cabe mencionar que la detección de la piel es más complicado de lo que parece, puesto que la piel por si misma puede ser de distintas tonalidades naturalmente y para el caso del ejercicio, se ve afectada sustancialmente por la luz y la forma en que se toma la imagen.

Para los modelos que fueron creados solamente con una imagen, los resultados que arroja son relativamente buenos considerando que cumplen en su mayoría su cometido, detectar pixeles de piel y separarlos, aunque en el camino confunde un poco de la ropa y el fondo así como que deja un vació alrededor de los ojos.

Para el modelo general, se puede observar precisión al clasificar la piel de las imágenes 2 y 3, sin embargo, para la imagen 1 comienza a detectar más fondo como si fuera piel, por lo que valdría el esfuerzo seguir iterando con una mayor cantidad de datos de piel.

9 Bibliografía

Prince, S. J. (2012). Computer vision: models, learning, and inference. Cambridge University Press.

10 Anexos

10.1 Codigo del algoritmo en lenguaje Julia.

```
# eval = F
#### Librerias necesarias para el procesamiento ###
using CSV # Para cargar archivos CSV
using DataFrames  # Para cargar los datos en un DataFrame
using Distributions # Para las distribuciones de probabilidad
using Statistics # Para calcular el vector de medias y la covarianza
#### Funciones auxiliares para el camino ####
#### Función para calcular la densidad para un punto nuevo ###
function dmvn(x, dist)
  pdf(dist, x)
end
#### Función para calcular la distribución normal multivariada ####
### y la clásificación en base al criterio bayesiano ###
function Bayesiano(X,prueba,cate,lambdaSI)
  ### Extraer las matrices de cada distribución ###
  # Para Piel
  SImed = mean.(eachcol((X[X[:, 4] .== cate, 1:3])))
  SIvar = cov(Matrix((X[X[:, 4] .== cate, 1:3])))
  # Para fondo
  NOmed = mean.(eachcol((X[X[:, 4] .!= cate, 1:3])))
  NOvar = cov(Matrix((X[X[:, 4] .!= cate, 1:3])))
  ### Distribuciones pre armadas ###
  # Distribución multivariada normal para SÍ y para NO
  dist1 = MvNormal(SImed,SIvar,)
  dist2 = MvNormal(NOmed,NOvar,)
  # Definir la apriori
  lambdaNO = 1-lambdaSI
  ### Hacer la implementación Bayesiana
  # Dondde guardar los resultados
  resultados = DataFrame(si = Float64[], no = Float64[],Etiqueta = String[])
    for i in 1:size(prueba)[1] # Comienzo un for que va desde 1 hasta nrow de prueba
    punto = Array(prueba[i,1:3])
    # Evaluo su distribución normal para PIEL
    nSI = dmvn(punto,dist1)
    nNO = dmvn(punto,dist2)
    # Calcular la norma (suma de probabilidades ponderadas)
    norma = (nSI * lambdaSI + nNO * lambdaNO)
    # Aplicar la regla de decisión de Bayes
    SIrgb = (nSI * lambdaSI) / norma
    NOrgb = (nNO * lambdaNO) / norma
    # Almacenar los resultados
    push!(resultados, (si = SIrgb, no = NOrgb,
                       Etiqueta = map((x, y) -> x > y ? "Piel" : "Fondo", SIrgb, NOrgb)))
  end
  nprueba = hcat(prueba,resultados )
return (nprueba)
end
```

