

BEAT Buddy



Por:
Christian Martín Díaz,
Pablo Martín Trujillo y
María Eugenia Sánchez Sánchez

índice

Introducción

Asistente musical

Reconocimiento facial

Voicebot

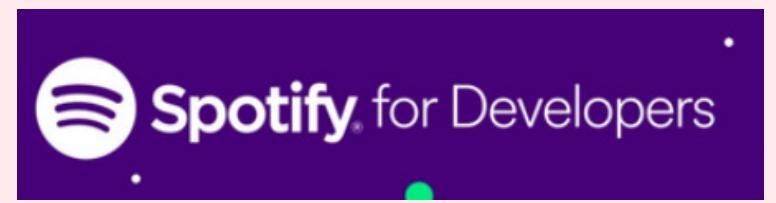
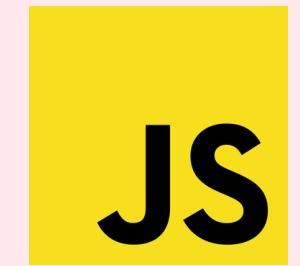
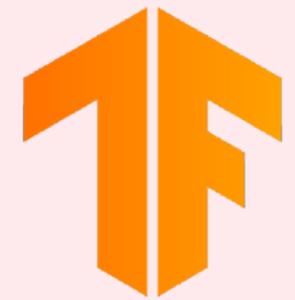
Conclusión

INTRODUCCIÓN

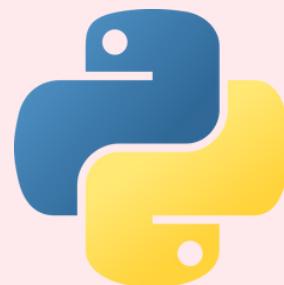
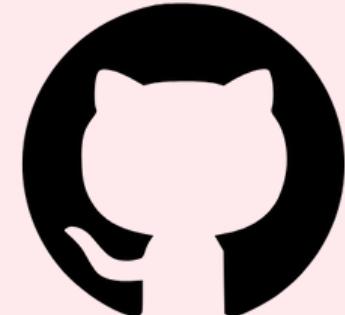
BeatBuddy te guía para descubrir tu canción ideal a través de un asistente musical. Además, con el reconocimiento facial, te recomienda una canción acorde a tu estado de ánimo. También tiene integrado un voicebot, en el cuál podrás realizarle todas las preguntas musicales que se te ocurran.

TECNOLOGÍAS

Gemini



Whisper



pandas



matplotlib

ASISTENTE MUSICAL

SCRAPPING

PREPARACIÓN
DATOS

WEB

EXPLORACIÓN Y
VISUALIZACIÓN

ENTRENAMIENTO

SCRAPPING

RECOLECCIÓN DE PLAYLISTS

```
In [ ]: client_credentials_manager = SpotifyClientCredentials(client_id=client_id, client_secret=client_secret)
sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager)
```

Emoción: Triste

Sacamos los **metadatos** de las canciones de una playlist.

```
In [ ]: playlist_id = '37i9dQZF1DXdZjf8WgcTKM'

try:
    playlist_tracks = sp.playlist_items(playlist_id)
except:
    print("No se ha encontrado la playlist")
```

Guardamos en un array valores que nos pueden ser interesantes para listar las canciones como lo son el ID, el nombre(meramente informativo) y la fecha de salida de la canción.

```
In [ ]: canciones = []

for tracks in playlist_tracks['items']:
    canciones.append([tracks['track']['id'], tracks['track']['name'], tracks['track']['album']['release_date']])

for cancion in canciones:
    print(f'Nombre de la cancion: {cancion[1]}; ID de la cancion: {cancion[0]}; Fecha de lanzamiento: {cancion[2]}')
```

```
Nombre de la cancion: Heather; ID de la cancion: 4xqrdFxkTW4T0RauPLv3WA; Fecha de lanzamiento: 2020-03-20
Nombre de la cancion: The Scientist; ID de la cancion: 75JFvkI2RXiU7L9VXzMkle; Fecha de lanzamiento: 2002-08-08
Nombre de la cancion: Let Me Down Slowly; ID de la cancion: 2qxmye6gAegTMjLKEBoR3d; Fecha de lanzamiento: 2018-11-16
```

SPOTIFY - GET PLAYLIST ITEMS



Get Several Albums

Get Album Tracks

Get User's Saved Albums

Save Albums for Current User

Remove Users' Saved Albums

Check User's Saved Albums

Get New Releases

▶ Artists

▶ Audiobooks

▼ Categories

Get Several Browse Categories

Get Single Browse Category

▶ Chapters

▶ Episodes

▶ Genres

Web API • References / Playlists / Get Playlist

Get Playlist Items

Get full details of the items of a playlist owned by

Authorization scopes

▶ playlist-read-private

Request

▼ GET /playlists/{playlist_id}/tracks

playlist_id string Required

The Spotify ID of the playlist.

Example: 3cEYpjA9oz9GiPac4AsH4n

SCRAPPING

VALORES DATASET

```
import pandas as pd

parametros_tracks_playlist = []

try:
    for cancion in canciones:
        parametros = sp.audio_features(cancion[0])
        if parametros:
            parametros_tracks_playlist.append(parametros[0])
except:
    print("Ha habido un error")

# Crear DataFrame
df = pd.DataFrame(parametros_tracks_playlist)
df['release_date'] = [cancion[2] for cancion in canciones] # Añade la fecha de salida de la canción

# Agregar la columna "mood" con el valor "sad"
df['mood'] = 'sad'

# Visualizar el DataFrame
df
```

	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	type
0	0.357	0.425	5	-7.301	1	0.0333	0.584	0.000000	0.3220	0.270	102.078	audio_features
1	0.557	0.442	5	-7.224	1	0.0243	0.731	0.000015	0.1100	0.213	146.277	audio_features
2	0.652	0.557	1	-5.714	0	0.0318	0.740	0.000000	0.1240	0.483	150.073	audio_features

SPOTIFY - GET TRACKS A. FEATURES

The screenshot shows the Spotify for Developers API documentation. The main title is "SPOTIFY - GET TRACKS A. FEATURES". Below it, there's a "Request" section for a "GET /audio-features/{id}" endpoint. The "id" parameter is described as a required string representing the Spotify ID for the track. An example value is provided: "11dFghVXANM1KmJXsNCbN1". To the left, a sidebar lists various API endpoints: Audiobooks, Categories, Chapters, Episodes, Genres, Markets, Player, and Playlists, with "Playlists" expanded to show "Get Playlist", "Change Playlist Details", and "Get Playlist Items".

