

Universidad Peruana de Ciencias Aplicadas



INFORME DE CURSO INTELIGENCIA ARTIFICIAL

Carrera de Ciencias de la Computación

Sección: CC54

Alumno	Código
Francesca Nicole Bances Torres	u202214090
Marcos Aaron Bedia Torres	u202118582
Gonzalo Benjamin Huaranga Blanco	u202119611

2024

Índice

Descripción del problema.....	2
Objetivo.....	2
Descripción y visualización del conjunto de datos(dataset).....	2
Propuesta.....	3
Diseño del aplicativo.....	4
CRISP-DM.....	4
1.- Entendimiento de negocio.....	4
2.- Entendimiento de los datos.....	5
3.- Preparación de los datos.....	6
Frontend.....	13
Bibliografía.....	13

Descripción del problema.

De acuerdo a la Dra. Ilda Rodriguez Atanes (s.f) el cabello tiene un impacto significativo en nuestra sociedad, un fenómeno que no es actual, puesto que a lo largo de la historia se expresaba, a través de este, distintas características como condición social, religiosa y particularmente para las mujeres de manera profesional. De acuerdo con estudios sociológicos muchas mujeres se definen por su apariencia externa, esto da lugar a que quieran tener un cabello vistoso y saludable. En 2020, el COVID-19 trajo consigo una época de digitalización notable, pues aproximadamente unas 465.71 millones de personas en toda América latina tiene acceso a internet lo que significa que se usa mucho más las redes sociales. Ante este impacto significativo, muchas mujeres se ven en la necesidad de transmitir una buena imagen de cara a los usuarios de las redes sociales, siendo su cabello una característica importante para ellas. El cabello puede tener variedad de estilos y colores y el no saber con qué estilo quedarse, puede suponer una tarea difícil para las mujeres.

Nuestro equipo busca facilitar el trabajo de búsqueda de estilos en corte de cabello para las damas creando una inteligencia artificial que logre proponer distintos estilos de acuerdo a las medidas y facciones del rostro de una mujer, ya que se debe tener en cuenta que el tipo de rostro van a influir en el tipo de corte o estilo del cabello. Esta inteligencia artificial trabajará con fotos de las personas, y una vez analizado la imagen con el rostro será capaz de poder brindar distintos cortes que van según a la fisiología facial de la persona.

Con este proyecto el equipo prevé que podrá ayudar a una gran cantidad de mujeres, primeramente, aquí en Lima metropolitana y según a los resultados y la acogida, se trabajará a profundidad para que pueda llegar a todo el Perú y en un pronto futuro al globo.

Objetivo

Desarrollar una inteligencia artificial que recomiende estilos de cabello personalizados para mujeres, utilizando análisis de características faciales.

Objetivos Específicos:

- Analizar la estructura facial de cada usuario a través de imágenes.
- Identificar estilos de cabello que complementen características faciales específicas.
- Desarrollar una interfaz accesible que permita a los usuarios interactuar con la IA.

Descripción y visualización del conjunto de datos(dataset).

- Redactar la características y origen de los datos motivo de análisis.
- Realizar un análisis visual de los datos (análisis EDA).
- Seleccionar y fundamentar de forma preliminar cuales características utilizarán para el entrenamiento del algoritmo seleccionado.

Título: Face Shape Dataset

Origen: El dataset fue extraído de Kaggle.

Dataset: <https://www.kaggle.com/datasets/niten19/face-shape-dataset>

Documentos del Dataset:

En dataset encontramos dos carpetas de training_set y testing_set, ambas carpetas contienen cinco subcarpetas, cada una correspondiente a uno de los cinco tipos de rostro

- Oval: Rostro ovalado
- Heart: Rostro con forma de corazón.
- Oblong: Rostro rectangular
- Round: Rostro redondo
- Square: Rostro cuadrado

Características del Dataset:

- **Número de Observaciones:** 5000
- El training_set de cada categoría contiene 800 imágenes, mientras que el testing_set contiene 200 imágenes.

