DETECCIÓN AUTOMÁTICA DE PIEL CON APRENDIZAJE SUPERVISADO

1) Dada una imagen RGB en donde aparezca piel y fondo variable.

   A) Tomar algunas muestras Ipiel=I0=10 de pixeles de piel y calcular una gaussiana tridimensional que represente los colores de la piel y su dispersión.
   - μpiel, Σpiel en espacio RGB.
   - Con esto queda definido G(X,μpiel, Σpiel).

   B) Tomar algunas muestras de Ifondo=I1=10 de pixeles de fondo de la imagen (no piel) y calcular una gaussiana tridimensional que represente los colores del fondo y su dispersión.
   - μpiel, Σpiel en espacio RGB.
   - Con esto queda definido G(X,μpiel, Σpiel).

   C) Calcular la probabilidad a priori P(W), Pr(W=piel),Pr(W=fondo).
   D) Generar el mapa termico de los likelihood de no piel log(P(X|W=fondo=0)).
   E) Mapa termico de piel log(P(X|W=piel=1)).
   F) Mapa termico P(W=1|X*).
   G) Imagen binaria P(W=1|X*)>Threshold T=0.4, T=0.5, T=0.6.

El código esta implementado en python:

```
from email.base64mime import header_length
from email.mime import image
from re import M
from sys import maxunicode
import numpy as np
import cv2
import random
import math
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from mpl_toolkits.mplot3d import Axes3D
import time
from scipy.special import expit
inicio=time.time()
skinc=np.array([[0,0]])
backc=np.array([[0,0]])
puntos=10
clicks=0
p=math.pi
Iskin=0.6
Iback=0.4

#funcion para obtener los arrays de piel y fondo
def mouse1(event,x,y,flags,param):
```

```
        global clicks
        #array para obtener el array de piel
        if event == cv2.EVENT_LBUTTONDOWN and clicks<puntos:
                global skinc
                aux=np.array([[x,y]])
                skinc=np.append(skinc,aux,axis=0)
                clicks=clicks+1
                print('De click en un pixel de piel, faltan:',(puntos-clicks))

        #array para obtener el array de fondo
        if event == cv2.EVENT_LBUTTONDOWN and clicks<2*puntos+1 and clicks>=puntos:
                global backc
                aux=np.array([[x,y]])
                backc=np.append(backc,aux,axis=0)
                clicks=clicks+1
                print('De click en un pixel de fondo, faltan:',(2*puntos-clicks+1))

        #funcion para cerrar clickeo en imagen
        if clicks==2*puntos+1:
        cv2.destroyAllWindows()

#Transpuesta de matriz
def transpose(matrix):
        result = [[None for i in range(len(matrix))] for j in range(len(matrix[0]))]
        for i in range(len(matrix[0])):
                for j in range(len(matrix)):
                        result[i][j] = matrix[j][i]
return result

#Carga de imagen
imagen = cv2.imread('imagen5.jpg')
#cv2.imshow('imagen',imagen)
resolucionx=1200
resoluciony=800
print("La resolución de la imagen es:")
print('X:',resolucionx)
print('Y:',resoluciony)
print("\nEl total de pixeles es:")
print(resolucionx*resoluciony)
print('De click en un pixel de piel faltan:',(puntos-clicks))

while clicks<2*puntos+1:
        cv2.waitKey(0) & 0xFF
        cv2.namedWindow('imagen')
        cv2.setMouseCallback('imagen',mouse1)
        cv2.imshow('imagen',imagen)
#-----------------------------------CALCULO DE VECTOR MEDIO PIEL-----------------------------------
```

```
#Vector auxiliar para calcular vector promedio
sumrgb=np.zeros((3,1))

#Contador auxiliar para calcular promedio
cont=0

#Creacion de vector promedio
vmskin=np.zeros((3,1))