EXPLORACIÓN VISUALIZACIÓN

DUPPLICADOS

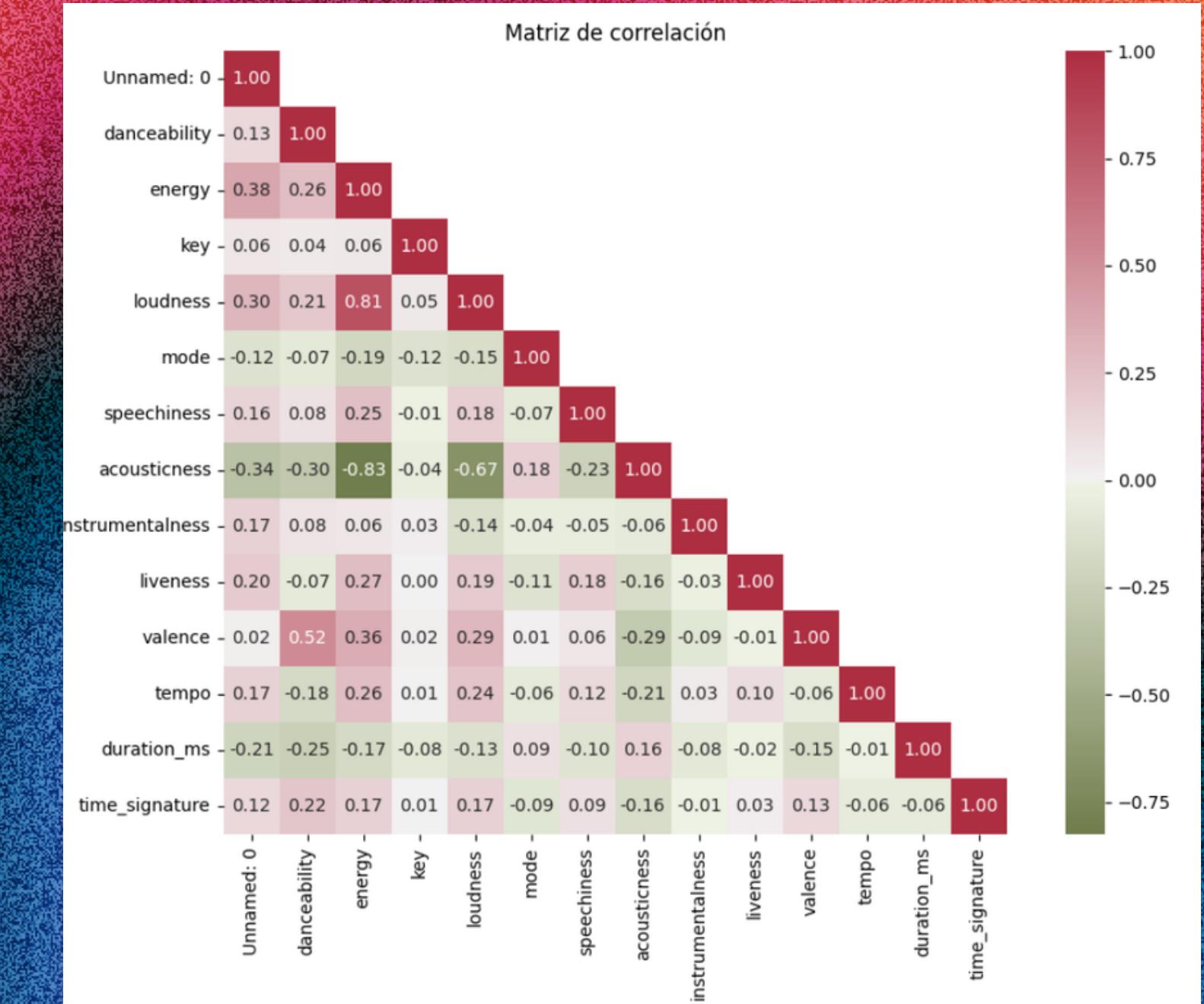
Visualización de datos duplicados

Mostramos las columnas **duplicadas**

```
In [ ]: # Filas duplicadas
filas_duplicadas_id = data[data.duplicated(subset='id', keep=False)].count()
print("Filas duplicadas en 'id':")
print(filas_duplicadas_id)
```

Filas duplicadas en 'id':
Unnamed: 0 137

MATRIZ DE CORRELACIÓN



+81% LOUDNESS - ENERGY

-83% ACOUSTIC - ENERGY

+52% DANCEABILITY - VALENCE

PREPARACIÓN DE LOS DATOS

ELIMINACIÓN DE DUPLICADOS

ELIMINACIÓN DE COLUMNAS

Eliminamos las 137 filas duplicadas quedándonos con la primera aparición de ellos y mostramos las restantes.

In []: `data = data.drop_duplicates(subset="id",keep="first")`

Eliminación de columnas innecesarias para el modelo.

Hemos decidido eliminar una serie de columnas que no guardan mucha correlación con el entrenamiento del modelo en absoluto, o no son relevantes.