Propuesta

- **Objetivo de la propuesta:**

El objetivo de esta propuesta es desarrollar una solución basada en inteligencia artificial para ayudar a las mujeres a elegir estilos de corte de cabello que mejor se adapten a la fisiología de su rostro. La propuesta se enfoca en utilizar redes neuronales convolucionales (CNN) para analizar imágenes faciales y sugerir cortes de cabello apropiados según las características faciales, como la forma del rostro, la distancia entre los ojos, la mandíbula, y otros atributos relevantes.

- **Técnica y Metodología Utilizada:**

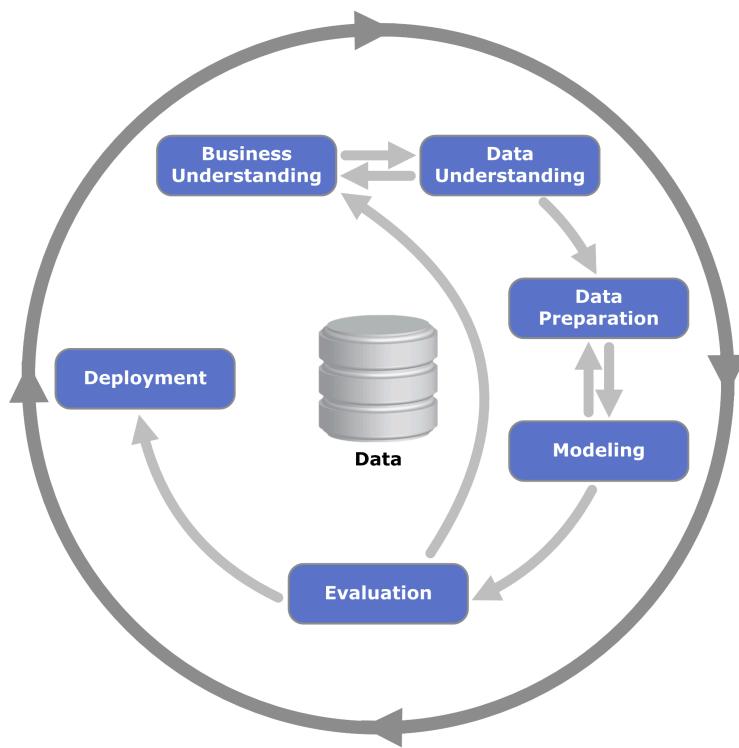
La metodología se centra en el uso de redes neuronales convolucionales (CNN), que son efectivas para el análisis de imágenes y reconocimiento de patrones visuales. Una descripción de los pasos clave del enfoque serían:

1. **Preprocesamiento de Imágenes:** Las imágenes de los rostros serán preprocesadas para mejorar la calidad del análisis. Esto incluirá técnicas de normalización de iluminación, detección de bordes y alineación del rostro para asegurar que los rasgos faciales clave estén correctamente posicionados antes de pasar al modelo.
2. **Representación de Imágenes:** Se utilizarán redes neuronales convolucionales para extraer características faciales de las imágenes. Estas redes serán entrenadas para identificar atributos como la forma de la mandíbula, la frente, y otros aspectos faciales importantes que influyen en la selección del corte de cabello.
3. **Entrenamiento de la Red Neuronal:** El modelo se entrenará utilizando un conjunto de datos de imágenes de rostros etiquetados con estilos de cortes de cabello recomendados según la forma del rostro. Se utilizará el *CelebFaces Attributes Dataset (CelebA)*, que contiene imágenes de

- celebridades con una variedad de atributos faciales y características etiquetadas que serán útiles para este proyecto.
4. **Selección de Estilos de Cabello:** Una vez que el modelo esté entrenado, tomará una imagen facial como entrada y propondrá una lista de estilos de corte de cabello recomendados para esa persona. El modelo tendrá en cuenta los rasgos faciales extraídos y sugerirá estilos que resaltan o complementen la fisiología de cada persona.
 5. **Interfaz de Usuario y Presentación de Resultados:** El sistema final incluirá una interfaz de usuario amigable donde las mujeres podrán subir sus fotos y recibir sugerencias personalizadas de cortes de cabello basadas en su rostro. Las recomendaciones serán presentadas de manera clara, incluyendo ejemplos visuales de los estilos sugeridos.