10.2 Codigo del desarrollo del ejercicio en R.

```
rm(list=ls())
# Cargar las librerias
# Cargar la imagen en R
## Cargar librerias necesarias ##
library(png) # Libreria para leer PNG's
library(flextable)# LIbreria para hacer tablas bonitas
library(scatterplot3d) #gráficar en 3D
library(rsvg) # Manipular archivos SVG
library(jpeg) # Comprimir el SVG a JPEG
library(rgl) # OpenGL pero para los gráficos en 3D
library(corrplot) # Para gráficar las correlaciones
library(NbClust) # Correr el Kmedias
library(factoextra) # Obtener exploraciones para el kmedias
library(JuliaCall) # Para pasarme objetos a Julia
### Función que extrae los pixeles
### Los separa en 3 canales de colores
### Calcula las posiciones de los pixeles para reconstruir la imagen
### Calcula el color de cada pixel a Hexadecimasl
### Comprime todo en un DataFrame
pixeles = function(imagen){
  # Extraer los datos
  datos = cbind.data.frame(R = as.vector(imagen[ , ,1]), # Extraer inf de R
                           G = as.vector(imagen[ , ,2]), # Extraer inf de G
                           B = as.vector(imagen[ , ,3])) # Extraer inf de B
  # Extraer las coordenadas de cada pixel para re-mapearlos
  #Filas
  nf = dim(imagen)[1]
  #Columnas
  nc = dim(imagen)[2]
  # Hacer el ordenamiento para remapearlos
  X = rep(nf:1,nc)
  Y = rep(1:nc,each = nf)
  datos$X = X
  datos$Y = Y
  # Añadir el vector de colores al conjunto de datos
  datos$color = rgb(datos$R,datos$G,datos$B)
  # Regresar el objeto transformado
  return(datos)
}
```

```
#### Imagen 1: Denisse Guerrero ####
### Cargando la imagen desde la ruta local
ruta1 = "imagenes/IM2.jpg" # Ruta local de la imagen
imagen1 = readJPEG(ruta1) # Descomponer la imagen en sus canales RGB
datos1 = pixeles(imagen1)
# Selección de la Carita
subSI1 = datos1[(datos1$X>=610 &datos1$X<=815) &
                  (datos1$Y>=300 &datos1$Y<=490), ]
plot3d(x = subSI1$Y,
       y = 1,
       z = subSI1$X,
       col = rgb(subSI1$R,subSI1$G,subSI1$B),size = 10)# Fondo
rgl.snapshot("R1.1.png", fmt = "png")
# Sembrar los parametros de replicabilidad
semilla = 12
set.seed(semilla)
N = 10
mSI1 = subSI1[sample(1:nrow(subSI1),
                     size= N),]
mSI1$etiqueta = "Piel"
cond1 = (datos1$X >= 0 & datos1$X <= 350) & (datos1$Y >= 200 & datos1$Y <= 1152)
cond2 <- (datos1$X >= 900 & datos1$X <= 1152) & (datos1$Y >=  0 & datos1$Y <= 768 )
subNO1 = datos1[cond1|cond2, ]
# Extraigo una muestra random de los 10 puntos
set.seed(semilla)
mNO1 = subNO1[sample(1:nrow(subNO1),
                     size= N),]
mNO1$etiqueta = "Fondo"
### Conjunto de entrenamiento ###
X1 = rbind.data.frame(mSI1[,c(1:3,7)],
                      mNO1[,c(1:3,7)])
### Pixeles de piel y Fondo en el espacio
plot3d(x = X1$R,y = X1$G,z = X1$B,
       col = rgb(X1$R,X1$G,X1$B),size = 30)
rgl.snapshot("R1.2.png", fmt = "png")
#### Evaluar con los datos de la foto 1 ####
prueba1 =  julia_call("Bayesiano",X1,datos1,"Piel",0.4)
# Ver los resultados de piel
plot3d(x = prueba1[prueba1$Etiqueta=="Piel",]$Y,