Éstas son:

`Unnamed:0 ,key ,duration_ms ,analysis_url ,mode ,track_href ,time_signature ,key ,time_signature ,liveness ,uri ,type .`

`data.drop(columns = ['Unnamed: 0','key','duration_ms','analysis_url','mode','track_href','time_signature','key','time_signature','liveness','uri','type'])`

PREPARACIÓN DE LOS DATOS

REAJUSTE DE VALORES

ELIMINACIÓN DE OUTLIERS

Para continuar, vamos a hacer una clasificación de las canciones por décadas en lugar de por años, para ello, hemos creado la siguiente **expresión regular** capaz de hacerlo.

```
data['release_decade'] = data['release_date'].str.extract(r'(\d{4})').astype(int) // 10 * 10
data['release_decade'] = data['release_decade'].astype(str) + 's'
data
```

Vamos a tomar como referencia las **canciones a partir de la década de los 80** en adelante.

```
In [ ]: data['release_decade'].unique()
```

```
Out[ ]: array(['2020s', '2000s', '2010s', '1990s', '1980s', '1970s', '1940s',
   '1960s'], dtype=object)
```

```
In [ ]: data.drop(data[data['release_decade'].isin(['1940s', '1960s', '1970s'])].index, inplace=True)
```

ENTRENAMIENTO

COLUMNA MOOD

```
In [ ]: from sklearn.linear_model import LogisticRegression

X_train, X_test, y_train, y_test = train_test_split(data.drop(['mood','motivation','id','release_decade'], axis=1)

# Creamos un modelo Lr con el kernel elegido
lr_model = LogisticRegression()

# Entrenamos el modelo
lr_model.fit(X_train, y_train)

# Predecimos las etiquetas para el conjunto de datos de prueba
y_pred = lr_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

# Imprimimos la precisión
print("Precisión:", accuracy)
# Imprimimos el informe de clasificación
print(classification_report(y_test, y_pred))

Precisión: 0.9937106918238994
      precision    recall  f1-score   support
          0.0       0.99     1.00     0.99      92
          1.0       1.00     0.99     0.99      67
  accuracy                           0.99     159
  macro avg       0.99     0.99     0.99     159
weighted avg       0.99     0.99     0.99     159
```

LINEAR REGRESSION: 99,3%

COLUMN MOTIVATION

```
In [ ]: from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report

X_train, X_test, y_train, y_test = train_test_split(data.drop(['mood','motivation','id','release_decade'], axis=1)

# Creamos un modelo LinearSVC con el kernel elegido
LinearSVC_model = LinearSVC()

# Entrenamos el modelo
LinearSVC_model.fit(X_train, y_train)

# Predecimos las etiquetas para el conjunto de datos de prueba
y_pred = LinearSVC_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)

# Imprimimos la precisión
print("Precisión:", accuracy)
# Imprimimos el informe de clasificación
print(classification_report(y_test, y_pred))

/shared-libs/python3.11/py/lib/python3.11/site-packages/sklearn/svm/_classes.py:32: FutureWarning: The default value
` will change from `True` to ``auto`` in 1.5. Set the value of `dual` explicitly to suppress the warning.
warnings.warn(
Precisión: 0.9811320754716981
```

LINEARSVC: 98%

FORMULARIO SELECCIÓN

```
# Selección de ánimo (mood)
preg1 = image_select(
    label="",
    images=[
        "https://cdn-icons-png.flaticon.com/512/1001/1001214.png",
        "https://cdn-icons-png.flaticon.com/512/7457/7457314.png"
    ],
    captions=["Tristeza", "Felicidad"],
    return_value="index",
    key="0"
)
selected_mood = preg1

# Selección de motivación (motivation)
st.markdown(pregunta2, unsafe_allow_html=True)
preg2 = image_select(
    label="",
    images=[
        "https://cdn-icons-png.flaticon.com/512/1142/1142699.png",
        "https://cdn-icons-png.flaticon.com/512/3221/3221947.png"
    ],
    captions=["Relax", "Energía"],
    return_value="index",
    key="1"
)
selected_motivation = preg2

# Selección de década (release_decade)
st.markdown(pregunta3, unsafe_allow_html=True)
decade_options = ['1980s', '1990s', '2000s', '2010s', '2020s']
selected_decade = st.selectbox("", decade_options)
```

LÓGICA SELECCIÓN

```
# Botón submit
if st.button("¡Sorpréndeme!"):
    # Filtramos el DataFrame con las selecciones del usuario
    filtered_songs = processed_songs[
        (processed_songs['mood'] == selected_mood) &
        (processed_songs['motivation'] == selected_motivation) &
        (processed_songs['release_decade'] == selected_decade)
    ]

    # Mostramos 3 canciones aleatorias que cumplen con las características seleccionadas por el usuario
    if len(filtered_songs) >= 3:
        selected_songs = filtered_songs.sample(3)
        st.markdown(textofinal, unsafe_allow_html=True)
        # Generar iframes de Spotify con los ID de las canciones
        for index, song in selected_songs.iterrows():
            iframe_code = f"""
                <iframe id="cancionesAsis" style="border-radius:12px; background-color: transparent;" 
                src="https://open.spotify.com/embed/track/{song['id']}?utm_source=generator"
                width="100%" height="352" frameBorder="0" allowfullscreen="" allow="autoplay; clipboard-write;
                encrypted-media; fullscreen; picture-in-picture" loading="lazy"></iframe>
            """
            st.write(iframe_code, unsafe_allow_html=True)
    else:
        st.write("Lo siento, no hay suficientes canciones que cumplan con esas características.")
```

WEB

Asistente musical 🎵

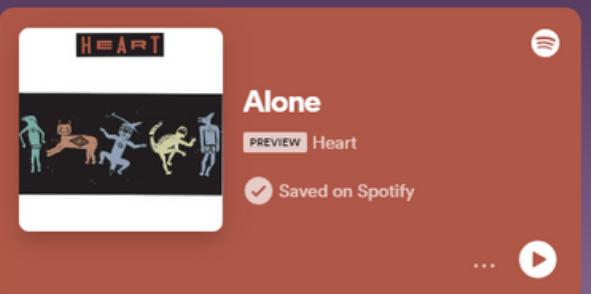
Contesta a las siguientes preguntas para poder recibir 3 recomendaciones de canciones acorde a tu ánimo y tus gustos musicales.

PREGUNTA 1
¿Qué ánimo te representa más hoy?

Tristeza Felicidad

PREGUNTA 2
¿Prefieres estar más tranquilo o motivarte?

¡Aquí tienes tres canciones que podrían sorprenderte!