Diseño del aplicativo

CRISP-DM



1.- Entendimiento de negocio

El objetivo es desarrollar una solución de inteligencia artificial para ayudar a mujeres a encontrar estilos de corte de cabello que se adapten a sus características faciales. Esto surge de la necesidad social e individual de mantener una buena apariencia en la era de la digitalización, donde la imagen es importante, especialmente en redes sociales. El proyecto apunta a mejorar la confianza y satisfacción personal al facilitar la elección de estilos de cabello de forma personalizada.

El enfoque de negocio se centra en:

1. **Problema identificado:** Las mujeres enfrentan dificultades al elegir un estilo de cabello que complemente sus facciones.
2. **Impacto social y objetivo:** Mejorar la experiencia de elección de estilos de cabello mediante una IA que ofrezca sugerencias basadas en análisis de imágenes, inicialmente en Lima Metropolitana, con la proyección de escalar a nivel nacional y potencialmente internacional.
3. **Beneficios esperados:** Ayudar a una gran cantidad de usuarias a ahorrar tiempo y obtener resultados estéticamente satisfactorios, mejorando la imagen y confianza de cara a las interacciones en redes sociales.

2.- Entendimiento de los datos

Estamos tomando imágenes de un Dataset llamado Face Shape Dataset, dentro de la carpeta descargada hay 5 subcarpetas “Heart”, “Oblong”, “Oval”, “Round” y “Square” en las cuales hay 800 en cada. Además, las imágenes son de tamaños distintos.

VERIFICACIÓN DEL TAMAÑO DE IMAGEN Y LA CANTIDAD INICIAL:

Subcarpeta(Heart): Se tiene como cantidad 800 imágenes, en este caso presentaremos un ejemplo de todo el database.



286 x 350



351 x 447

Como podemos ver se ve una clara diferencia de tamaño de las imágenes.

3.- Preparación de los datos

Para tener los datos preparados hemos decidido realizar 2 pasos:

- Eliminar las imágenes duplicadas
- Eliminar las imágenes corruptas (archivos y formatos diferentes a las imágenes pensadas).

```
# Ruta de la carpeta principal que contiene las subcarpetas de imágenes
carpeta_principal = "C:/Inteligencia_Artificial/Trabajo_Final/facetype/FaceShape Dataset/training_set"
subcarpetas = ["Heart", "Oblong", "Oval", "Round", "Square"]

# Función para cargar y verificar imágenes
def cargar_imagen(ruta):
    try:
        img = Image.open(ruta)
        img.verify() # Verifica si la imagen está corrupta
        img = Image.open(ruta).convert("RGB") # Convierte a RGB si está en otro modo
        return img
    except Exception as e:
        print(f"Imagen corrupta o no válida: {ruta}, error: {e}")
        return None

# Función para calcular un hash para detectar duplicados
def calcular_hash(imagen):
    arr = np.array(imagen).flatten()
    return hash(arr.tobytes())
```

```

# Procesar cada subcarpeta y actualizar las imágenes
for subcarpeta in subcarpetas:
    ruta_subcarpeta = os.path.join(carpeta_principal, subcarpeta)
    ruta_subcarpeta_temp = os.path.join(carpeta_principal, f"{subcarpeta}_temp")

    # Crear una carpeta temporal para almacenar imágenes procesadas
    os.makedirs(ruta_subcarpeta_temp, exist_ok=True)

    imagenes_procesadas = {}

    # Verifica que la subcarpeta exista antes de procesar
    if not os.path.exists(ruta_subcarpeta):
        print(f"La carpeta {ruta_subcarpeta} no existe.")
        continue

    for archivo in os.listdir(ruta_subcarpeta):
        ruta_imagen = os.path.join(ruta_subcarpeta, archivo)

        # Cargar y verificar la imagen
        imagen = cargar_imagen(ruta_imagen)
        if imagen is None:
            continue # Saltar imágenes malas

        # Calcular hash para detectar duplicados
        hash_imagen = calcular_hash(imagen)

        if hash_imagen not in imagenes_procesadas:
            imagenes_procesadas[hash_imagen] = ruta_imagen
            # Guardar imagen única en la carpeta temporal
            imagen.save(os.path.join(ruta_subcarpeta_temp, archivo))
        else:
            print(f"Imagen duplicada encontrada y eliminada: {ruta_imagen}")

```

```

# Eliminar la carpeta original y renombrar la temporal
shutil.rmtree(ruta_subcarpeta) # Elimina la carpeta original
os.rename(ruta_subcarpeta_temp, ruta_subcarpeta) # Renombra la carpeta temporal

print(f"Carpeta {subcarpeta} actualizada con imágenes únicas y válidas.")