#Extraer componentes B,G,R de un solo pixel (opencv los da en ese orden y se cargan las
coordenadas en orden Y X]
for i in range (1,puntos+1,1):
        x=skinc[i][0]
        y=skinc[i][1]
        (b, g, r) = imagen[y, x]
        rgb=np.array([
        [r],
        [g],
        [b]
        ])
        rgb=rgb/255;
        sumrgb=sumrgb+rgb
        cont=cont+1

#Calculo de vector promedio
vmskin=sumrgb/cont;
print("\nEl vector promedio de piel es:")
print(vmskin)

#-----------------------CREACION DE MATRIZ DE COVARIANZA DE PIEL-----------------------
#Creacion de matriz auxiliar
sumamatriz=np.zeros((3,3))
cont=0

#Extraer componentes B,G,R de un solo pixel (opencv los da en ese orden y se cargan las
coordenadas en orden Y X]
for i in range (1,puntos+1,1):
        x=skinc[i][0]
        y=skinc[i][1]
        (b, g, r) = imagen[y, x]
        rgb=np.array([
        [r],
        [g],
        [b]
        ])
        rgb=rgb/255;
        #Calculo de vector x-m
```

```
        x_m=rgb-vmskin
        #Transpuesta de x-m
        x_mt=transpose(x_m)
        #Creacion de matriz (x-m)(x-m)^t
        matriz=x_m*x_mt
        sumamatriz=sumamatriz+matriz
        cont=cont+1

cxskin=np.zeros((3,3))
cxskin=sumamatriz/(cont-1);
print("\nLa matriz de covarianza de piel es:")
print(cxskin)

#-----------------------------------------CALCULO DE VECTOR MEDIO DE
FONDO-----------------------------
#Vector auxiliar para calcular vector promedio
sumrgbback=np.zeros((3,1))

#Contador auxiliar para calcular promedio
cont=0

#Creacion de vector promedio
vmback=np.zeros((3,1))

#Extraer componentes B,G,R de un solo pixel (opencv los da en ese orden) y se cargan las
coordenadas en orden Y X)
for i in range (2,puntos+2,1):
        x=backc[i][0]
        y=backc[i][1]
        (b, g, r) = imagen[y, x]
        rgb=np.array([
        [r],
        [g],
        [b]
        ])
        rgb=rgb/255;
        sumrgbback=sumrgbback+rgb
        cont=cont+1

#Calculo de vector promedio
vmback=sumrgbback/cont;
print("\nEl vector promedio de fondo es:")
print(vmback)

#-------------------------CREACION DE MATRIZ DE COVARIANZA DE FONDO-------------------------
#Creacion de matriz auxiliar
sumamatrizback=np.zeros((3,3))
```

```
cont=0

#Extraer componentes B,G,R de un solo pixel (opencv los da en ese orden y se cargan las
coordenadas en orden Y X)
for i in range (2,puntos+2,1):
        x=backc[i][0]
        y=backc[i][1]
        (b, g, r) = imagen[y, x]
        rgb=np.array([
        [r],
        [g],
        [b]
        ])
        rgb=rgb/255;

#Calculo de vector x-m
x_m=rgb-vmback
#Transpuesta de x-m
x_mt=transpose(x_m)
#Creacion de matriz (x-m)(x-m)^t
matriz=x_m*x_mt
sumamatrizback=sumamatrizback+matriz
cont=cont+1

cxback=np.zeros((3,3))
cxback=sumamatrizback/(cont-1);
print("\nLa matriz de covarianza de fondo es:")
print(cxback)

#----------------------------PROBABILIDAD PIEL-----------------------------
matriznskin=np.zeros((resoluciony,resolucionx))
logmatriznskin=np.zeros((resoluciony,resolucionx))
#Calculo del determinante de la matriz de covarianza
det=np.linalg.det(cxskin)
for y in range(resoluciony):
        for x in range(resolucionx):
                (b, g, r) = imagen[y, x]
                rgb=np.array([
        [r],
        [g],
        [b]
        ])
        rgb=rgb/255;
        x_mu=rgb-vmskin
        x_mut=transpose(x_mu)
        auxiliar2=np.zeros((1,3))
        cxi=np.linalg.inv(cxskin)
```