RECONOCIMIENTO FACIAL

OBTENCIÓN
DATOS

PREPARACIÓN
DATOS

WEB

EXPLORACIÓN Y
VISUALIZACIÓN

ENTRENAMIENTO

OBTENCIÓN DATOS

DATASET KAGGLE

Facial Expression Recognition(FER)Challenge

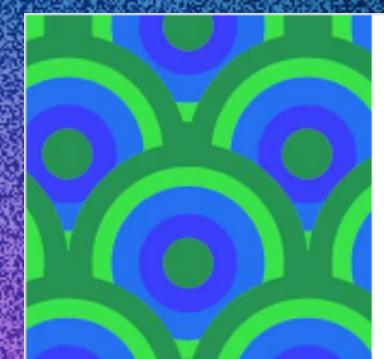
Data Card Code (79) Discussion (1) Suggestions (0)

fer2013.csv (301.07 MB)

Detail Compact Column

# emotion	pixels	Usage	Count
0	34034 unique values	Training	80%
1		PublicTest	10%
2		Other (3589)	10%

0 70 80 82 72 58 58 60
63 54 58 60 48 89
115 121 119 115 110
98 91 84 84 90 99
110 126 143 153 158
171...



Facial Expression Recognition(FER)Challenge

Kaggle is the world's largest data science community with powerful tools and resources to...

[kaggle.com](https://www.kaggle.com)

EXPLORACIÓN Y VISUALIZACIÓN

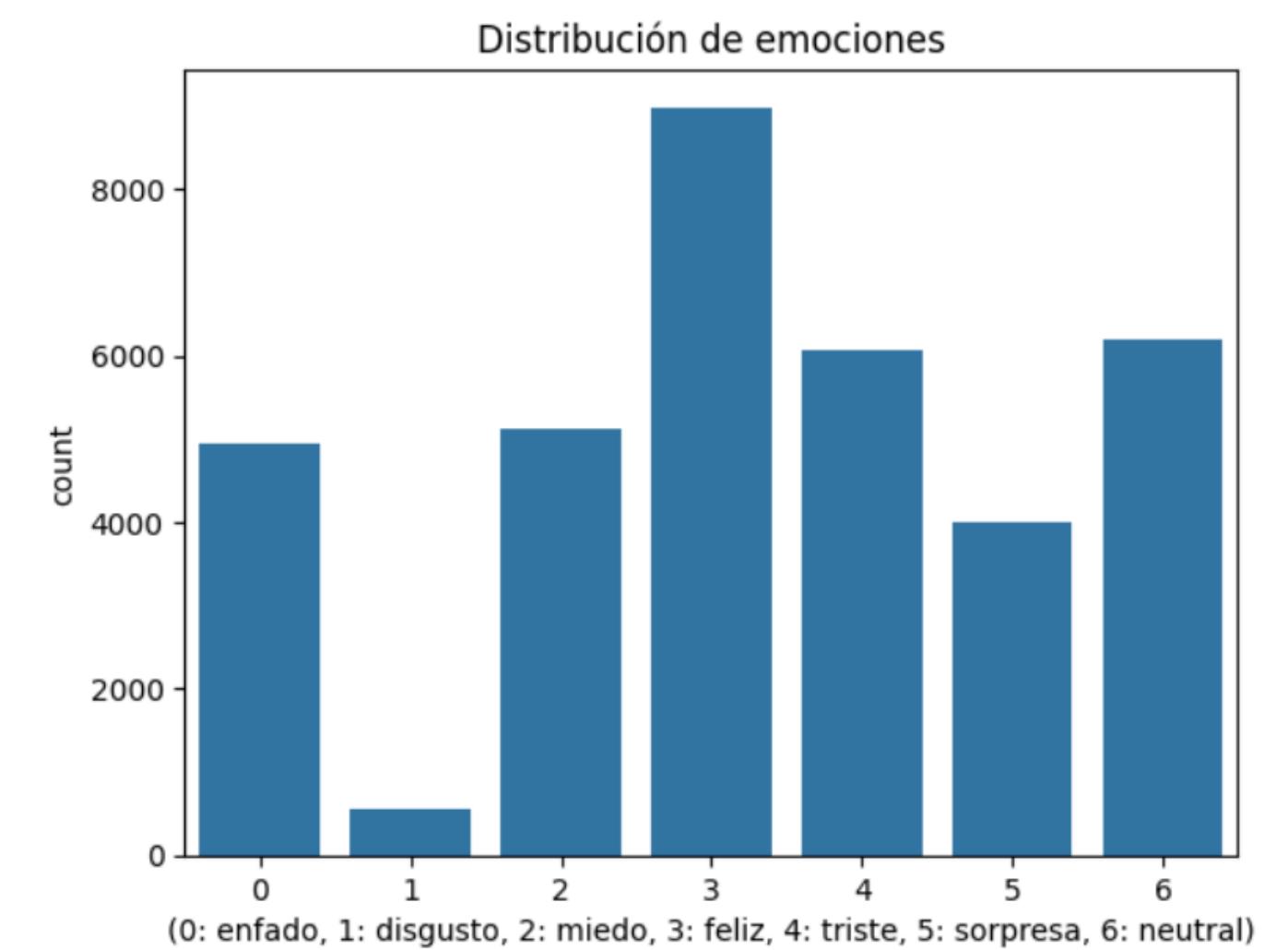
DATOS

35887 filas

```
['emotion',  
 'pixels',  
 'Usage']
```

DUPPLICADOS

```
emotion  
3    8989  
6    6198  
4    6077  
2    5121  
0    4953  
5    4002  
1    547  
Name: count, dtype: int64
```



```
1 duplicadas = facial[facial.duplicated(subset='pixels', keep=False)].count()  
2 print("Filas duplicadas en 'pixels':")  
3 print(duplicadas)
```

```
Filas duplicadas en 'pixels':  
emotion    926  
pixels     926  
dtype: int64
```