```

Después de la ejecución comprobamos las imágenes eliminadas.

```

Imagen duplicada encontrada y eliminada: C:/Inteligencia_Artificial/Trabajo_Final/facetype/FaceShape Dataset/training_set\Heart\heart (545).jpg
Imagen corrupta o no válida: C:/Inteligencia_Artificial/Trabajo_Final/facetype/FaceShape Dataset/training_set\Heart\heart (633).jpg, error: image file is truncated (8 bytes not process
Carpeta Heart actualizada con imágenes únicas y válidas.
Imagen duplicada encontrada y eliminada: C:/Inteligencia_Artificial/Trabajo_Final/facetype/FaceShape Dataset/training_set\Oblong\oblong (639).jpg
Carpeta Oblong actualizada con imágenes únicas y válidas.
Imagen duplicada encontrada y eliminada: C:/Inteligencia_Artificial/Trabajo_Final/facetype/FaceShape Dataset/training_set\Oval\oval (584).jpg
Imagen duplicada encontrada y eliminada: C:/Inteligencia_Artificial/Trabajo_Final/facetype/FaceShape Dataset/training_set\Oval\oval (721).jpg
Carpeta Oval actualizada con imágenes únicas y válidas.
Carpeta Round actualizada con imágenes únicas y válidas.
Imagen corrupta o no válida: C:/Inteligencia_Artificial/Trabajo_Final/facetype/FaceShape Dataset/training_set\Square\square (84).jpg, error: image file is truncated (0 bytes not proces
Carpeta Square actualizada con imágenes únicas y válidas.

```

Visualización de imágenes

Definimos las rutas a las carpetas que contienen los conjuntos de imágenes para entrenamiento y prueba. Además se establece el tamaño deseado para redimensionar las imágenes, en este caso, **100x100 píxeles**.

```

# Ruta a las carpetas
DATASET_TRAINING_DIR = "C:/códigos/TF_IA/FaceShape Dataset/training_set"
DATASET_TESTING_DIR = "C:/códigos/TF_IA/FaceShape Dataset/testing_set"

Tamaño_IMG=(100,100)

```

A continuación convertimos las imágenes a tensores, normalizando los valores de píxeles al rango [0, 1] y las redimensionamos a 100x100 píxeles.

```

# Transformaciones para preprocesar las imágenes
transform = transforms.Compose([
    transforms.Resize(Tamaño_IMG),      # Redimensionar imágenes
    transforms.ToTensor(),
])

```

```

# Cargar el conjunto de datos con ImageFolder sin transformaciones
dataset = ImageFolder(root=DATASET_TRAINING_DIR, transform=transform)

# Lista para almacenar los pares (imagen, etiqueta)
datos_entrenamiento = []
for image, label in dataset: # Iterar sobre todas las imágenes y etiquetas
    # Convertir tensor a NumPy y ajustar dimensiones
    image_np = image.numpy().transpose(1, 2, 0) # De (1, 100, 100) a (100, 100,
    datos_entrenamiento.append([image_np, label])
random.shuffle(datos_entrenamiento)

```

Visualizamos las imágenes.



Comprobamos las dimensiones de la imagen después del procesamiento. Como podemos ver ahora tiene un tamaño de 100x100 píxeles y tiene tres canales de color, lo que corresponde al formato RGB.

```
print('Size image: ' + str(image_np.shape))
```

```
Size image: (100, 100, 3)
```