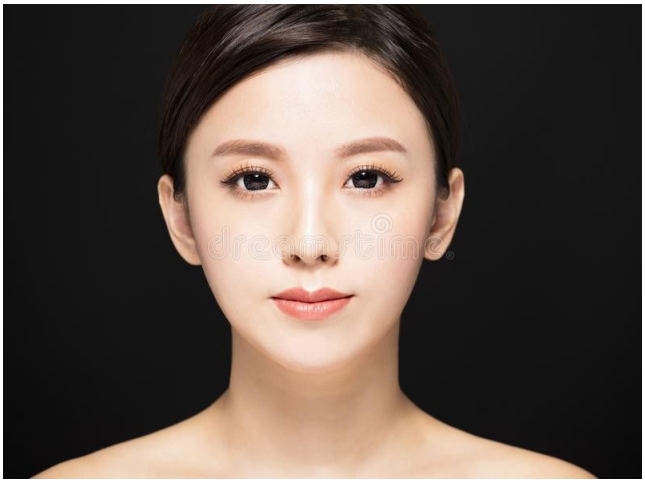
```
        auxiliar1=np.matmul(x_mut,cxi)
        auxiliar2=np.matmul(auxiliar1,x_mu)
        matriznskin[y][x]=(1/((2*p)**1.5))*(1/((det)**0.5))*(math.exp(-0.5*auxiliar2))
        logmatriznskin[y][x]=math.log(matriznskin[y][x],10)
fig=plt.figure()
sns.heatmap(logmatriznskin)
plt.title('log(p(X|W=piel=1))')
plt.show

#----------------------PROBABILIDAD FONDO-----------------------
matriznback=np.zeros((resoluciony,resolucionx))
#Calculo del determinante de la matriz de covarianza
det=np.linalg.det(cxback)
for y in range(resoluciony):
        for x in range(resolucionx):
        (b, g, r) = imagen[y, x]
        rgb=np.array([
        [r],
        [g],
        [b]
        ])
        rgb=rgb/255;
        x_mu=rgb-vmback
        x_mut=transpose(x_mu)
        auxiliar2=np.zeros((1,3))
        cxi=np.linalg.inv(cxback)
        auxiliar1=np.matmul(x_mut,cxi)
        auxiliar2=np.matmul(auxiliar1,x_mu)
        prelog=(1/((2*p)**1.5))*(1/((det)**0.5))*(math.exp(-0.5*auxiliar2))
        matriznback[y][x]=math.log(prelog,10)

fig=plt.figure()
sns.heatmap(matriznback)
plt.title('log(p(X|W=fondo=0))')
plt.show


#----------------------MAPA DE CALOR DE PIEL-----------------------
pskin=expit(matriznskin)
hm=pskin*lskin;
fig=plt.figure()
sns.heatmap(hm)
plt.title('MAPA DE CALOR DE PIEL p(W=1|X)')
plt.show


#----------------------IMAGEN BINARIA 0.4-----------------------
```