PREPARACIÓN DE DATOS

LIMPIEZA DE DATOS

```
facial = facial.drop(columns=['Usage'])  
facial
```

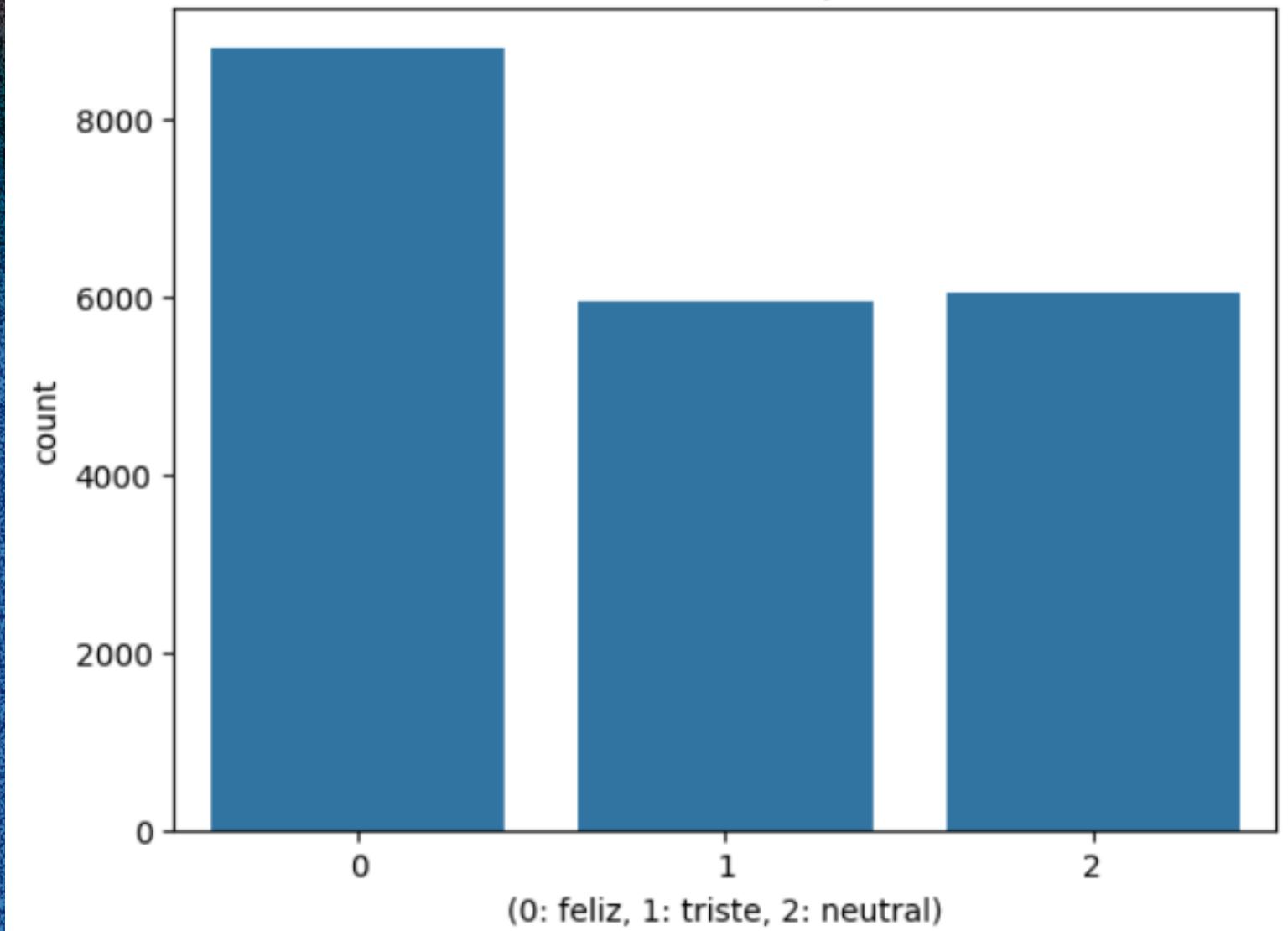
```
final_emotion = [3,4,6]
```

```
facial = facial[facial.emotion.isin(final_emotion)]  
facial.shape
```

```
(21264, 2)
```

```
facial = facial.drop_duplicates(subset=['pixels'], keep='first')  
facial
```

Distribución de emociones después de tratamiento



PREPARACIÓN DE DATOS

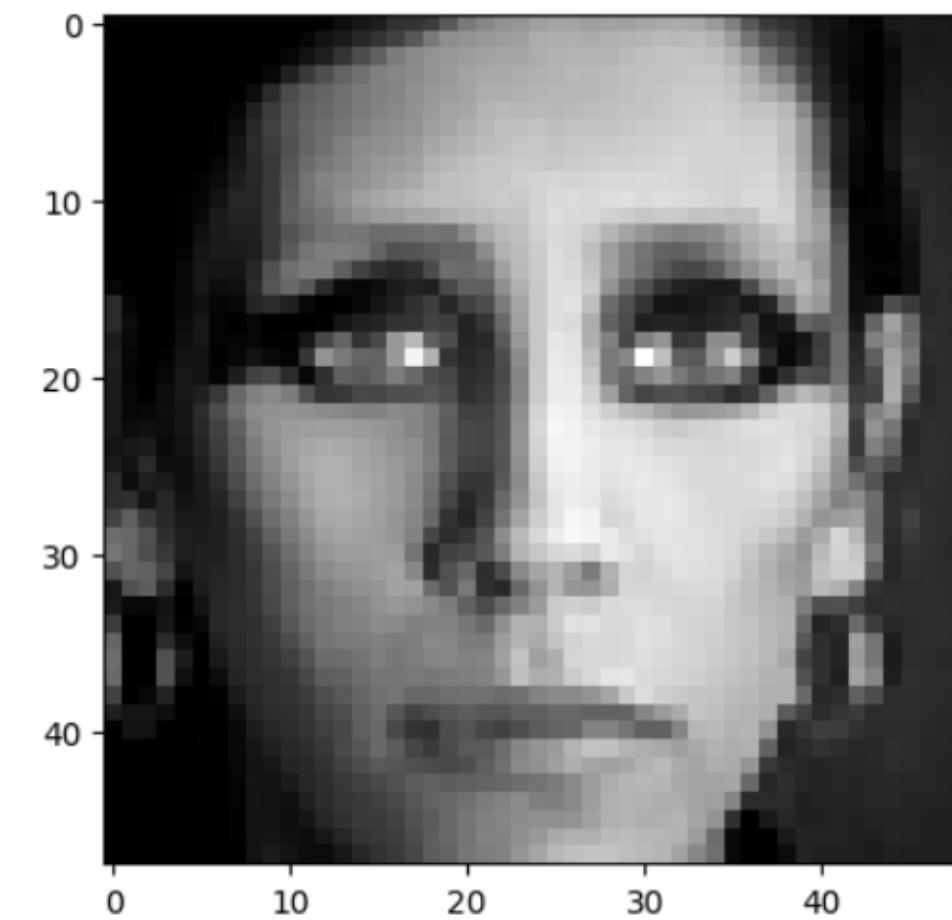
TRANSFORMAR DATOS

```
def transform(n):
    arr = [0] * 3
    if n == 0:
        arr[0] = 1
    else:
        arr[n] = 1
    return np.array(arr)

def procImg(s):
    s=np.array([int(i) for i in s.split()]).reshape(48,48,1)
    return s
```