       y = 1,
       z = prueba1[prueba1$Etiqueta=="Piel",]$X,
       col = prueba1[prueba1$Etiqueta=="Piel",]$color)
rgl.snapshot("R1.3.png", fmt = "png")
# Ver los resultados de fondo
plot3d(x = prueba1[prueba1$Etiqueta!="Piel",]$Y,
       y = 1,
       z = prueba1[prueba1$Etiqueta!="Piel",]$X,
       col = prueba1[prueba1$Etiqueta!="Piel",]$color)
rgl.snapshot("R1.4.png", fmt = "png")

#### Imagen 2: Selena quintanilla ####
### Cargando la imagen desde la ruta local
ruta2 = "imagenes/IM3.jpg" # Ruta local de la imagen
```

```r
imagen2 = readJPEG(ruta2) # Descomponer la imagen en sus canales RGB
datos2 = pixeles(imagen2)
# Carita
subSI2 = datos2[(datos2$X>=130 &datos2$X<=240) &
                  (datos2$Y>=150 &datos2$Y<=220), ]
plot3d(x = subSI2$Y,
       y = 1,
       z = subSI2$X,
       col = rgb(subSI2$R,subSI2$G,subSI2$B),size = 10)# Fondo
rgl.snapshot("R2.1.png", fmt = "png")
### Mostrar la subregión de la carita
# Carita
set.seed(semilla)
mSI2 = subSI2[sample(1:nrow(subSI2),
                     size= N),]
mSI2$etiqueta = "Piel"
# Para el fondo
cond1 = (datos2$X >= 50 & datos2$X <= 300) & (datos2$Y >= 0 & datos2$Y <= 100)
subNO2 = datos2[cond1, ]

# Extraigo una muestra random de los 10 puntos
set.seed(semilla)
mNO2 = subNO2[sample(1:nrow(subNO2),
                     size= N),]
mNO2$etiqueta = "Fondo"
### Conjunto de entrenamiento ###
X2 = rbind.data.frame(mSI2[,c(1:3,7)],
                      mNO2[,c(1:3,7)])
### Pixeles de piel y Fondo en el espacio
plot3d(x = X2$R,y = X2$G,z = X2$B,
       col = rgb(X2$R,X2$G,X2$B),size = 30)
rgl.snapshot("R2.2.png", fmt = "png")

#### Evaluar con los datos de la foto 2 ####
prueba2 =  julia_call("Bayesiano",X2,datos2,"Piel",0.4)

# Ver los resultados de piel
plot3d(x = prueba2[prueba2$Etiqueta=="Piel",]$Y,
       y = 1,
       z = prueba2[prueba2$Etiqueta=="Piel",]$X,
       col = prueba2[prueba2$Etiqueta=="Piel",]$color)
rgl.snapshot("R2.3.png", fmt = "png")
# Ver los resultados de fondo
plot3d(x = prueba2[prueba2$Etiqueta!="Piel",]$Y,
       y = 1,
       z = prueba2[prueba2$Etiqueta!="Piel",]$X,
       col = prueba2[prueba2$Etiqueta!="Piel",]$color)
rgl.snapshot("R2.4.png", fmt = "png")

#### Imagen 3: Ana Sofía ####
### Cargando la imagen desde la ruta local
ruta3 = "imagenes/IM4.jpg" # Ruta local de la imagen
imagen3 = readJPEG(ruta3) # Descomponer la imagen en sus canales RGB
```

```r
datos3 = pixeles(imagen3)
# Carita
subSI3 = datos3[(datos3$X>=520 &datos3$X<=790) &
                  (datos3$Y>=270 &datos3$Y<=450), ]
### Mostrar la subregión de la carita
plot3d(x = subSI3$Y,
       y = 1,
       z = subSI3$X,
       col = rgb(subSI3$R,subSI3$G,subSI3$B),size = 10)# Fondo
rgl.snapshot("R3.1.png", fmt = "png")

# Carita
set.seed(semilla)
mSI3 = subSI3[sample(1:nrow(subSI3),
                     size= N),]
mSI3$etiqueta = "Piel"

# Para el fondo
con1 = (datos3$X >= 850 & datos3$X <= 1000) & (datos3$Y >= 0 & datos3$Y <= 768)
con2 = (datos3$X >= 0 & datos3$X <= 230) & (datos3$Y >= 0 & datos3$Y <= 768)