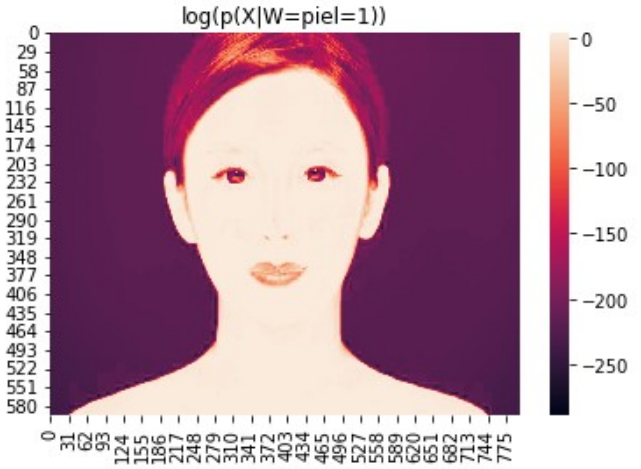
```
ib=np.zeros((resoluciony,resolucionx))
for y in range(resoluciony):
        for x in range(resolucionx):
        if hm[y][x]>=0.4:
                ib[y][x]=1
        else:
                ib[y][x]=0
fig=plt.figure()
sns.heatmap(ib,vmin=0,vmax=1)
plt.title('Threshold=0.4')
plt.show

#-------------------IMAGEN BINARIA 0.5-----------------------
ib=np.zeros((resoluciony,resolucionx))
for y in range(resoluciony):
        for x in range(resolucionx):
        if hm[y][x]>=0.5:
                ib[y][x]=1
        else:
                ib[y][x]=0
fig=plt.figure()
sns.heatmap(ib,vmin=0,vmax=1)
plt.title('Threshold=0.5')
plt.show

#-------------------IMAGEN BINARIA 0.6-----------------------
ib=np.zeros((resoluciony,resolucionx))
for y in range(resoluciony):
        for x in range(resolucionx):
        if hm[y][x]>=0.6:
                ib[y][x]=1
        else:
                ib[y][x]=0
fig=plt.figure()
sns.heatmap(ib,vmin=0,vmax=1)
plt.title('Threshold=0.6')
plt.show
```

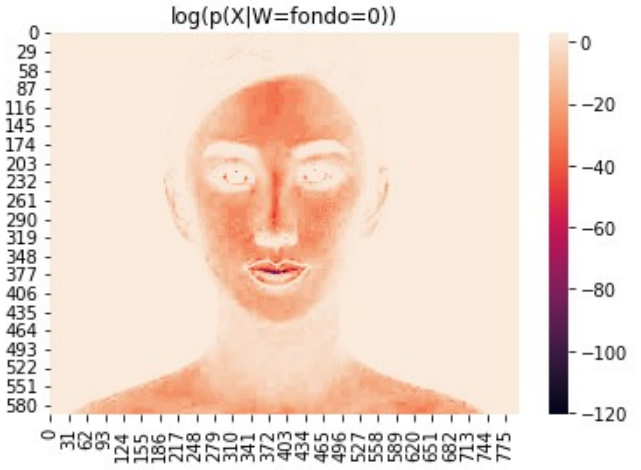
Analisis de las imagenes:

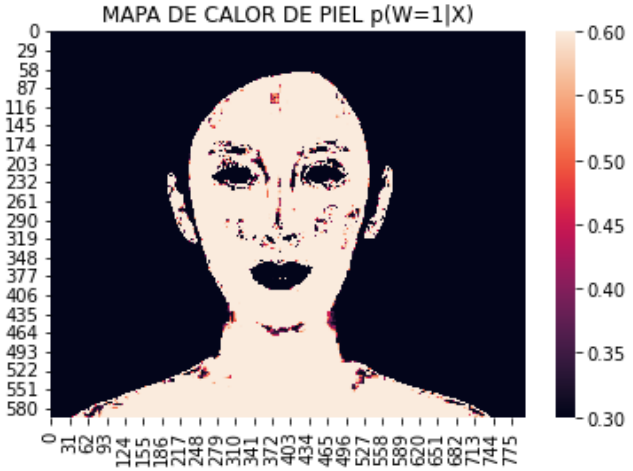
Imagen 1

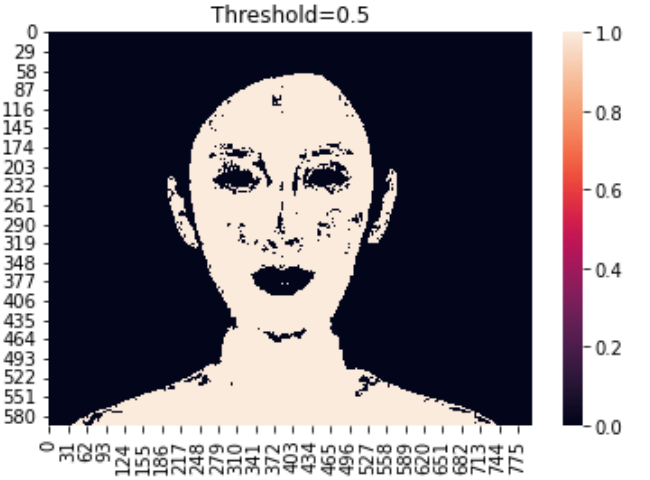
Primero se realizan pruebas con personas de piel blanca y fondo negro obteniendo estos resultados:

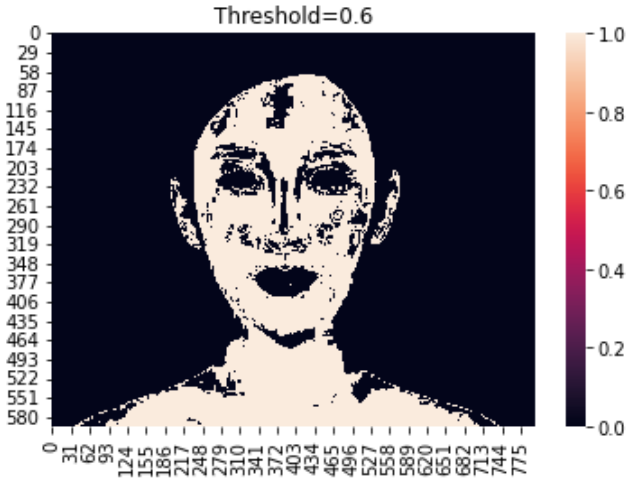












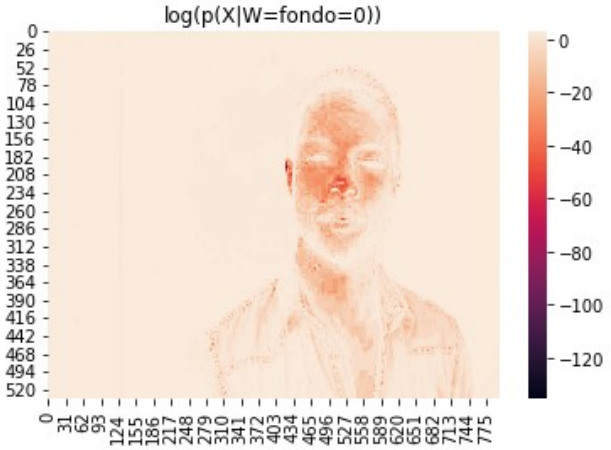
Como se puede observar el umbral que mejor discretiza la piel del fondo es la de T=0.4 y se puede notar que mientras mayor sea el umbral mas pixeles de no piel se van detectando de manera errónea.

El vector promedio de piel es:
[[0.89882353]
[0.77019608]
[0.68705882]]

El vector promedio de fondo es:
[[0.29686275]
[0.22941176]
[0.21098039]]

Imagen 2

Ahora se realizan pruebas con una imagen de piel negra con fondo blanco para ver que tal se comporta, obteniendo los siguientes resultados:



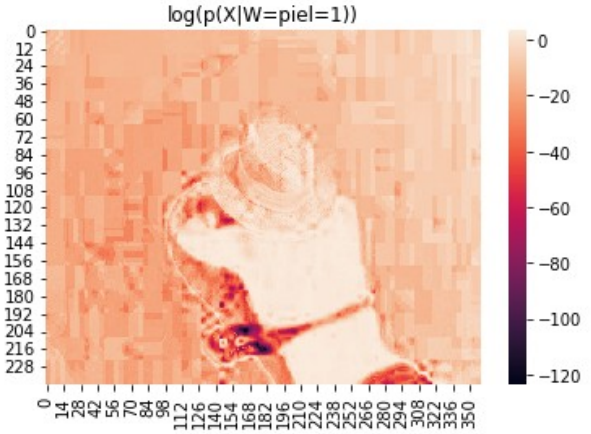
Como se puede ver para pieles negras con fondos blancos también es posible discretizar los pixeles entrenando de nuevo al modelo para esta imagen, siendo el umbral de T=0.4 el que mejor obtiene los datos de la piel.