Comprobamos que las imágenes se hayan normalizado correctamente. Los valores de los píxeles se encuentran entre 0 y 1, lo que significa que la normalización se realizó con éxito.

```
datos_entrenamiento[0]
```

```
[array([[[0.00392157, 0.00392157, 0.00392157],  
        [0.00392157, 0.00392157, 0.00392157],  
        [0.00392157, 0.00392157, 0.00392157],  
        ...,  
        [0.00392157, 0.00392157, 0.00392157],  
        [0.00392157, 0.00392157, 0.00392157],  
        [0.00392157, 0.00392157, 0.00392157]],  
  
       [[0.00392157, 0.00392157, 0.00392157],  
        [0.00392157, 0.00392157, 0.00392157],  
        [0.00392157, 0.00392157, 0.00392157],  
        ...,  
        [0.00392157, 0.00392157, 0.00392157],  
        [0.00392157, 0.00392157, 0.00392157],  
        [0.00392157, 0.00392157, 0.00392157]],  
  
       [[0.00392157, 0.00392157, 0.00392157],  
        [0.00392157, 0.00392157, 0.00392157],  
        [0.00392157, 0.00392157, 0.00392157],  
        ...,  
        [0.00392157, 0.00392157, 0.00392157],  
        [0.00392157, 0.00392157, 0.00392157],  
        [0.00392157, 0.00392157, 0.00392157]],  
  
       ...  
      ...]
```

Entrenamiento del modelo:

Creación del modelo

- Conv2D (32 filtros, tamaño 3x3): Aplica convoluciones a las imágenes de entrada con una activación ReLU.
- MaxPooling2D (2x2): Reduce la dimensionalidad de las características mediante una operación de submuestreo, extrayendo las características más destacadas.
- Conv2D (64 filtros, tamaño 3x3): Aplica otra convolución, aumentando la cantidad de filtros para extraer características más complejas.
- MaxPooling2D (2x2): Nuevo submuestreo para reducir el tamaño de las características.
- Conv2D (128 filtros, tamaño 3x3): Otra convolución, con más filtros.

- MaxPooling2D (2x2): Reducción adicional del tamaño de las características.
- Dropout (0.5): Se aplica una tasa de abandono (50%) para evitar el sobreajuste, desconectando aleatoriamente algunas neuronas durante el entrenamiento.
- Flatten: Convierte la salida de las capas convolucionales en un vector unidimensional.
- Dense (100 unidades): Capa completamente conectada con 100 unidades y activación ReLU.
- Dense (5 unidades, activación softmax): Capa de salida con 5 unidades (una para cada clase) y activación softmax para obtener probabilidades de las 5 clases.

```
# Modelo CNN con Dropout para 5 clases
modeloCNN2 = tf.keras.models.Sequential([
    tf.keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(100, 100, 3)),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(64, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Conv2D(128, (3,3), activation='relu'),
    tf.keras.layers.MaxPooling2D(2, 2),
    tf.keras.layers.Dropout(0.5),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(100, activation='relu'),
    tf.keras.layers.Dense(5, activation='softmax') # Cambiado a 5 salidas
])
```

Dividimos los datos en entrenamiento y validación para poder comenzar con el entrenamiento del modelo

```
from sklearn.model_selection import train_test_split

# Dividir los datos en entrenamiento (80%) y validación (25%)
x_train, x_val, y_train, y_val = train_test_split(x, y, test_size=0.25, random_state=42)
```

Entrenamos el modelo y utilizando:

- Optimizador Adam para disminuir la función de pérdida durante el entrenamiento.
- sparse_categorical_crossentropy, que establece la función de pérdida como entropía cruzada categórica. Se utiliza cuando las etiquetas están en formato entero.
- metrics=['accuracy']: Indica que se medirá la precisión durante el entrenamiento para evaluar el rendimiento del modelo.

```
modeloCNN2.compile(optimizer='adam',
                    loss='sparse_categorical_crossentropy', # O categorical_crossentropy si usas one-hot
                    metrics=['accuracy'])

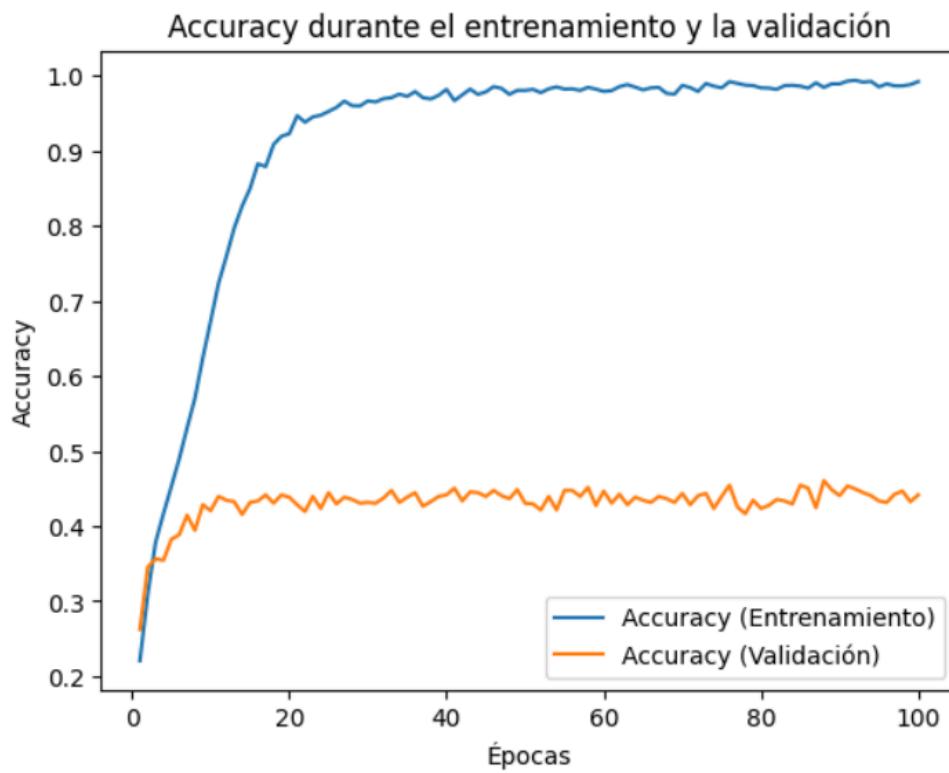
historyCNN2 = modeloCNN2.fit(x_train, y_train,
                             epochs=100,
                             validation_data=(x_val, y_val),
                             batch_size=32)
```