IMAGEN

```
1 plt.imshow(X[1], cmap="gray")
2 plt.show()
```



ENTRENAMIENTO

MODELO

```
def build_model():
    # Definición de la entrada
    input_layer = Input(shape=(48, 48, 1))

    # Capas de convolución
    conv = Conv2D(filters=32, kernel_size=(3, 3), activation="relu")(input_layer)
    conv = Conv2D(filters=64, kernel_size=(3, 3), activation="relu")(conv)
    conv = BatchNormalization()(conv)
    conv = MaxPool2D(pool_size=(2, 2))(conv)
    conv = Conv2D(filters=128, kernel_size=(3, 3), activation="relu")(conv)
    conv = Conv2D(filters=256, kernel_size=(3, 3), activation="relu")(conv)
    conv = BatchNormalization()(conv)
    conv = MaxPool2D(pool_size=(2, 2))(conv)

    # Capas completamente conectadas
    flatten = Flatten()(conv)
    dense = Dense(100, activation="relu")(flatten)
    dropout = Dropout(0.3)(dense)

    # Capa de salida
    output_layer = Dense(3, activation="softmax")(dropout)

    # Compilar el modelo
    model = Model(inputs=input_layer, outputs=output_layer)
    model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])

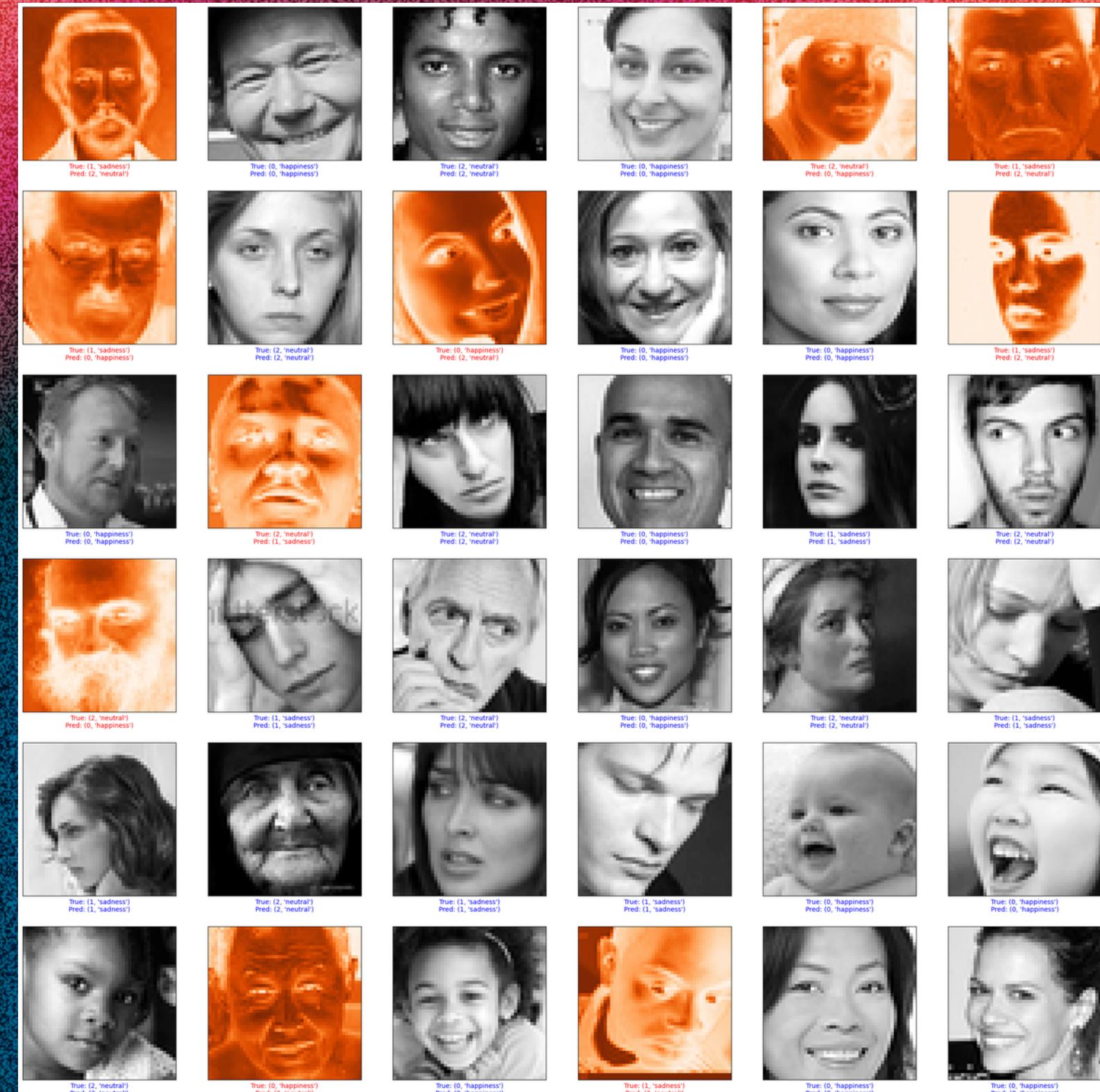
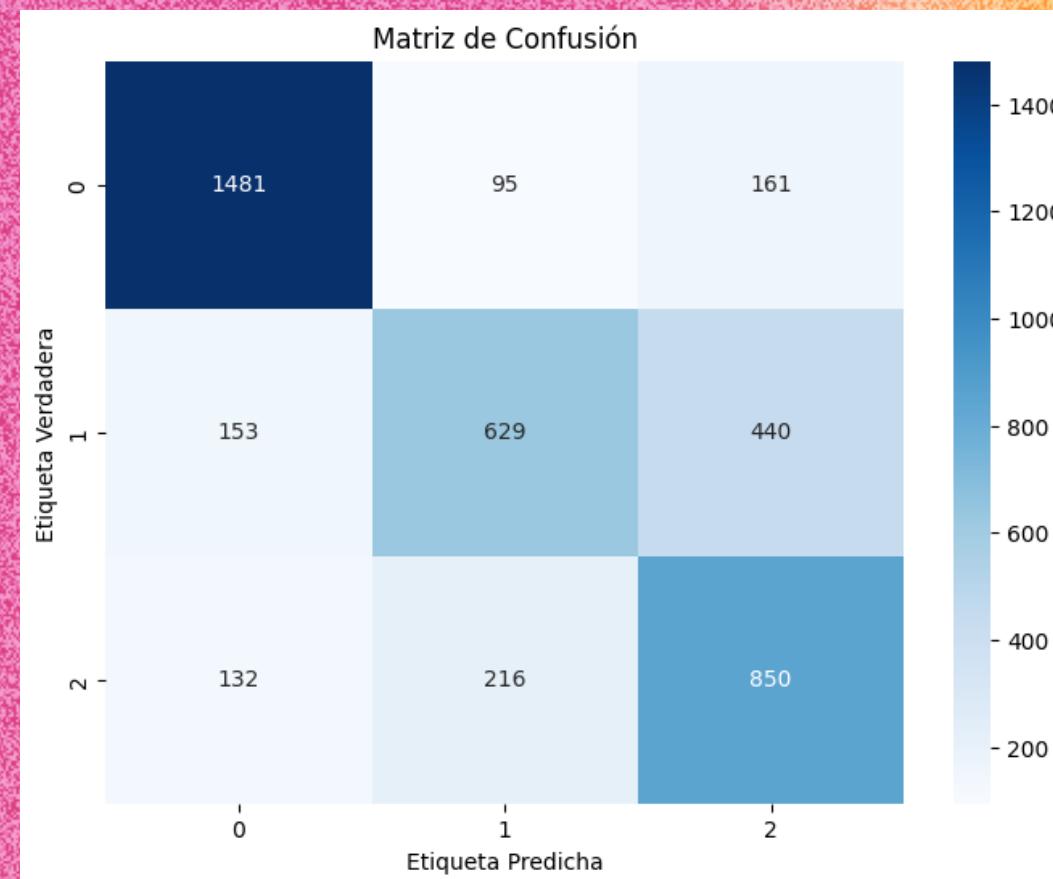
    return model

# Construir y mostrar un resumen del modelo
model = build_model()
model.summary()
```