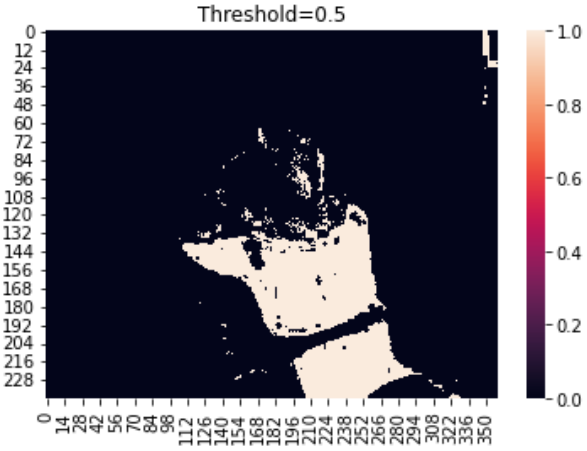
El vector promedio de piel es:
[[0.44784314]
[0.31529412]
[0.29764706]]

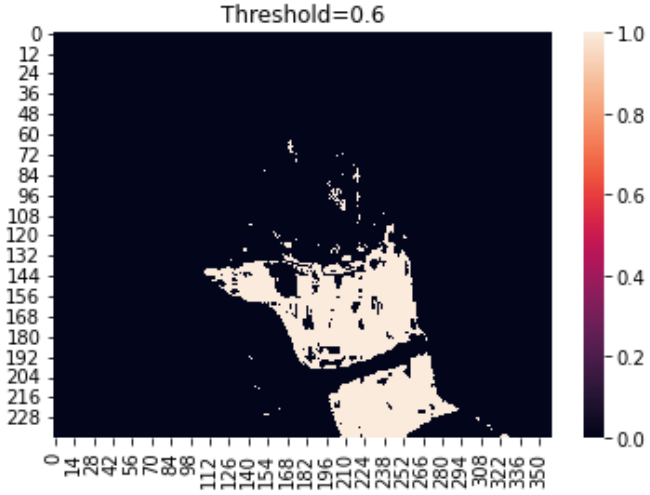
El vector promedio de fondo es:
[[0.6454902]
[0.64901961]
[0.66470588]]

Imagen 3

Ahora se analiza el modelo, con una imagen de una mujer en la playa donde el tono de piel es muy similar al color de la arena, y se obtienen estos resultados:







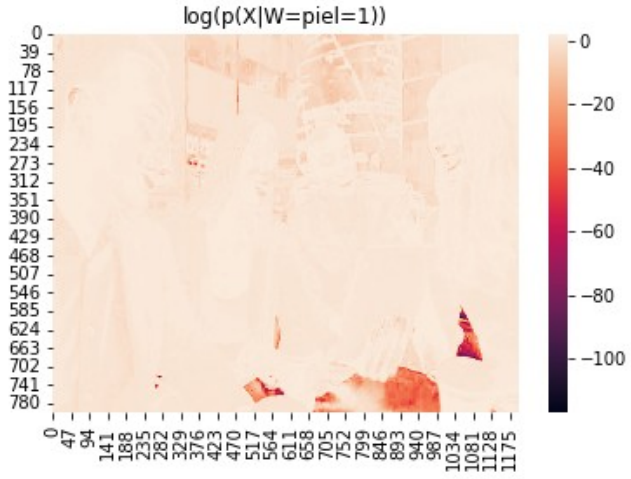
Como se puede observar el modelo si es capaz de discretizar aun cuando el fondo y la piel tienen colores muy similares y el umbral de T=0.4 sigue siendo el que mejor lo discretiza.