Entrenamos el modelo durante 100 épocas.

```

Epoch 1/100
94/94 8s 68ms/step - accuracy: 0.2131 - loss: 1.6217 - val_accuracy: 0.2623 - val_loss: 1.5917
Epoch 2/100
94/94 6s 64ms/step - accuracy: 0.2891 - loss: 1.5547 - val_accuracy: 0.3453 - val_loss: 1.5254
Epoch 3/100
94/94 6s 63ms/step - accuracy: 0.3719 - loss: 1.4707 - val_accuracy: 0.3564 - val_loss: 1.4932
Epoch 4/100
94/94 6s 63ms/step - accuracy: 0.4190 - loss: 1.4130 - val_accuracy: 0.3544 - val_loss: 1.4866
Epoch 5/100
94/94 6s 64ms/step - accuracy: 0.4507 - loss: 1.3362 - val_accuracy: 0.3824 - val_loss: 1.4623
Epoch 6/100
94/94 7s 73ms/step - accuracy: 0.4894 - loss: 1.2553 - val_accuracy: 0.3884 - val_loss: 1.4352
Epoch 7/100
94/94 7s 73ms/step - accuracy: 0.5353 - loss: 1.1602 - val_accuracy: 0.4144 - val_loss: 1.4204
Epoch 8/100
94/94 7s 73ms/step - accuracy: 0.5890 - loss: 1.0549 - val_accuracy: 0.3944 - val_loss: 1.4915
Epoch 9/100
94/94 7s 73ms/step - accuracy: 0.6317 - loss: 0.9630 - val_accuracy: 0.4284 - val_loss: 1.4734
Epoch 10/100
94/94 7s 73ms/step - accuracy: 0.6874 - loss: 0.8259 - val_accuracy: 0.4204 - val_loss: 1.5051
Epoch 11/100
94/94 6s 68ms/step - accuracy: 0.7353 - loss: 0.7120 - val_accuracy: 0.4394 - val_loss: 1.5676
Epoch 12/100
94/94 6s 66ms/step - accuracy: 0.7750 - loss: 0.6043 - val_accuracy: 0.4344 - val_loss: 1.7273
Epoch 13/100
...
Epoch 99/100
94/94 6s 64ms/step - accuracy: 0.9913 - loss: 0.0295 - val_accuracy: 0.4324 - val_loss: 4.5194
Epoch 100/100
94/94 6s 64ms/step - accuracy: 0.9903 - loss: 0.0317 - val_accuracy: 0.4414 - val_loss: 4.3542
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

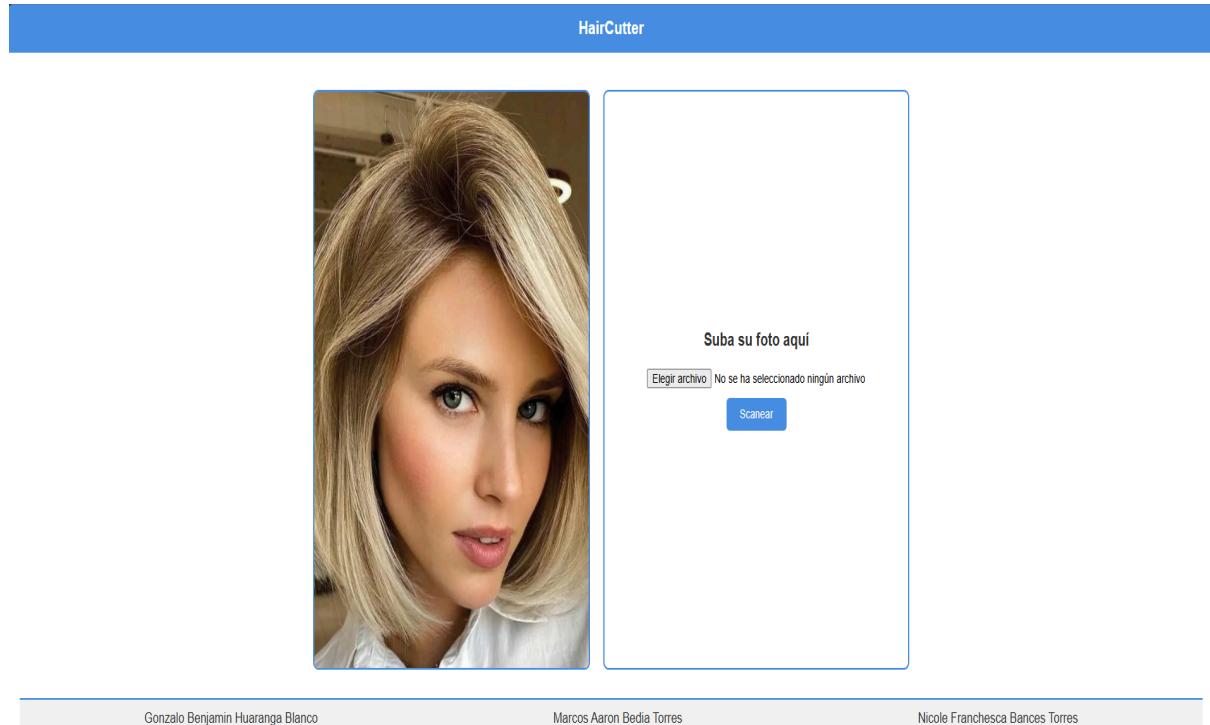
```



Analizando los resultados podemos ver que el modelo es bastante preciso con los datos de entrenamiento pero al probarlo con los datos de validación, no es muy exacto.

Frontend

Esta imagen es lo que pensamos de cómo sería nuestra página web.



Gonzalo Benjamin Huaranga Blanco

Marcos Aaron Bedia Torres

Nicole Franchesca Bances Torres

Bibliografía

Atanes, I. R. (2010, noviembre 19). La importancia social del pelo: dime cómo llevas tu pelo y te diré quién eres. *Svenson Soluciones Capilares*.

<https://www.svenson.es/blog/la-importancia-social-del-pelo-dime-como-llevas-tu-pelo-y-te-dire-quien-eres/>

Redes sociales, Dark Factor y bienestar en jóvenes peruanos. (s/f). Sistema de Gestión de la Información sobre la Investigación (CRIS Ulima). Recuperado el 23 de septiembre de 2024, de

<https://cris.ulima.edu.pe/es/projects/redes-sociales-dark-factor-y-bienestar-en-j%C3%BAvenes-peruanos>

¿Qué son las redes neuronales convolucionales? (2023, julio 5). *Ibm.com.*

<https://www.ibm.com/es-es/topics/convolutional-neural-networks>