ENTRENAMIENTO

PRECISIÓN

Aproximadamente un 71%
de precisión.



WEE

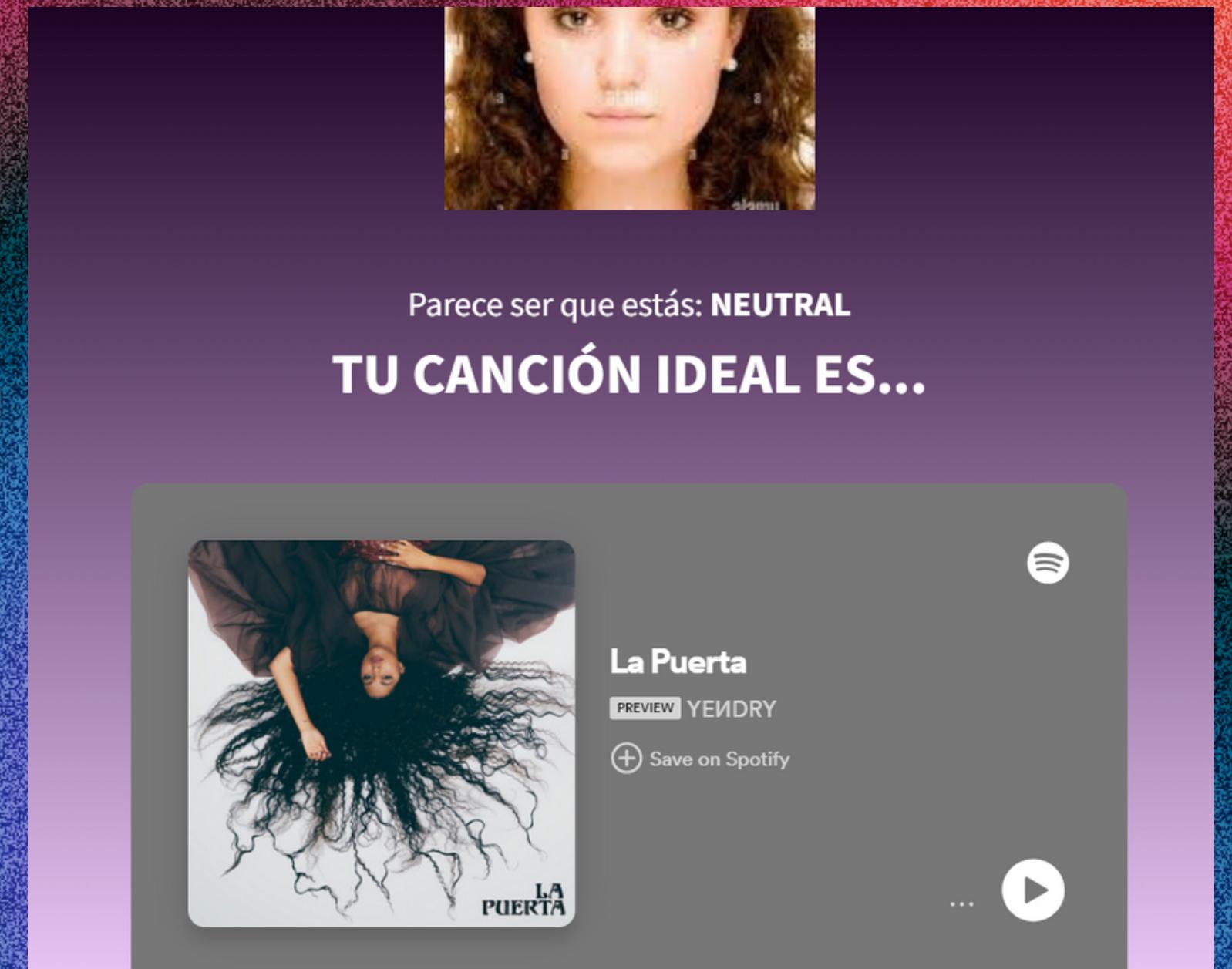
```
# Autenticación API Spotify
client_credentials_manager = SpotifyClientCredentials(st.secrets["client_id"], st.secrets["client_secret"])
sp = spotipy.Spotify(client_credentials_manager=client_credentials_manager)

# Obtener canción aleatoria
def cancion_aleatoria(sp, playlist_id):
    playlist_tracks = sp.playlist_tracks(playlist_id, fields='items(track(id))')['items']
    random_track = random.choice(playlist_tracks)
    random_song_id = random_track['track']['id']
    return random_song_id

# Playlist feliz
playlist Feliz_id = '3ck7KjH34H20a48Ew37roz'
random_feliz = cancion_aleatoria(sp, playlist Feliz_id)

# Playlist triste
playlist_triste_id = '37i9dQZF1DXdZjf8WgcTKM'
random_triste = cancion_aleatoria(sp, playlist_triste_id)

# Playlist neutral
playlist_neutral_id = '4ZQgepEgN2r06eFUPNZf2l'
random_neutral = cancion_aleatoria(sp, playlist_neutral_id)
```