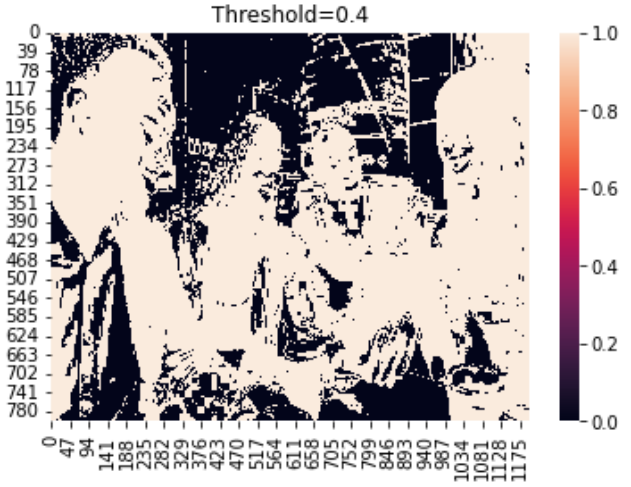
El vector promedio de piel es:
[[0.73686275]
[0.57372549]
[0.48039216]]

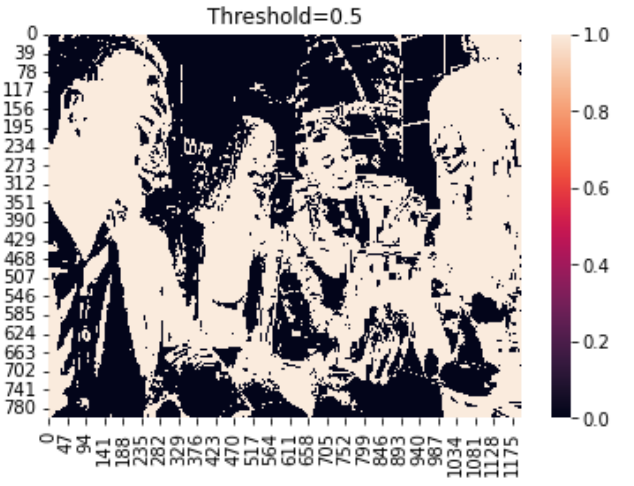
El vector promedio de fondo es:
[[0.70156863]
[0.59843137]
[0.52352941]]

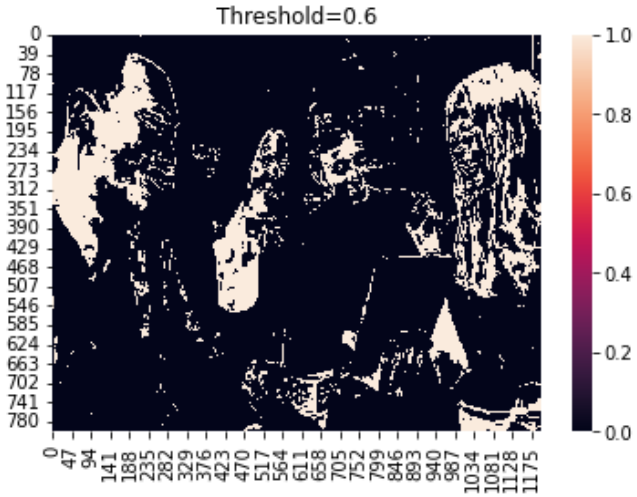
Imagen 4

Para esta imagen se evaluá como se comporta el modelo cuando los colores de la piel son diferentes y el color de fondo es mas variado, estos resultados se obtuvieron:









Como se puede ver, el modelo si es capaz de discretizar de buena manera los diferentes colores de piel en la misma imagen, sin embargo en este caso el mejor un umbral es el de T=0.6, sin embargo el cabello de la mujer de la derecha lo termina considerando en su mayoría como piel.

El vector promedio de piel es:
[[0.65294118]
 [0.51490196]
 [0.4454902]]

El vector promedio de fondo es:
[[0.69333333]
 [0.5545098]
 [0.49803922]]