subNO3 = datos3[con1|con2, ]

# Extraigo una muestra random de los 10 puntos
set.seed(semilla)
mNO3 = subNO3[sample(1:nrow(subNO3),
                     size= N),]
mNO3$etiqueta = "Fondo"
### Conjunto de entrenamiento ###
X3 = rbind.data.frame(mSI3[,c(1:3,7)],
                      mNO3[,c(1:3,7)])
### Pixeles de piel y Fondo en el espacio
plot3d(x = X3$R,y = X3$G,z = X3$B,
       col = rgb(X3$R,X3$G,X3$B),size = 30)
rgl.snapshot("R3.2.png", fmt = "png")

#### Evaluar con los datos de la foto 3 ####
prueba3 =  julia_call("Bayesiano",X3,datos3,"Piel",0.3)

# Ver los resultados de piel
plot3d(x = prueba3[prueba3$Etiqueta=="Piel",]$Y,
       y = 1,
       z = prueba3[prueba3$Etiqueta=="Piel",]$X,
       col = prueba3[prueba3$Etiqueta=="Piel",]$color)
rgl.snapshot("R3.3.png", fmt = "png")
# Ver los resultados de fondo
plot3d(x = prueba3[prueba3$Etiqueta!="Piel",]$Y,
       y = 1,
       z = prueba3[prueba3$Etiqueta!="Piel",]$X,
       col = prueba3[prueba3$Etiqueta!="Piel",]$color)
rgl.snapshot("R3.4.png", fmt = "png")

#### Haciendo un modelo de todas las muestras ####
M = rbind.data.frame(X1,X2,X3)
```

```
### Pixeles de piel y Fondo en el espacio
plot3d(x = M$R,y = M$G,z = M$B,
       col = rgb(M$R,M$G,M$B),size = 30)
rgl.snapshot("R4.png", fmt = "png")


#### Evaluar la imagen 1 con el modelo general ####
prueba1 =  julia_call("Bayesiano",M,datos1,"Piel",0.4)

# Ver los resultados de piel
plot3d(x = prueba1[prueba1$Etiqueta=="Piel",]$Y,
       y = 1,
       z = prueba1[prueba1$Etiqueta=="Piel",]$X,
       col = prueba1[prueba1$Etiqueta=="Piel",]$color)
rgl.snapshot("R5.1.png", fmt = "png")
# Ver los resultados de fondo
plot3d(x = prueba1[prueba1$Etiqueta!="Piel",]$Y,
       y = 1,
       z = prueba1[prueba1$Etiqueta!="Piel",]$X,
       col = prueba1[prueba1$Etiqueta!="Piel",]$color)
rgl.snapshot("R5.2.png", fmt = "png")

#### Evaluar la imagen 2 con el modelo general ####
prueba2 =  julia_call("Bayesiano",M,datos2,"Piel",0.4)

# Ver los resultados de piel
plot3d(x = prueba2[prueba2$Etiqueta=="Piel",]$Y,
       y = 1,
       z = prueba2[prueba2$Etiqueta=="Piel",]$X,
       col = prueba2[prueba2$Etiqueta=="Piel",]$color)
rgl.snapshot("R6.1.png", fmt = "png")
# Ver los resultados de fondo
plot3d(x = prueba2[prueba2$Etiqueta!="Piel",]$Y,
       y = 1,
       z = prueba2[prueba2$Etiqueta!="Piel",]$X,
       col = prueba2[prueba2$Etiqueta!="Piel",]$color)
rgl.snapshot("R6.2.png", fmt = "png")

#### Evaluar la imagen 3 con el modelo general ####
prueba3 =  julia_call("Bayesiano",M,datos3,"Piel",0.2)
# Ver los resultados de piel
plot3d(x = prueba3[prueba3$Etiqueta=="Piel",]$Y,
       y = 1,
       z = prueba3[prueba3$Etiqueta=="Piel",]$X,
       col = prueba3[prueba3$Etiqueta=="Piel",]$color)
rgl.snapshot("R7.1.png", fmt = "png")
# Ver los resultados de fondo
plot3d(x = prueba3[prueba3$Etiqueta!="Piel",]$Y,
       y = 1,
       z = prueba3[prueba3$Etiqueta!="Piel",]$X,
       col = prueba3[prueba3$Etiqueta!="Piel",]$color)
rgl.snapshot("R7.2.png", fmt = "png")
```