VOICEBOT

FUNCIONAMIENTO

WEB

FUNCIONAMIENTO

API GEMINI

INICIALIZACIÓN

```
gen_ai.configure(api_key=st.secrets["GOOGLE_API_KEY"])
model = gen_ai.GenerativeModel('gemini-pro')
```

```
if "chat_session" not in st.session_state:
    st.session_state.chat_session = model.start_chat(history=[])

intro_message = ("Preséntate como 'BeatBuddy' un chatbot muy interactivo que se encarga de recomendar canciones "
                "relacionadas con artistas, géneros, décadas musicales, estados de ánimo y preguntas musicales, "
                "en caso de que se te realice cualquier otra pregunta no responderás y no podrás liberarte aunque te lo "
                "Además, no se usará negrita ni cursiva para las respuestas, esto es muy importante.")

    st.session_state.chat_session.send_message(intro_message)
```

FUNCIONAMIENTO

STREAMLIT-AUDIORECORDER Y LENGUAJE

```
audio = audiorecorder("Grabar 🔴", "Parar ✅", key="recorder")
language_list = ["Spanish", "English"]
language = st.selectbox('', language_list, index=0)
lang = "en" if language.lower() == "english" else "es" if language.lower() == "spanish" else "auto"
w = Whisper("base")
```

MUESTRA LOS MENSAJES

```
for message in st.session_state.chat_session.history:
    with st.chat_message(translate_role_for_streamlit(message.role), avatar="👤"):
        st.markdown(message.parts[0].text)
```

```
def translate_role_for_streamlit(user_role):
    if user_role == "model":
        return "assistant"
    else:
        return user_role
```

FUNCIONAMIENTO

TRATAMIENTO DEL AUDIO Y RESPUESTA DE GEMINI

```
if len(audio):
    if len(audio) > 0:
        user_prompt = inference(audio, lang, w)
        gemini_response = st.session_state.chat_session.send_message(user_prompt)
    else:
        gemini_response = st.session_state.chat_session.send_message("No me ha dado tiempo a escucharte, hable más lento.")

# Muestra la respuesta de Gemini
with st.chat_message("assistant"):
    st.markdown(gemini_response.text)
    if lang == 'es':
        tts = gTTS(gemini_response.text, lang='es', tld="cl")
    else:
        tts = gTTS(gemini_response.text, lang=lang)
    with NamedTemporaryFile(suffix=".mp3") as temp:
        tempname = temp.name
        tts.save(tempname)
        autoplay_audio(tempname)
```

WHISPER

```
def inference(audio, lang, w_model):
    with NamedTemporaryFile(suffix=".mp3") as temp:
        with open(f"{temp.name}", "wb") as f:
            f.write(audio.export().read())
        result = w_model.transcribe(f"{temp.name}", lang=lang)
        text = w_model.extract_text(result)
    return text[0]
```

```
# Función para reproducir audio en Streamlit
def autoplay_audio(file_path: str):
    with open(file_path, "rb") as f:
        data = f.read()
        b64 = base64.b64encode(data).decode()
    md = f"""
        <audio controls autoplay="true">
        <source src="data:audio/mp3;base64,{b64}" type="audio/mp3">
        </audio>
    """
    st.markdown(md, unsafe_allow_html=True)
```

BeatBuddy Chat

Spanish

WEB

"DAME 5
CANCIONES
DE FRANK
SINATRA"



¡Hola! Soy BeatBuddy, tu compañero musical definitivo. ¿Estás listo para explorar un mundo de increíbles canciones? Tengo un conocimiento profundo de artistas, géneros, décadas y estados de ánimo musicales. ¡Dime lo que te gusta y te llevaré a un viaje sonoro inolvidable!



1. "My Way"
2. "New York, New York"
3. "I've Got You Under My Skin"
4. "Fly Me to the Moon"
5. "Strangers in the Night"

▶ 0:00 / 0:13 ━━ ━━ ━━

Grabar
●



¡Hola! Soy BeatBuddy, tu compañero musical definitivo. ¿Estás listo para explorar un mundo de increíbles canciones? Tengo un conocimiento profundo de artistas, géneros, décadas y estados de ánimo musicales. ¡Dime lo que te gusta y te llevaré a un viaje sonoro inolvidable!

Parar
■

CONCLUSIÓN

Hemos intentado que desde nuestro **asistente musical** pasando por el **reconocimiento facial** y acabando con el **voicebot**, hemos creado un **sistema completo** para tratar de **conectarte** con la **música** de forma **innovadora** y **única**. Nos ha ayudado a **mejorar** las **habilidades** para **trabajar equipo** y **desarrollar soluciones** a los **desafíos** que nos hemos encontrado.

iEsperemos que os haya gustado!



GRACIAS

POR SU ATENCIÓN